Red Hat Enterprise Linux 5

Online Storage Reconfiguration Guide

For Red Hat Enterprise Linux 5

Edition 1

Last Updated: 2017-11-14

Red Hat Enterprise Linux 5 Online Storage Reconfiguration Guide

For Red Hat Enterprise Linux 5
Edition 1

Mike Christie
Engineering Services and Operations Kernel Storage
mchristi@redhat.com

Tom Coughlan
Engineering Services and Operations Kernel Storage
coughlan@redhat.com

Jacquelynn East
Engineering Services and Operations Content Services
jeast@redhat.com

Rob Evers
Engineering Services and Operations Kernel Storage
revers@redhat.com

Pasi Kärkkäinen
pasik@iki.fi

Red Hat Enterprise Linux Engineering

Development Community

Abstract

This document outlines the different procedures involved in reconfiguring storage devices while the system is running. The document currently addresses iSCSI and Fibre Channel interconnects. Other interconnects may be added in future versions of this document.

Table of Contents

INTRODUCTION

It is often desirable to add, remove or re-size storage devices while the operating system is running, and without rebooting. This manual outlines the procedures that may be used to reconfigure storage devices on Red Hat Enterprise Linux 5 host systems while the system is running. It covers iSCSI and Fibre Channel storage interconnects; other interconnect types may be added it the future.

The *Online Storage Reconfiguration guide*focuses on adding, removing, modifying, and monitoring storage devices. It does not discuss the Fibre Channel or iSCSI protocols in detail. For more information about these protocols, refer to other documentation.

This manual assumes that you have advanced working knowledge of Red Hat Enterprise Linux 5, along with first-hand experience in managing storage devices in Linux. Before consulting this book, verify if your host bus adapter vendor or hardware vendor have their own documentation. It is recommended that you consult such documents in conjunction with this manual.

NOTE

This manual makes reference to various `sysfs` objects. Red Hat advises that the `sysfs` object names and directory structure are subject to change in major Red Hat Enterprise Linux 5 releases. This is because the upstream Linux kernel does not provide a stable internal API. For guidelines on how to reference `sysfs` objects in a transportable way, refer to the document `Documentation/sysfs-rules.txt` in the kernel source tree for guidelines.

WARNING

Online storage reconfiguration must be done carefully. System failures or interruptions during the process can lead to unexpected results. Red Hat advises that you reduce system load to the maximum extent possible during the change operations. This will reduce the chance of I/O errors, out-of-memory errors, or similar errors occurring in the midst of a configuration change. The following sections provide more specific guidelines regarding this.

In addition, Red Hat recommends that you back up all data before performing online storage reconfiguration.

CHAPTER 1. ONLINE STORAGE RECONFIGURATION

1.1. FIBRE CHANNEL

This section discusses the Fibre Channel API, native Red Hat Enterprise Linux 5 Fibre Channel drivers, and the Fibre Channel capabilities of these drivers.

1.1.1. Fibre Channel API

Below is a list of `/sys/class/` directories that contain files used to provide the userspace API. In each item, host numbers are designated by ***H***, bus numbers are ***B***, targets are ***T***, logical unit numbers (LUNs) are ***L***, and remote port numbers are ***R***.

IMPORTANT

If your system is using multipath software, Red Hat recommends that you consult your hardware vendor before changing any of the values described in this section.

Transport: `/sys/class/fc_transport/targetH:B:T/`

- **`port_id`** — 24-bit port ID/address
- **`node_name`** — 64-bit node name
- **`port_name`** — 64-bit port name

Remote Port: `/sys/class/fc_remote_ports/rport-H:B-R/`

- **`port_id`**
- **`node_name`**
- **`port_name`**
- **`dev_loss_tmo`** — number of seconds to wait before marking a link as "bad". Once a link is marked bad, IO running on its corresponding path (along with any new IO on that path) will be failed.

  The default **`dev_loss_tmo`** value varies, depending on which driver/device is used. If a Qlogic adapter is used, the default is 35 seconds, while if an Emulex adapter is used, it is 30 seconds. The **`dev_loss_tmo`** value can be changed via the **`scsi_transport_fc`** module parameter **`dev_loss_tmo`**, although the driver can override this timeout value.

  The maximum **`dev_loss_tmo`** value is 600 seconds. If **`dev_loss_tmo`** is set to zero or any value greater than 600, the driver's internal timeouts will be used instead.
- **`fast_io_fail_tmo`** — length of time to wait before failing IO executed when a link problem is detected. IO that reaches the driver will fail. If IO is in a blocked queue, it will not be failed until **`dev_loss_tmo`** expires and the queue is unblocked.

Host: `/sys/class/fc_host/hostH/`

- **`port_id`**

- **`issue_lip`** – instructs the driver to rediscover remote ports.

1.1.2. Native Fibre Channel Drivers and Capabilities

Red Hat Enterprise Linux 5 ships with the following native fibre channel drivers:

- `lpfc`
- `qla2xxx`
- `zfcp`
- `mptfc`

Table 1.1, "Fibre-Channel API Capabilities" describes the different fibre-channel API capabilities of each native Red Hat Enterprise Linux 5 driver. X denotes support for the capability.

Table 1.1. Fibre-Channel API Capabilities

	`lpfc`	`qla2xxx`	`zfcp`	`mptfc`
Transport **`port_id`**	X	X	X	X
Transport **`node_name`**	X	X	X	X
Transport **`port_name`**	X	X	X	X
Remote Port **`dev_loss_tmo`**	X	X	X	X
Remote Port **`fast_io_fail_tmo`**	X	X [a]		
Host **`port_id`**	X	X	X	X
Host **`issue_lip`**	X	X		

[a] Supported as of Red Hat Enterprise Linux 5.4

1.2. ISCSI

This section describes the iSCSI API and the **`iscsiadm`** utility. Before using the **`iscsiadm`** utility, install the **`iscsi-initiator-utils`** package first; to do so, run **`yum install iscsi-initiator-utils`**.

In addition, the iSCSI service must be running in order to discover or log in to targets. To start the iSCSI service, run `service iscsi start`

1.2.1. iSCSI API

To get information about running sessions, run:

`iscsiadm -m session -P 3`

This command displays the session/device state, session ID (sid), some negotiated parameters, and the SCSI devices accessible through the session.

For shorter output (for example, to display only the sid-to-node mapping), run:

`iscsiadm -m session -P 0`

or

`iscsiadm -m session`

These commands print the list of running sessions with the format:

```
driver [sid] target_ip:port,target_portal_group_tag proper_target_name
```

For example:

`iscsiadm -m session`

```

tcp [2] 10.15.84.19:3260,2 iqn.1992-08.com.netapp:sn.33615311
tcp [3] 10.15.85.19:3260,3 iqn.1992-08.com.netapp:sn.33615311
```

For more information about the iSCSI API, refer to **`/usr/share/doc/iscsi-initiator-utils-`*`version`*`/README`**.

1.3. PERSISTENT NAMING

The operating system issues I/O to a storage device by referencing the path that is used to reach it. For SCSI devices, the path consists of the following:

- PCI identifier of the host bus adapter (HBA)
- channel number on that HBA
- the remote SCSI target address
- the Logical Unit Number (LUN)

This path-based address is not persistent. It may change any time the system is reconfigured (either by on-line reconfiguration, as described in this manual, or when the system is shutdown, reconfigured, and rebooted). It is even possible for the path identifiers to change when no physical reconfiguration has been done, as a result of timing variations during the discovery process when the system boots, or when a bus is re-scanned.

The operating system provides several non-persistent names to represent these access paths to storage devices. One is the **/dev/sd** name; another is the **major:minor** number. A third is a symlink maintained in the **/dev/disk/by-path/** directory. This symlink maps from the path identifier to the current **/dev/sd** name. For example, for a Fibre Channel device, the PCI info and ***Host*:*BusTarget*:*LUN*** info may appear as follows:

```
pci-0000:02:0e.0-scsi-0:0:0:0 -> ../../sda
```

For iSCSI devices, **by-path/** names map from the target name and portal information to the **sd** name.

It is generally *not* appropriate for applications to use these path-based names. This is because the storage device these paths reference may change, potentially causing incorrect data to be written to the device. Path-based names are also not appropriate for multipath devices, because the path-based names may be mistaken for separate storage devices, leading to uncoordinated access and unintended modifications of the data.

