Dell EMC SC Series with Red Hat Enterprise Linux 7x

Abstract

This paper provides guidelines for volume discovery, multipath configuration, file system, queue depth management, and performance tuning of Red Hat® Enterprise Linux® (RHEL) 7.x with Dell™ Storage Center Operating System (SCOS) 7.x.x.

January 2018

Revisions

Date	Description
July 2014	Initial release
May 2015	Introduce connectivity to Dell Storage SCv2000
July 2017	Refresh for RHEL 7.3
January 2018	Minor updates

Acknowledgements

Author: Steven Lemons

Table of contents

Executive summary

Red Hat® Enterprise Linux® (RHEL) is an extremely robust and scalable enterprise-class operating system. Correctly configured using the best practices presented in this paper, the RHEL operating system provides an optimized experience with Dell EMC™ SC Series storage. These recommendations include guidelines for volume discovery, multipath configuration, file system, queue depth management, and performance tuning.

This paper presents features of RHEL 7.x with Dell™ Storage Center Operating System (SCOS) version 7.x.x. There are often various methods for accomplishing the described tasks, and this paper provides a starting point for end users and system administrators.

This paper focuses almost exclusively on the command line interface (CLI) because it is the most universally applicable across UNIX® and Linux distributions.

1 Overview

SC Series storage provides Linux-compatible and SCSI-3 compliant disk volumes that remove the complexity of allocating, administering, using, and protecting mission-critical data. A properly configured SC Series array removes the need for cumbersome physical disk configuration exercises and management along with complex RAID configuration practices. The SC Series array also provides RAID 10 speed and reliability at the storage layer so that volumes do not need to be further RAID-managed within the Linux operating system layer.

The full range of Linux utilities such as mirroring, backup, multiple file systems, multipath, boot from SAN, and disaster recovery can be used with SC Series volumes.

1.1 New in RHEL 7.x

RHEL 7.x delivers dramatic improvements in reliability, performance, and scalability. This paper identifies improved RHEL 7.x features that provide simpler use and integration with the SC Series storage and features.

- XFS is the new default filesystem for boot, root, and user data partitions.
- The XFS filesystem size limit has increased from 100TB to 500TB, and the ext4 filesystem size limit has increased from 16TB to 50TB.
- New software implementation of the iSCSI and Fibre Channel over Ethernet (FCoE) targets are located in the kernel, instead of the user space.

2 Volume management

Understanding how volumes are managed in Linux requires basic understanding of the /sys pseudo filesystem. The /sys filesystem is a structure of files that allow interaction with various elements of the kernel and modules. While the read-only files store current values, read/write files trigger events with the correct commands. Generally, the **cat** and **echo** commands are used with a redirect as STDIN instead of being opened with a traditional text editor.

To interact with the HBAs (FC, iSCSI, and SAS), commands are issued against special files located in the /sys/class/scsi_host/ folder. Each port on a multiport card represents a unique HBA, and each HBA has its own hostX folder containing files for issuing scans and reading HBA parameters. The folder layout, files, and functionality can vary depending on the HBA vendor or type (for example, QLogic® Fibre Channel, Emulex® Fibre Channel, software-iSCSI based HBAs, or Dell EMC 12Gbps SAS HBAs).

2.1 Scanning for new volumes

The driver modules required for the QLogic 24xx/25xx Series HBAs and the Emulex HBAs are merged into base kernel code. The following instructions apply to the default HBA driver modules. If the vendor (QLogic, Emulex) proprietary driver has been used, consult the vendor-specific documentation for instructions and further details.

This script identifies the major revision number of the Linux operating system and applies the echo command to the respective hostX devices within the /sys/class/scsi_host/ folder. This script scans the HBA ports, and discovers and identifies existing and new volumes presented to the host from the storage array. This script can be used to discover both FC and iSCSI devices presented to the host and it applies for RHEL versions 5.x–7.x.

Note: STDOUT is not generated from this script. Check the contents of **/var/log/messages** or the output from the **dmesg** or **lsscsi** commands to identify any newly discovered volumes.