In addition, path-based names are system-specific. This can cause unintended data changes when the device is accessed by multiple systems, such as in a cluster.

For these reasons, several persistent, system-independent, methods for identifying devices have been developed. The following sections discuss these in detail.

1.3.1. WWID

The *World Wide Identifier* (WWID) can be used in reliably identifying devices. It is a persistent, system-independent ID that the SCSI Standard requires from all SCSI devices. The WWID identifier is guaranteed to be unique for every storage device, and independent of the path that is used to access the device.

This identifier can be obtained by issuing a SCSI Inquiry to retrieve the *Device Identification Vital Product Data* (page **0x83**) or *Unit Serial Number* (page **0x80**). The mappings from these WWIDs to the current **/dev/sd** names can be seen in the symlinks maintained in the **/dev/disk/by-id/** directory.

For example, a device with a page **0x83** identifier would have:

```
scsi-3600508b400105e210000900000490000 -> ../../sda
```

Or, a device with a page **0x80** identifier would have:

```
scsi-SSEAGATE_ST373453LW_3HW1RHM6 -> ../../sda
```

Red Hat Enterprise Linux 5 automatically maintains the proper mapping from the WWID-based device name to a current **/dev/sd** name on that system. Applications can use the **/dev/disk/by-id/** name to reference the data on the disk, even if the path to the device changes, and even when accessing the device from different systems.

If there are multiple paths from a system to a device, **device-mapper-multipath** uses the WWID to detect this. **Device-mapper-multipath** then presents a single "pseudo-device" in **/dev/mapper/wwid**, such as **/dev/mapper/3600508b400105df70000e00000ac0000**.

The command **multipath -l** shows the mapping to the non-persistent identifiers: ***Host*:*Channel*:*Target*:*LUN***, **/dev/sd** name, and the **major:minor** number.

```
3600508b400105df70000e00000ac0000 dm-2 vendor,product
[size=20G][features=1 queue_if_no_path][hwhandler=0][rw]
```

```
\_ round-robin 0 [prio=0][active]
 \_ 5:0:1:1 sdc 8:32  [active][undef]
 \_ 6:0:1:1 sdg 8:96  [active][undef]
\_ round-robin 0 [prio=0][enabled]
 \_ 5:0:0:1 sdb 8:16  [active][undef]
 \_ 6:0:0:1 sdf 8:80  [active][undef]
```

Device-mapper-multipath automatically maintains the proper mapping of each WWID-based device name to its corresponding `/dev/sd` name on the system. These names are persistent across path changes, and they are consistent when accessing the device from different systems.

When the `user_friendly_names` feature (of **device-mapper-multipath**) is used, the WWID is mapped to a name of the form `/dev/mapper/mpath`*n*. By default, this mapping is maintained in the file `/var/lib/multipath/bindings`. These `mpathn` names are persistent as long as that file is maintained.

WARNING

The multipath bindings file (by default, `/var/lib/multipath/bindings`) must be available at boot time. If `/var` is a separate filesystem from `/`, then you must change the default location of the file. For more information, refer to http://kbase.redhat.com/faq/docs/DOC-17650.

IMPORTANT

If you use `user_friendly_names`, then additional steps are required to obtain consistent names in a cluster. Refer to the Consistent Multipath Device Names section in the Using Device-Mapper Multipath book.

In addition to these persistent names provided by the system, you can also use **udev** rules to implement persistent names of your own, mapped to the WWID of the storage. For more information about this, refer to http://kbase.redhat.com/faq/docs/DOC-7319.

1.3.2. UUID and Other Persistent Identifiers

If a storage device contains a filesystem, then that filesystem may provide one or both of the following:

- *Universally Unique Identifier* (UUID)
- Filesystem label

These identifiers are persistent, and based on metadata written on the device by certain applications. They may also be used to access the device using the symlinks maintained by the operating system in the `/dev/disk/by-label/` (e.g. `boot -> ../../sda1`) and `/dev/disk/by-uuid/` (e.g. `f8bf09e3-4c16-4d91-bd5e-6f62da165c08 -> ../../sda1`) directories.

md and LVM write metadata on the storage device, and read that data when they scan devices. In each case, the metadata contains a UUID, so that the device can be identified regardless of the path (or system) used to access it. As a result, the device names presented by these facilities are persistent, as long as the metadata remains unchanged.

1.4. REMOVING A STORAGE DEVICE

Before removing access to the storage device itself, it is advisable to back up data from the device first. Afterwards, flush I/O and remove all operating system references to the device (as described below). If the device uses multipathing, then do this for the multipath "pseudo device" (Section 1.3.1, “WWID”) and each of the identifiers that represent a path to the device. If you are only removing a path to a multipath device, and other paths will remain, then the procedure is simpler, as described in Section 1.6, “Adding a Storage Device or Path” .

Removal of a storage device is not recommended when the system is under memory pressure, since the I/O flush will add to the load. To determine the level of memory pressure, run the command **`vmstat 1 100`**; device removal is not recommended if:

- Free memory is less than 5% of the total memory in more than 10 samples per 100 (the command **`free`** can also be used to display the total memory).
- Swapping is active (non-zero **`si`** and **`so`** columns in the **`vmstat`** output).

The general procedure for removing all access to a device is as follows:

Procedure 1.1. Ensuring a Clean Device Removal

1. Close all users of the device and backup device data as needed.
2. Use **`umount`** to unmount any file systems that mounted the device.
3. Remove the device from any **`md`** and LVM volume using it. If the device is a member of an LVM Volume group, then it may be necessary to move data off the device using the **`pvmove`** command, then use the **`vgreduce`** command to remove the physical volume, and (optionally) **`pvremove`** to remove the LVM metadata from the disk.
4. If the device uses multipathing, run **`multipath -l`** and note all the paths to the device. Afterwards, remove the multipathed device using **`multipath -f device`**.
5. Run **`blockdev –flushbufs`** ***`device`*** to flush any outstanding I/O to all paths to the device. This is particularly important for raw devices, where there is no **`umount`** or **`vgreduce`** operation to cause an I/O flush.
6. Remove any reference to the device's path-based name, like **`/dev/sd`**, **`/dev/disk/by-path`** or the **`major:minor`** number, in applications, scripts, or utilities on the system. This is important in ensuring that different devices added in the future will not be mistaken for the current device.
7. Finally, remove each path to the device from the SCSI subsystem. To do so, use the command **`echo 1 > /sys/block/`*`device-name`*`/device/delete`** where ***`device-name`*** may be **`sde`**, for example.

   Another variation of this operation is **`echo 1 > /sys/class/scsi_device/`*`h`*`:`*`c`*`:`*`t`*`:`*`l`*`/device/delete`**, where ***`h`*** is the HBA number, ***`c`*** is the channel on the HBA, ***`t`*** is the SCSI target ID, and ***`l`*** is the LUN.

NOTE

The older form of these commands, `echo "scsi remove-single-device 0 0 0 0" > /proc/scsi/scsi`, is deprecated.

You can determine the ***`device-name`***, HBA number, HBA channel, SCSI target ID and LUN for a device from various commands, such as `lsscsi`, `scsi_id`, `multipath -l`, and `ls -l /dev/disk/by-*`.

After performing Procedure 1.1, “Ensuring a Clean Device Removal” , a device can be physically removed safely from a running system. It is not necessary to stop I/O to other devices while doing so.

Other procedures, such as the physical removal of the device, followed by a rescan of the SCSI bus (as described in Section 1.8, “Scanning Storage Interconnects”) to cause the operating system state to be updated to reflect the change, are not recommended. This will cause delays due to I/O timeouts, and devices may be removed unexpectedly. If it is necessary to perform a rescan of an interconnect, it must be done while I/O is paused, as described in Section 1.8, “Scanning Storage Interconnects”.

1.5. REMOVING A PATH TO A STORAGE DEVICE

If you are removing a path to a device that uses multipathing (without affecting other paths to the device), then the general procedure is as follows:

Procedure 1.2. Removing a Path to a Storage Device

1. Remove any reference to the device's path-based name, like `/dev/sd` or `/dev/disk/by-path` or the `major:minor` number, in applications, scripts, or utilities on the system. This is important in ensuring that different devices added in the future will not be mistaken for the current device.

2. Take the path offline using `echo offline > /sys/block/sda/device/state`.

   This will cause any subsequent IO sent to the device on this path to be failed immediately. **Device-mapper-multipath** will continue to use the remaining paths to the device.

3. Remove the path from the SCSI subsystem. To do so, use the command `echo 1 > /sys/block/`***`device-name`***`/device/delete` where ***`device-name`*** may be `sde`, for example (as described in Procedure 1.1, “Ensuring a Clean Device Removal”).

After performing Procedure 1.2, “Removing a Path to a Storage Device” , the path can be safely removed from the running system. It is not necessary to stop I/O while this is done, as **device-mapper-multipath** will re-route I/O to remaining paths according to the configured path grouping and failover policies.

Other procedures, such as the physical removal of the cable, followed by a rescan of the SCSI bus to cause the operating system state to be updated to reflect the change, are not recommended. This will cause delays due to I/O timeouts, and devices may be removed unexpectedly. If it is necessary to perform a rescan of an interconnect, it must be done while I/O is paused, as described in Section 1.8, “Scanning Storage Interconnects”.

1.6. ADDING A STORAGE DEVICE OR PATH

When adding a device, be aware that the path-based device name (`/dev/sd` name, `major:minor` number, and `/dev/disk/by-path` name, for example) the system assigns to the new device may have been previously in use by a device that has since been removed. As such, ensure that all old references to the path-based device name have been removed. Otherwise, the new device may be mistaken for the old device.

Procedure 1.3. Add a storage device or path

1. The first step in adding a storage device or path is to physically enable access to the new storage device, or a new path to an existing device. This is done using vendor-specific commands at the Fibre Channel or iSCSI storage server. When doing so, note the LUN value for the new storage that will be presented to your host. If the storage server is Fibre Channel, also take note of the *World Wide Node Name* (WWNN) of the storage server, and determine whether there is a single WWNN for all ports on the storage server. If this is not the case, note the *World Wide Port Name* (WWPN) for each port that will be used to access the new LUN.

2. Next, make the operating system aware of the new storage device, or path to an existing device. The recommended command to use is:

   ```
   # echo "c t l" > /sys/class/scsi_host/hosth/scan
   ```

   In the previous command, ***h*** is the HBA number, ***c*** is the channel on the HBA, ***t*** is the SCSI target ID, and ***l*** is the LUN.

   NOTE

   The older form of this command, **`echo "scsi add-single-device 0 0 0 0" > /proc/scsi/scsi`**, is deprecated.

   a. In some Fibre Channel hardware, a newly created LUN on the RAID array may not be visible to the operating system until a *Loop Initialization Protocol* (LIP) operation is performed. Refer to Section 1.8, “Scanning Storage Interconnects” for instructions on how to do this.

      IMPORTANT

      It will be necessary to stop I/O while this operation is executed if an LIP is required.

   b. If a new LUN has been added on the RAID array but is still not being configured by the operating system, confirm the list of LUNs being exported by the array using the **`sg_luns`** command, part of the sg3_utils package. This will issue the **`SCSI REPORT LUNS`** command to the RAID array and return a list of LUNs that are present.

   For Fibre Channel storage servers that implement a single WWNN for all ports, you can determine the correct ***h***,***c***,and ***t*** values (i.e. HBA number, HBA channel, and SCSI target ID) by searching for the WWNN in **`sysfs`**. For example, if the WWNN of the storage server is **`0x5006016090203181`**, use:

   ```
   # grep 5006016090203181 /sys/class/fc_transport/*/node_name
   ```

   This should display output similar to the following:

   ```
   /sys/class/fc_transport/target5:0:2/node_name:0x5006016090203181
   /sys/class/fc_transport/target5:0:3/node_name:0x5006016090203181
   /sys/class/fc_transport/target6:0:2/node_name:0x5006016090203181
   ```

```
/sys/class/fc_transport/target6:0:3/node_name:0x5006016090203181
```

This indicates there are four Fibre Channel routes to this target (two single-channel HBAs, each leading to two storage ports). Assuming a LUN value is **56**, then the following command will configure the first path:

```
# echo "0 2 56" > /sys/class/scsi_host/host5/scan
```

This must be done for each path to the new device.

For Fibre Channel storage servers that do not implement a single WWNN for all ports, you can determine the correct HBA number, HBA channel, and SCSI target ID by searching for each of the WWPNs in **`sysfs`**.

Another way to determine the HBA number, HBA channel, and SCSI target ID is to refer to another device that is already configured on the same path as the new device. This can be done with various commands, such as **`lsscsi`**, **`scsi_id`**, **`multipath -l`**, and **`ls -l /dev/disk/by-*`**. This information, plus the LUN number of the new device, can be used as shown above to probe and configure that path to the new device.

3. After adding all the SCSI paths to the device, execute the **`multipath`** command, and check to see that the device has been properly configured. At this point, the device can be added to **`md`**, LVM, **`mkfs`**, or **`mount`**, for example.

If the steps above are followed, then a device can safely be added to a running system. It is not necessary to stop I/O to other devices while this is done. Other procedures involving a rescan (or a reset) of the SCSI bus, which cause the operating system to update its state to reflect the current device connectivity, are not recommended while storage I/O is in progress.

1.7. TECHNOLOGY PREVIEW: FIBRE-CHANNEL OVER ETHERNET INTERFACE

The fcoe-utilspackage, which provides a Fibre-Channel Over Ethernet interface, is enabled as a Technology Preview in Red Hat Enterprise Linux 5.10. However, this Technology Preview excludes the dcdb package, which provides the data center bridging demon.

IMPORTANT

For further information about Technology Previews, refer to https://access.redhat.com/support/offerings/techpreview/.

1.8. SCANNING STORAGE INTERCONNECTS

There are several commands available that allow you to reset and/or scan one or more interconnects, potentially adding and removing multiple devices in one operation. This type of scan can be disruptive, as it can cause delays while I/O operations timeout, and remove devices unexpectedly. As such, Red Hat recommends that this type of scan be used *only when necessary*. In addition, the following restrictions must be observed when scanning storage interconnects:

1. All I/O on the effected interconnects must be paused and flushed before executing the procedure, and the results of the scan checked before I/O is resumed.

2. As with removing a device, interconnect scanning is not recommended when the system is under memory pressure. To determine the level of memory pressure, run the command

`vmstat 1 100`; interconnect scanning is not recommended if free memory is less than 5% of the total memory in more than 10 samples per 100. It is also not recommended if swapping is active (non-zero `si` and `so` columns in the `vmstat` output). The command `free` can also display the total memory.

The following commands can be used to scan storage interconnects.

`echo "1" > /sys/class/fc_host/host/issue_lip`

This operation performs a *Loop Initialization Protocol* (*LIP*) and then scans the interconnect and causes the SCSI layer to be updated to reflect the devices currently on the bus. A LIP is, essentially, a bus reset, and will cause device addition and removal. This procedure is necessary to configure a new SCSI target on a Fibre Channel interconnect.

Bear in mind that `issue_lip` is an asynchronous operation. The command may complete before the entire scan has completed. You must monitor `/var/log/messages` to determine when it is done.

The `lpfc` and `qla2xxx` drivers support `issue_lip`. For more information about the API capabilities supported by each driver in Red Hat Enterprise Linux, refer to Table 1.1, “Fibre-Channel API Capabilities”.

`/usr/bin/rescan-scsi-bus.sh`

This script is included in Red Hat Enterprise Linux 5.4 and all future updates. By default, this script scans all the SCSI buses on the system, updating the SCSI layer to reflect new devices on the bus. The script provides additional options to allow device removal and the issuing of LIPs. For more information about this script (including known issues), refer to Section 1.15, “Adding/Removing a Logical Unit Through rescan-scsi-bus.sh”.

`echo "- - -" > /sys/class/scsi_host/host`*`h`*`/scan`

This is the same command described in Section 1.6, “Adding a Storage Device or Path” to add a storage device or path. In this case, however, the channel number, SCSI target ID, and LUN values are replaced by wildcards. Any combination of identifiers and wildcards is allowed, allowing you to make the command as specific or broad as needed. This procedure will add LUNs, but not remove them.

`rmmod driver-name` or `modprobe driver-name`

These commands completely re-initialize the state of all interconnects controlled by the driver. Although this is extreme, it may be appropriate in some situations. This may be used, for example, to re-start the driver with a different module parameter value.

WARNING

Various problems have been seen when unloading and reloading drivers. Although some of these problems have been addressed, the problems will be present in previous releases. Additionally, current releases may have unresolved problems. Driver loading and unloading should not be done in a production environment.

1.9. ISCSI DISCOVERY CONFIGURATION

The default iSCSI configuration file is `/etc/iscsi/iscsid.conf`. This file contains iSCSI settings used by **iscsid** and **iscsiadm**.

During target discovery, the **iscsiadm** tool uses the settings in `/etc/iscsi/iscsid.conf` to create two types of records:

Node records in `/var/lib/iscsi/nodes`

When logging into a target, **iscsiadm** uses the settings in this file.

Discovery records in `/var/lib/iscsi/discovery_type`

When performing discovery to the same destination, **iscsiadm** uses the settings in this file.

Before using different settings for discovery, delete the current discovery records (i.e. `/var/lib/iscsi/discovery_type`) first. To do this, use the following command:

```
iscsiadm -m discovery -t discovery_type -p target_IP:port -o delete
```
[1]

Here, ***discovery_type*** can be either **sendtargets**, **isns**, or **fw**. [2]

There are two ways to reconfigure discovery record settings:

- Edit the `/etc/iscsi/iscsid.conf` file directly prior to performing a discovery. Discovery settings use the prefix **discovery**; to view them, run:

  ```
  iscsiadm -m discovery -t discovery_type -p target_IP:port
  ```

- Alternatively, **iscsiadm** can also be used to directly change discovery record settings, as in:

  ```
  iscsiadm -m discovery -t discovery_type -p target_IP:port -o update -n
  setting -v %value
  ```

  Refer to **man iscsiadm** for more information on available ***setting***s and valid ***value***s for each.

After configuring discovery settings, any subsequent attempts to discover new targets will use the new settings. Refer to Section 1.11, “Scanning iSCSI Interconnects” for details on how to scan for new iSCSI targets.

For more information on configuring iSCSI target discovery, refer to the **man** pages of **iscsiadm** and **iscsid**. The `/etc/iscsi/iscsid.conf` file also contains examples on proper configuration syntax.

1.10. CONFIGURING ISCSI OFFLOAD AND INTERFACE BINDING

This chapter describes how to set up iSCSI interfaces in order to bind a session to a NIC port when using software iSCSI. It also describes how to set up interfaces for use with network devices that support offloading; namely, devices from Chelsio, Broadcom and ServerEngines.

The network subsystem can be configured to determine the path/NIC that iSCSI interfaces should use for binding. For example, if portals and NICs are set up on different subnets, then it is not necessary to manually configure iSCSI interfaces for binding.

Before attempting to configure an iSCSI interface for binding, run the following command first:

```
ping -I ethX target_IP[1]
```

If **ping** fails, then you will not be able to bind a session to a NIC. If this is the case, check the network settings first.

1.10.1. Viewing Available iface Configurations

Red Hat Enterprise Linux 5.5 supports iSCSI offload and interface binding for the following iSCSI initiator implementations:

- *Software iSCSI* — like the **scsi_tcp** and **ib_iser** modules, this stack allocates an iSCSI host instance (i.e. **scsi_host**) per session, with a single connection per session. As a result, **/sys/class_scsi_host** and **/proc/scsi** will report a **scsi_host** for each connection/session you are logged into.

- *Offload iSCSI* — like the Chelsio **cxgb3i**, Broadcom **bnx2i** and ServerEngines **be2iscsi** modules, this stack allocates a **scsi_host** for each PCI device. As such, each port on a host bus adapter will show up as a different PCI device, with a different **scsi_host** per HBA port.

To manage both types of initiator implementations, **iscsiadm** uses the **iface** structure. With this structure, an **iface** configuration must be entered in **/var/lib/iscsi/ifaces** for each HBA port, software iSCSI, or network device (**ethX**) used to bind sessions.

To view available **iface** configurations, run **iscsiadm -m iface**. This will display **iface** information in the following format:

```
iface_name
transport_name,hardware_address,ip_address,net_ifacename,initiator_name
```

Refer to the following table for an explanation of each value/setting.

Table 1.2. iface Settings

Setting	Description
iface_name	**iface** configuration name.
transport_name	Name of driver
hardware_address	MAC address
ip_address	IP address to use for this port
net_iface_name	Name used for the **vlan** or alias binding of a software iSCSI session. For iSCSI offloads, **net_iface_name** will be **<empty>** because this value is not persistent across reboots.

Setting	Description
initiator_name	This setting is used to override a default name for the initiator, which is defined in **/etc/iscsi/initiatorname.iscsi**

The following is a sample output of the **iscsiadm -m iface** command:

```
iface0 qla4xxx,00:c0:dd:08:63:e8,20.15.0.7,default,iqn.2005-
06.com.redhat:madmax
iface1 qla4xxx,00:c0:dd:08:63:ea,20.15.0.9,default,iqn.2005-
06.com.redhat:madmax
```

For software iSCSI, each **iface** configuration must have a unique name (with less than 65 characters). The **iface_name** for network devices that support offloading appears in the format ***transport_name*.*hardware_name***.

For example, the sample output of **iscsiadm -m iface** on a system using a Chelsio network card might appear as:

```
default tcp,<empty>,<empty>,<empty>,<empty>
iser iser,<empty>,<empty>,<empty>,<empty>
cxgb3i.00:07:43:05:97:07 cxgb3i,00:07:43:05:97:07,<empty>,<empty>,<empty>
```

It is also possible to display the settings of a specific **iface** configuration in a more friendly way. To do so, use the option **-I *iface_name***. This will display the settings in the following format:

```
iface.setting = value
```

Using the previous example, the **iface** settings of the same Chelsio video card (i.e. **iscsiadm -m iface -I cxgb3i.00:07:43:05:97:07**) would appear as:

```
# BEGIN RECORD 2.0-871
iface.iscsi_ifacename = cxgb3i.00:07:43:05:97:07
iface.net_ifacename = <empty>
iface.ipaddress = <empty>
iface.hwaddress = 00:07:43:05:97:07
iface.transport_name = cxgb3i
iface.initiatorname = <empty>
# END RECORD
```

1.10.2. Configuring an iface for Software iSCSI

As mentioned earlier, an **iface** configuration is required for each network object that will be used to bind a session.

To create an **iface** configuration for software iSCSI, run the following command:

iscsiadm -m iface -I *iface_name* --op=new

This will create a new *empty* `iface` configuration with a specified ***`iface_name`***. If an existing `iface` configuration already has the same ***`iface_name`***, then it will be overwritten with a new, empty one.

To configure a specific setting of an `iface` configuration, use the following command:

```
iscsiadm -m iface -I iface_name --op=update -n iface.setting -v hw_address
```

For example, to set the MAC address (`hardware_address`) of `iface0` to `00:0F:1F:92:6B:BF`, run:

```
iscsiadm -m iface -I iface0 - -op=update -n iface.hwaddress -v
00:0F:1F:92:6B:BF
```

WARNING

Do not use `default` or `iser` as `iface` names. Both strings are special values used by `iscsiadm` for backward compatibility. Any manually-created `iface` configurations named `default` or `iser` will disable backwards compatibility.

1.10.3. Configuring an iface for iSCSI Offload

By default, `iscsiadm` will create an `iface` configuration for each Chelsio, Broadcom, and ServerEngines port. To view available `iface` configurations, use the same command for doing so in software iSCSI, i.e. `iscsiadm -m iface`.

Before using the `iface` of a network card for iSCSI offload, first set the IP address (***`target_IP`***[1]) that the device should use. For ServerEngines devices that use the `be2iscsi` driver (i.e. ServerEngines iSCSI HBAs), the IP address is configured in the ServerEngines BIOS setup screen.

For Chelsio and Broadcom devices, the procedure for configuring the IP address is the same as for any other `iface` setting. So to configure the IP address of the `iface`, use:

```
iscsiadm -m iface -I iface_name -o update -n iface.ipaddress -v target_IP
```

For example, to set the `iface` IP address of a Chelsio card (with `iface` name `cxgb3i.00:07:43:05:97:07`) to `20.15.0.66`, use:

```
iscsiadm -m iface -I cxgb3i.00:07:43:05:97:07 -o update -n iface.ipaddress -
v 20.15.0.66
```

1.10.4. Binding/Unbinding an iface to a Portal

Whenever `iscsiadm` is used to scan for interconnects, it will first check the `iface.transport` settings of each `iface` configuration in `/var/lib/iscsi/ifaces`. The `iscsiadm` utility will then bind discovered portals to any `iface` whose `iface.transport` is `tcp`.

This behavior was implemented for compatibility reasons. To override this, use the `-I` ***`iface_name`*** to specify which portal to bind to an `iface`, as in:

```
iscsiadm -m discovery -t st -p target_IP:port -I iface_name -P 1[1]
```

By default, the **iscsiadm** utility will not automatically bind any portals to **iface** configurations that use offloading. This is because such **iface** configurations will not have **iface.transport** set to **tcp**. As such, the **iface** configurations of Chelsio, Broadcom, and ServerEngines ports need to be manually bound to discovered portals.

It is also possible to prevent a portal from binding to any existing **iface**. To do so, use **default** as the ***iface_name***, as in:

iscsiadm -m discovery -t st -p *IP:port* -I default -P 1

To remove the binding between a target and **iface**, use:

iscsiadm -m node -targetname *proper_target_name* -I iface0 --op=delete[3]

To delete all bindings for a specific **iface**, use:

iscsiadm -m node -I *iface_name* --op=delete

To delete bindings for a specific portal (e.g. for EqualLogic targets), use:

iscsiadm -m node -p *IP:port* -I *iface_name* --op=delete

NOTE

If there are no **iface** configurations defined in **/var/lib/iscsi/iface** and the **-I** option is not used, **iscsiadm** will allow the network subsystem to decide which device a specific portal should use.

1.11. SCANNING ISCSI INTERCONNECTS

For iSCSI, if the targets send an iSCSI async event indicating new storage is added, then the scan is done automatically. Cisco MDS™ and EMC Celerra™ support this feature.

However, if the targets do not send an iSCSI async event, you need to manually scan them using the **iscsiadm** utility. Before doing so, however, you need to first retrieve the proper **--targetname** and the **--portal** values. If your device model supports only a single logical unit and portal per target, use **iscsiadm** to issue a **sendtargets** command to the host, as in:

iscsiadm -m discovery -t sendtargets -p *target_IP:port*[1]

The output will appear in the following format:

```
target_IP:port,target_portal_group_tag proper_target_name
```

For example, on a target with a ***proper_target_name*** of **iqn.1992-08.com.netapp:sn.33615311** and a ***target_IP:port*** of **10.15.85.19:3260**, the output may appear as:

```
10.15.84.19:3260,2 iqn.1992-08.com.netapp:sn.33615311
10.15.85.19:3260,3 iqn.1992-08.com.netapp:sn.33615311
```

In this example, the target has two portals, each using ***target_ip:port***s of **10.15.84.19:3260** and **10.15.85.19:3260**.

To see which **`iface`** configuration will be used for each session, add the **`-P 1`** option. This option will print also session information in tree format, as in:

```
    Target: proper_target_name
        Portal: target_IP:port,target_portal_group_tag
           Iface Name: iface_name
```

For example, with **`iscsiadm -m discovery -t sendtargets -p 10.15.85.19:3260 -P 1`**, the output may appear as:

```
Target: iqn.1992-08.com.netapp:sn.33615311
    Portal: 10.15.84.19:3260,2
       Iface Name: iface2
    Portal: 10.15.85.19:3260,3
       Iface Name: iface2
```

This means that the target **`iqn.1992-08.com.netapp:sn.33615311`** will use **`iface2`** as its **`iface`** configuration.

With some device models (e.g. from EMC and Netapp), however, a single target may have multiple logical units and/or portals. In this case, issue a **`sendtargets`** command to the host first to find new portals on the target. Then, rescan the existing sessions using:

`iscsiadm -m session --rescan`

You can also rescan a specific session by specifying the session's ***`SID`*** value, as in:

`iscsiadm -m session -r` ***`SID`*** **`--rescan`**[4]

If your device supports multiple targets, you will need to issue a **`sendtargets`** command to the hosts to find new portals for *each* target. Then, rescan existing sessions to discover new logical units on existing sessions (i.e. using the **`--rescan`** option).

IMPORTANT

The **`sendtargets`** command used to retrieve **`--targetname`** and **`--portal`** values overwrites the contents of the **`/var/lib/iscsi/nodes`** database. This database will then be repopulated using the settings in **`/etc/iscsi/iscsid.conf`**. However, this will not occur if a session is currently logged in and in use.

To safely add new targets/portals or delete old ones, use the **`-o new`** or **`-o delete`** options, respectively. For example, to add new targets/portals without overwriting **`/var/lib/iscsi/nodes`**, use the following command:

```
iscsiadm -m discovery -t st -p target_IP -o new
```

To delete **`/var/lib/iscsi/nodes`** entries that the target did not display during discovery, use:

```
iscsiadm -m discovery -t st -p target_IP -o delete
```

You can also perform both tasks simultaneously, as in:

```
iscsiadm -m discovery -t st -p target_IP -o delete -o new
```

The **`sendtargets`** command will yield the following output:

```
ip:port,target_portal_group_tag proper_target_name
```

For example, given a device with a single target, logical unit, and portal, with **`equallogic-iscsi1`** as your ***`target_name`***, the output should appear similar to the following:

```
10.16.41.155:3260,0 iqn.2001-05.com.equallogic:6-8a0900-ac3fe0101-
63aff113e344a4a2-dl585-03-1
```

Note that ***`proper_target_name`*** and ***`ip:port,target_portal_group_tag`*** are identical to the values of the same name in Section 1.2.1, “iSCSI API”.

At this point, you now have the proper **`--targetname`** and **`--portal`** values needed to manually scan for iSCSI devices. To do so, run the following command:

`iscsiadm --mode node --targetname` ***`proper_target_name`*** **`--portal`** ***`ip:port,target_portal_group_tag`***

Using our previous example (where ***`proper_target_name`*** is **`equallogic-iscsi1`**), the full command would be:

`iscsiadm --mode node --targetname \ iqn.2001-05.com.equallogic:6-8a0900-ac3fe0101-63aff113e344a4a2-dl585-03-1 \ --portal 10.16.41.155:3260,0` [5]

NOTE

It is possible to append a **`--login`** option to a discovery command. This combines the two steps of discovering target portals and logging in to them. For example, **`iscsiadm --mode node --targetname`** ***`proper_target_name`*** **`--portal`** ***`ip:port,target_portal_group_tag`*** **`\ --login`**[5]

1.12. LOGGING IN TO AN ISCSI TARGET

As mentioned in Section 1.2, “iSCSI”, the iSCSI service must be running in order to discover or log into targets. To start the iSCSI service, run:

```
service iscsi start
```

When this command is executed, the iSCSI **`init`** scripts will automatically log into targets where the **`node.startup`** setting is configured as **`automatic`**. This is the default value of **`node.startup`** for all targets.

To prevent automatic login to a target, set **`node.startup`** to **`manual`**. To do this, run the following command:

```
iscsiadm -m node --targetname proper_target_name -p target_IP:port -o update
-n node.startup -v manual
```

Deleting the entire record will also prevent automatic login. To do this, run:

```
iscsiadm -m node --targetname proper_target_name -p target_IP:port -o delete
```

To automatically mount a file system from an iSCSI device on the network, add a partition entry for the mount in **`/etc/fstab`** with the **`_netdev`** option. For example, to automatically mount the iSCSI device **sdb** to **`/mount/iscsi`** during startup, add the following line to **`/etc/fstab`**:

```
/dev/sdb /mnt/iscsi ext3 _netdev 0 0
```

To manually log in to an iSCSI target, use the following command:

```
iscsiadm -m node --targetname proper_target_name -p target_IP:port -l
```

NOTE

The ***`proper_target_name`*** and ***`target_IP:port`*** refer to the full name and IP address/port combination of a target. For more information, refer to Section 1.2.1, “iSCSI API” and Section 1.11, “Scanning iSCSI Interconnects”.

1.13. ONLINE LOGICAL UNITS

This chapter will cover resizing online logical units, including resizing fibre channel logical units, resizing an iSCSI logical units, and updating the size of a multipath device, as well as changing the read/write state of onlin logical units.

1.13.1. Resizing an Online Logical Unit

In most cases, fully resizing an online *logical unit* involves two things: resizing the logical unit itself and reflecting the size change in the corresponding multipath device (if multipathing is enabled on the system).

To resize the online logical unit, start by modifying the logical unit size through the array management interface of your storage device. This procedure differs with each array; as such, consult your storage array vendor documentation for more information on this.

NOTE

In order to resize an online file system, the file system must not reside on a partitioned device.

1.13.1.1. Resizing Fibre Channel Logical Units

After modifying the online logical unit size, re-scan the logical unit to ensure that the system detects the updated size. To do this for Fibre Channel logical units, use the following command:

```
echo 1 > /sys/block/sdX/device/rescan
```

IMPORTANT

To re-scan fibre channel logical units on a system that uses multipathing, execute the aforementioned command for each sd device (i.e. `sd1`, `sd2`, and so on) that represents a path for the multipathed logical unit. To determine which devices are paths for a multipath logical unit, use `multipath -ll`; then, find the entry that matches the logical unit being resized. It is advisable that you refer to the WWID of each entry to make it easier to find which one matches the logical unit being resized.

1.13.1.2. Resizing an iSCSI Logical Unit

After modifying the online logical unit size, re-scan the logical unit to ensure that the system detects the updated size. To do this for iSCSI devices, use the following command:

```
iscsiadm -m node --targetname target_name -R
```
[1]

Replace ***`target_name`*** with the name of the target where the device is located.

NOTE

You can also re-scan iSCSI logical units using the following command:

```
iscsiadm -m node -R -I interface
```

Replace *`interface`* with the corresponding interface name of the resized logical unit (for example, `iface0`). This command performs two operations:

- It scans for new devices in the same way that the command `echo "- - -" > /sys/class/scsi_host/host/scan` does (refer to Section 1.11, "Scanning iSCSI Interconnects").
- It re-scans for new/modified logical units the same way that the command `echo 1 > /sys/block/sdX/device/rescan` does. Note that this command is the same one used for re-scanning fibre-channel logical units.

1.13.1.3. Updating the Size of Your Multipath Device

If multipathing is enabled on your system, you will also need to reflect the change in logical unit size to the logical unit's corresponding multipath device (*after* resizing the logical unit). For Red Hat Enterprise Linux 5.3 (or later), you can do this through `multipathd`. To do so, first ensure that `multipathd` is running using `service multipathd status`. Once you've verified that `multipathd` is operational, run the following command:

```
multipathd -k"resize map multipath_device"
```

The *`multipath_device`* variable is the corresponding multipath entry of your device in `/dev/mapper`. Depending on how multipathing is set up on your system, *`multipath_device`* can be either of two formats:

- `mpathX`, where *X* is the corresponding entry of your device (for example, `mpath0`)
- a WWID; for example, `3600508b400105e210000900000490000`

To determine which multipath entry corresponds to your resized logical unit, run `multipath -ll`. This displays a list of all existing multipath entries in the system, along with the major and minor numbers of their corresponding devices.

IMPORTANT

Do not use `multipathd -k"resize map multipath_device"` if there are any commands queued to *`multipath_device`*. That is, do not use this command when the `no_path_retry` parameter (in `/etc/multipath.conf`) is set to `"queue"`, and there are no active paths to the device.

If your system is using Red Hat Enterprise Linux 5.0-5.2, you will need to perform the following procedure to instruct the `multipathd` daemon to recognize (and adjust to) the changes you made to the resized logical unit:

Procedure 1.4. Resizing the Corresponding Multipath Device (Required for Red Hat Enterprise Linux 5.0 - 5.2)

1. Dump the device mapper table for the multipathed device using:

dmsetup table *multipath_device*

2. Save the dumped device mapper table as ***table_name***. This table will be re-loaded and edited later.

3. Examine the device mapper table. Note that the first two numbers in each line correspond to the start and end sectors of the disk, respectively.

4. Suspend the device mapper target:

   dmsetup suspend *multipath_device*

5. Open the device mapper table you saved earlier (i.e. ***table_name***). Change the second number (i.e. the disk end sector) to reflect the new number of 512 byte sectors in the disk. For example, if the new disk size is 2GB, change the second number to 4194304.

6. Reload the modified device mapper table:

   dmsetup reload *multipath_device* *table_name*

7. Resume the device mapper target:

   dmsetup resume *multipath_device*

For more information about multipathing, refer to the Using Device-Mapper Multipath guide (in http://www.redhat.com/docs/manuals/enterprise/).

1.14. CHANGING THE READ/WRITE STATE OF AN ONLINE LOGICAL UNIT

Certain storage devices provide the user with the ability to change the state of the device from Read/Write (R/W) to Read-Only (RO), and from RO to R/W. This is typically done through a management interface on the storage device. The operating system will not automatically update its view of the state of the device when a change is made. Follow the procedures described in this chapter to make the operating system aware of the change.

Run the following command, replacing XYZ with the desired device designator, to determine the operating system's current view of the R/W state of a device:

```
# blockdev --getro /dev/sdXYZ
```

The following command is also available for Red Hat Enterprise Linux 6:

```
# cat /sys/block/sdXYZ/ro 1 = read-only 0 = read-write
```

When using multipath, refer to the *ro* or *rw* field in the second line of output from the **multipath -ll** command. For example:

```
36001438005deb4710000500000640000 dm-8 GZ,GZ500
[size=20G][features=0][hwhandler=0][ro]
\_ round-robin 0 [prio=200][active]
 \_ 6:0:4:1  sdax 67:16  [active][ready]
 \_ 6:0:5:1  sday 67:32  [active][ready]
\_ round-robin 0 [prio=40][enabled]
 \_ 6:0:6:1  sdaz 67:48  [active][ready]
 \_ 6:0:7:1  sdba 67:64  [active][ready]
```

To change the R/W state, use the following procedure:

Procedure 1.5. Change the R/W state

1. To move the device from RO to R/W, see step 2.

   To move the device from R/W to RO, ensure no further writes will be issued. Do this by stopping the application, or through the use of an appropriate, application-specific action.

   Ensure that all outstanding write I/Os are complete with the following command:

   ```
   # blockdev -flushbufs /dev/device
   ```

   Replace *device* with the desired designator; for a device mapper multipath, this is the entry for your device in **`dev/mapper`**. For example, **`/dev/mapper/`*`mpath3`***.

2. Use the management interface of the storage device to change the state of the logical unit from R/W to RO, or from RO to R/W. The procedure for this differs with each array. Consult applicable storage array vendor documentation for more information.

3. Perform a re-scan of the device to update the operating system's view of the R/W state of the device. If using a device mapper multipath, perform this re-scan for each path to the device before issuing the command telling multipath to reload its device maps.

   This process is explained in further detail in Section 1.14.1, “Rescanning logical units”.

1.14.1. Rescanning logical units

After modifying the online logical unit Read/Write state, as described in Section 1.14, “Changing the Read/Write State of an Online Logical Unit”, re-scan the logical unit to ensure the system detects the updated state with the following command:

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

To re-scan logical units on a system that uses multipathing, execute the above command for each sd device that represents a path for the multipathed logical unit. For example, run the command on sd1, sd2 and all other sd devices. To determine which devices are paths for a multipath unit, use **`multipath -11`**, then find the entry that matches the logical unit to be changed.

Example 1.1. Use of the `multipath -11` command

For example, the **`multipath -11`** above shows the path for the LUN with WWID 36001438005deb4710000500000640000. In this case, enter:

```
# echo 1 > /sys/block/sdax/device/rescan
# echo 1 > /sys/block/sday/device/rescan
# echo 1 > /sys/block/sdaz/device/rescan
# echo 1 > /sys/block/sdba/device/rescan
```

1.14.2. Updating the R/W state of a multipath device

If multipathing is enabled, after rescanning the logical unit, the change in its state will need to be reflected in the logical unit's corresponding multipath drive. Do this by reloading the multipath device maps with the following command:

```
# multipath -r
```

The **`multipath -11`** command can then be used to confirm the change.

1.14.3. Documentation

Further information can be found in the Red Hat Knowledgebase. To access this, navigate to https://www.redhat.com/wapps/sso/login.html?redirect=https://access.redhat.com/knowledge/ and log in. Then access the article at https://access.redhat.com/kb/docs/DOC-32850.

1.15. ADDING/REMOVING A LOGICAL UNIT THROUGH RESCAN-SCSI-BUS.SH

The **`sg3_utils`** package provides the **`rescan-scsi-bus.sh`** script, which can automatically update the logical unit configuration of the host as needed (after a device has been added to the system). The **`rescan-scsi-bus.sh`** script can also perform an **`issue_lip`** on supported devices. For more information about how to use this script, refer to **`rescan-scsi-bus.sh --help`**.

To install the **`sg3_utils`** package, run **`yum install sg3_utils`**.

Known Issues With rescan-scsi-bus.sh

When using the **`rescan-scsi-bus.sh`** script, take note of the following known issues:

- A race condition requires that **`rescan-scsi-bus.sh`** be run twice if logical units are mapped for the first time. During the first scan, **`rescan-scsi-bus.sh`** only adds **`LUN0`**; all other logical units are added in the second scan.
- A bug in the **`rescan-scsi-bus.sh`** script incorrectly executes the functionality for recognizing a change in logical unit size when the **`--remove`** option is used.
- The **`rescan-scsi-bus.sh`** script does not recognize ISCSI logical unit removals.

1.16. MODIFYING LINK LOSS BEHAVIOR

This section describes how to modify the link loss behavior of devices that use either fibre channel or iSCSI protocols.

1.16.1. Fibre Channel

If a driver implements the Transport **`dev_loss_tmo`** callback, access attempts to a device through a link will be blocked when a transport problem is detected. To verify if a device is blocked, run the following command:

```
cat /sys/block/device/device/state
```

This command will return **`blocked`** if the device is blocked. If the device is operating normally, this command will return **`running`**.

Procedure 1.6. Determining The State of a Remote Port

1. To determine the state of a remote port, run the following command:

   ```
   cat /sys/class/fc_remote_port/rport-H:B:R/port_state
   ```

2. This command will return **`Blocked`** when the remote port (along with devices accessed through it) are blocked. If the remote port is operating normally, the command will return **`Online`**.

3. If the problem is not resolved within **`dev_loss_tmo`** seconds, the rport and devices will be unblocked and all IO running on that device (along with any new IO sent to that device) will be failed.

Procedure 1.7. Changing `dev_loss_tmo`

- To change the **`dev_loss_tmo`** value, **echo** in the desired value to the file. For example, to set **`dev_loss_tmo`** to 30 seconds, run:

  ```
  echo 30 > /sys/class/fc_remote_port/rport-H:B:R/dev_loss_tmo
  ```

For more information about **`dev_loss_tmo`**, refer to Section 1.1.1, "Fibre Channel API".

When a device is blocked, the fibre channel class will leave the device as is; i.e. **`/dev/sdx`** will remain **`/dev/sdx`**. This is because the **`dev_loss_tmo`** expired. If the link problem is fixed at a later time, operations will continue using the same SCSI device and device node name.

Fibre Channel: `remove_on_dev_loss`

If you prefer that devices are removed at the SCSI layer when links are marked bad (i.e. expired after **`dev_loss_tmo`** seconds), you can use the **`scsi_transport_fc`** module parameter **`remove_on_dev_loss`**. When a device is removed at the SCSI layer while **`remove_on_dev_loss`** is in effect, the device will be added back once all transport problems are corrected.

WARNING

The use of **`remove_on_dev_loss`** is not recommended, as removing a device at the SCSI layer does not automatically unmount any file systems from that device. When file systems from a removed device are left mounted, the device may not be properly removed from multipath or RAID devices.

Further problems may arise from this if the upper layers are not hotplug-aware. This is because the upper layers may still be holding references to the state of the device before it was originally removed. This can cause unexpected behavior when the device is added again.

1.16.2. iSCSI Settings With `dm-multipath`

If **dm-multipath** is implemented, it is advisable to set iSCSI timers to immediately defer commands to the multipath layer. To configure this, nest the following line under **device {** in **/etc/multipath.conf**:

```
features  "1 queue_if_no_path"
```

This ensures that I/O errors are retried and queued if all paths are failed in the **dm-multipath** layer.

You may need to adjust iSCSI timers further to better monitor your SAN for problems. Available iSCSI timers you can configure are *NOP-Out Interval/Timeouts* and **replacement_timeout**, which are discussed in the following sections.

1.16.2.1. NOP-Out Interval/Timeout

To help monitor problems the SAN, the iSCSI layer sends a NOP-Out request to each target. If a NOP-Out request times out, the iSCSI layer responds by failing any running commands and instructing the SCSI layer to requeue those commands when possible.

When **dm-multipath** is being used, the SCSI layer will fail those running commands and defer them to the multipath layer. The multipath layer then retries those commands on another path. If **dm-multipath** is *not* being used, those commands are retried five times before failing altogether.

Intervals between NOP-Out requests are 10 seconds by default. To adjust this, open **/etc/iscsi/iscsid.conf** and edit the following line:

```
node.conn[0].timeo.noop_out_interval = [interval value]
```

Once set, the iSCSI layer will send a NOP-Out request to each target every *[interval value]* seconds.

By default, NOP-Out requests time out in 10 seconds[6]. To adjust this, open **/etc/iscsi/iscsid.conf** and edit the following line:

```
node.conn[0].timeo.noop_out_timeout = [timeout value]
```

This sets the iSCSI layer to timeout a NOP-Out request after *[timeout value]* seconds.

Note that this will not work for iSCSI targets already logged in or discovered once. In this case run **iscsiadm** to update configuration values explicitly.

```
iscsiadm -m node -T $target_name -p $target_ip:$port  -o update -n \
node.session.timeo.replacement_timeout -v $timeout_value
```

SCSI Error Handler

If the SCSI Error Handler is running, running commands on a path will not be failed immediately when a NOP-Out request times out on that path. Instead, those commands will be failed *after* **replacement_timeout** seconds. For more information about **replacement_timeout**, refer to Section 1.16.2.2, "**replacement_timeout**".

To verify if the SCSI Error Handler is running, run:

```
iscsiadm -m session -P 3
```

1.16.2.2. `replacement_timeout`

`replacement_timeout` controls how long the iSCSI layer should wait for a timed-out path/session to reestablish itself before failing any commands on it. The default **`replacement_timeout`** value is 120 seconds.

To adjust **`replacement_timeout`**, open **`/etc/iscsi/iscsid.conf`** and edit the following line:

```
node.session.timeo.replacement_timeout = [replacement_timeout]
```

The **`1 queue_if_no_path`** option in **`/etc/multipath.conf`** sets iSCSI timers to immediately defer commands to the multipath layer (refer to Section 1.16.2, "iSCSI Settings With **`dm-multipath`**"). This setting prevents I/O errors from propagating to the application; because of this, you can set **`replacement_timeout`** to 15-20 seconds.

By configuring a lower **`replacement_timeout`**, I/O is quickly sent to a new path and executed (in the event of a NOP-Out timeout) while the iSCSI layer attempts to re-establish the failed path/session. If all paths time out, then the multipath and device mapper layer will internally queue I/O based on the settings in **`/etc/multipath.conf`** instead of **`/etc/iscsi/iscsid.conf`**.

Note that this will not work for iSCSI targets already logged in or discovered once. In this case run **`iscsiadm`** to update configuration values explicitly.

```
iscsiadm -m node -T $target_name -p $target_ip:$port  -o update -n \
node.session.timeo.replacement_timeout -v $timeout_value
```

IMPORTANT

Whether your considerations are failover speed or security, the recommended value for **`replacement_timeout`** will depend on other factors. These factors include the network, target, and system workload. As such, it is recommended that you thoroughly test any new configurations to **`replacements_timeout`** before applying it to a mission-critical system.

The iSCSI tools do not use the **`iscsid.conf`** settings for portals that have already been discovered and logged into. To modify the settings of a portal that has already been discovered and set up, run:

```
iscsiadm -m node -T $target_name -p $target_ip:$port  -o update -n \
node.session.timeo.replacement_timeout -v $timeout_value
```

The above command will modify the portal's record so the next time the iSCSI tools log in, that value will be used.

NOTE

The value on a running session cannot be modified. For the value to be used, either restart the iSCSI service, or run the **`iscsiadm logout`** command on that session, then log back in.

To set the value to be the new default for all newly discovered portals, set the value in **`iscsid.conf`**. Next time the **`iscsiadm discovery`** command is run and portals are found, the new value will be used.

1.16.3. iSCSI Root

When accessing the root partition directly through a iSCSI disk, the iSCSI timers should be set so that iSCSI layer has several chances to try to reestablish a path/session. In addition, commands should not be quickly re-queued to the SCSI layer. This is the opposite of what should be done when **`dm-multipath`** is implemented.

To start with, NOP-Outs should be disabled. You can do this by setting both NOP-Out interval and timeout to zero. To set this, open **`/etc/iscsi/iscsid.conf`** and edit as follows:

```
node.conn[0].timeo.noop_out_interval = 0
node.conn[0].timeo.noop_out_timeout = 0
```

In line with this, **`replacement_timeout`** should be set to a high number. This will instruct the system to wait a long time for a path/session to reestablish itself. To adjust **`replacement_timeout`**, open **`/etc/iscsi/iscsid.conf`** and edit the following line:

```
node.session.timeo.replacement_timeout = replacement_timeout
```

After configuring **`/etc/iscsi/iscsid.conf`**, you must perform a re-discovery of the affected storage. This will allow the system to load and use any new values in **`/etc/iscsi/iscsid.conf`**. For more information on how to discover iSCSI devices, refer to Section 1.11, "Scanning iSCSI Interconnects".

Configuring Timeouts for a Specific Session

You can also configure timeouts for a specific session and make them non-persistent (instead of using **`/etc/iscsi/iscsid.conf`**). To do so, run the following command (replace the variables accordingly):

```
iscsiadm -m node -T target_name -p target_IP:port -o update -n
node.session.timeo.replacement_timeout -v $timeout_value
```

IMPORTANT

The configuration described here is recommended for iSCSI sessions involving root partition access. For iSCSI sessions involving access to other types of storage (namely, in systems that use **`dm-multipath`**), refer to Section 1.16.2, "iSCSI Settings With `dm-multipath`".

1.17. CONTROLLING THE SCSI COMMAND TIMER AND DEVICE STATUS

The Linux SCSI layer sets a timer on each command. When this timer expires, the SCSI layer will quiesce the *host bus adapter* (HBA) and wait for all outstanding commands to either time out or complete. Afterwards, the SCSI layer will activate the driver's error handler.

When the error handler is triggered, it attempts the following operations in order (until one successfully executes):

1. Abort the command.
2. Reset the device.
3. Reset the bus.

4. Reset the host.

If all of these operations fail, the device will be set to the **`offline`** state. When this occurs, all IO to that device will be failed, until the problem is corrected and the user sets the device to **`running`**.

The process is different, however, if a device uses the fibre channel protocol and the **`rport`** is blocked. In such cases, the drivers wait for several seconds for the **`rport`** to become online again before activating the error handler. This prevents devices from becoming offline due to temporary transport problems.

Device States

To display the state of a device, use:

`cat /sys/block/`*`device-name`*`/device/state`

To set a device to **`running`** state, use:

`echo running > /sys/block/`*`device-name`*`/device/state`

Command Timer

To control the command timer, you can write to **`/sys/block/`*`device-name`*`/device/timeout`**. To do so, run:

`echo `*`value`*` /sys/block/`*`device-name`*`/device/timeout`

Here, ***`value`*** is the timeout value (in seconds) you want to implement.

Alternatively, you can also modify the timeout **udev** rule. To do so, open **`/etc/udev/rules.d/50-udev.rules`**. You should find the following lines:

```
ACTION=="add", SUBSYSTEM=="scsi" , SYSFS{type}=="0|7|14", \
RUN+="/bin/sh -c 'echo 60 > /sys$$DEVPATH/timeout'"
```

`echo 60` refers to the timeout length, in seconds; in this case, timeout is set at 60 seconds. Replace this value with your desired timeout length.

Note that the default timeout for normal file system commands is 60 seconds when **udev** is being used. If **udev** is not in use, the default timeout is 30 seconds.

1.18. TROUBLESHOOTING

This section provides solution to common problems users experience during online storage reconfiguration.

Logical unit removal status is not reflected on the host.

When a logical unit is deleted on a configured filer, the change is not reflected on the host. In such cases, **`lvm`** commands will hang indefinitely when **`dm-multipath`** is used, as the logical unit has now become *stale*.

To work around this, perform the following procedure:

Procedure 1.8. Working Around Stale Logical Units

1. Determine which `mpath` link entries in `/etc/lvm/cache/.cache` are specific to the stale logical unit. To do this, run the following command:

   ```
   ls -l /dev/mpath | grep stale-logical-unit
   ```

2. For example, if ***`stale-logical-unit`*** is 3600d0230003414f30000203a7bc41a00, the following results may appear:

   ```
   lrwxrwxrwx 1 root root 7 Aug  2 10:33
   /3600d0230003414f30000203a7bc41a00 -> ../dm-4
   lrwxrwxrwx 1 root root 7 Aug  2 10:33
   /3600d0230003414f30000203a7bc41a00p1 -> ../dm-5
   ```

   This means that 3600d0230003414f30000203a7bc41a00 is mapped to two `mpath` links: **`dm-4`** and **`dm-5`**.

3. Next, open `/etc/lvm/cache/.cache`. Delete all lines containing ***`stale-logical-unit`*** and the `mpath` links that ***`stale-logical-unit`*** maps to.

   Using the same example in the previous step, the lines you need to delete are:

   ```
   /dev/dm-4
   /dev/dm-5
   /dev/mapper/3600d0230003414f30000203a7bc41a00
   /dev/mapper/3600d0230003414f30000203a7bc41a00p1
   /dev/mpath/3600d0230003414f30000203a7bc41a00
   /dev/mpath/3600d0230003414f30000203a7bc41a00p1
   ```

[1] The ***`target_IP`*** and ***`port`*** variables refer to the IP address and port combination of a target/portal, respectively. For more information, refer to Section 1.2.1, "iSCSI API" and Section 1.11, "Scanning iSCSI Interconnects".

[2] For details on different types of discovery, refer to the *DISCOVERY TYPES* section of **`man iscsiadm`**.

[3] Refer to Section 1.11, "Scanning iSCSI Interconnects" for information on ***`proper_target_name`***.

[4] For information on how to retrieve a session's SID value, refer to Section 1.2.1, "iSCSI API".

[5] This is a single command split into multiple lines, to accommodate printed and PDF versions of this document. All concatenated lines — preceded by the backslash (\) — should be treated as one command, sans backslashes.

[6] Prior to Red Hat Enterprise Linux 5.4, the default NOP-Out requests time out was 15 seconds.

APPENDIX A. REVISION HISTORY

Revision 1-26	**Tue Apr 15 2014**	**Jacquelynn East**

Removed a paragraph in Known Issues with rescan-scsi-bus.sh that is no longer an issue BZ#840471.

Revision 1-24	**Tue Oct 1 2013**	**Jacquelynn East**

Build for 5.10 GA.

Revision 1-20	**Tue May 28 2013**	**Laura Bailey**

BZ#679275 Noted fcoe-utils as a Technology Preview.

Revision 1-19	**Fri May 17 2013**	**Jacquelynn East**

BZ#853780 fixed typo, removed --login from scanning command, added note to explain --login use.

Revision 1-17	**Wed May 15 2013**	**Jacquelynn East**

BZ#820284 added warning to modprobe information.

Revision 1-16	**Wed Jul 11 2012**	**Jacquelynn East**

BZ#610908 added extra chapter

Revision 1-15	**Fri May 25 2012**	**Jacquelynn East**

BZ#784408 added extra info to procedure

Revision 1-13	**Thu May 24 2012**	**Jacquelynn East**

BZ#610908 Add section on changing R/W states

Revision 1-9	**Fri Sep 16 2011**	**Jacquelynn East**

BZ#690065

INDEX