Rescanning the HBAs while mapping and discovering new volumes does not have any negative impact on the host.

```
#!/bin/bash

OSMajor=`uname -r | awk -F. '{print $(NF-1)}'`

      echo "INFO: OS Major rev. ${OSMajor} detected!"

      if [ "${OSMajor}" = "el7" -o "${OSMajor}" = "el6" ]; then
      for i in /sys/class/scsi_host/*
         do
         echo "- - -" >> ${i}/scan
         done
      elif [ "$(uname -r | awk -F. '{print $(NF)}')" = "el5" ]; then
         echo "INFO: OS Major rev. el5 detected instead!"
         for i in /sys/class/scsi_host/*
         do
         echo 1 >> ${i}/issue_lip
               echo "- - -" >> ${i}/scan
         done
```

```
      else
         echo "WARN: OSMajor parameter of unknown value, exit 1"
         exit 1
      fi
```

The following shows the sample output from this script:

```
# ./scan_bus.sh
INFO: OS Major Rev. el7 detected!
```

Alternatively, the installation of the **sg3_utils** package would provide a native Red Hat command **rescan-scsi-bus.**sh located in the **/usr/bin** folder.

```
# /usr/bin/rescan-scsi-bus.sh --alltargets
Scanning SCSI subsystem for new devices
[snip]
0 new or changed device(s) found.
0 remapped or resized device(s) found.
0 device(s) removed.
```

2.2 Partitions

Partitions (and partition tables) are not required for volumes other than the boot volume; use SC Series volumes as whole drives. This leverages the native strengths of the SC Series wide striping of volumes across all disks in the tier where the volume is provisioned. With a default RHEL 7.x installation, the boot volume is partitioned into two: an XFS filesystem (default) is applied to the boot partition, and the other partition is managed by Logical Volume Manager.

```
# parted
GNU Parted 3.1
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) select /dev/sda
Using /dev/sda
(parted) print
Model: SEAGATE ST9146803SS (scsi)
Disk /dev/sda: 147GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  525MB   524MB   primary  xfs          boot
 2      525MB   147GB   146GB   primary               lvm
```

2.3 Logical Volume Manager

Logical Volume Manager (LVM) can be applied and used to manage volumes in Linux. It installs LVM metadata (such as LVM signatures) to the volumes and uniquely identifies the physical volumes (PV), logical volumes (LV), and volume groups (VG) accordingly. Mounting the snapshot view volumes on the same host as the source volume is not recommended since it would result in duplicate LVM signatures.

As a best practice, use SC Series volumes as whole drives. However, LVM may still be used if it can provide features or benefits that are not provided by the storage layer.

2.3.1 LVM configuration and SCSI UNMAP/TRIM

If LVM is used with SC Series volumes, LVM can be configured to respect SCSI UNMAP/TRIM commands and pass these commands back to the SC Series storage.

This scenario applies where logical volumes are removed from the member volume group. The space recovered from this removed logical volume correlates to the SC Series pages that are freed and returned to the page pool.

Configuring LVM is performed by editing the /etc/lvm/lvm.conf file.

1. Edit the **/etc/lvm/lvm.conf** file.
2. Change the key value pair **issue_discards = 0** to **issue_discards =1**.
3. Save the changes and exit the editor session.

2.4 SCSI UNMAP/TRIM and filesystems

The behaviors of SCSI UNMAP/TRIM can also be configured to operate in the filesystem layer (such as ext4, xfs, or btrfs).

This scenario applies where files and directories are removed from an ext4, xfs, or btrfs filesystem. The space recovered from these removed files and directories correlates to the SC Series storage pages that are freed and returned to the page pool.

Enabling SCSI UNMAP/TRIM functions on the filesystem is done using the **mount** command. This mount command parameter is applied in a similar manner regardless of filesystem type (ext4, xfs, or btrfs).

```
# mount -o discard /dev/mapper/<volume_name> /<mountpoint_name>
```

This mount parameter can also be made persistent across reboots by adding the appropriate flag to the **/etc/fstab** file for the filesystems.

```
# cat /etc/fstab
[snip]
/dev/mapper/VolGroup00-home /home                   xfs   defaults          1 2
/dev/mapper/VolGroup00-swap swap                    swap  defaults          0 0
/dev/mapper/<volume_name>  /<mountpoint_name>       xfs   defaults,discard  0 0
[snip]
```

2.5 Persistent device management

Volumes discovered in Linux are given device designations such as /dev/sdd and /dev/sdf depending on the Linux discovery method used by the HBA ports connecting the server to the SAN.

Among other uses, these /dev/sdX device names designate the volumes for mount commands including entries in the /etc/fstab file. In static disk environments, /dev/sdX device names work well for entries in the /etc/fstab file. However, the dynamic nature of Fibre Channel or iSCSI connectivity inhibits Linux from tracking these disk designations persistently across reboots.

There are multiple ways to ensure that these volumes are assigned and referenced by a persistent naming scheme. This section presents using volume labels or universally unique identifiers (UUIDs) with /dev/sdX referenced volumes. Volume labels are exceptionally useful when scripting SC Series snapshot recovery. An example involves mounting a snapshot view volume of a production SC Series snapshot to a backup server. In this case, the snapshot view volume may be referenced by its volume label without needing to explicitly identify the associated /dev/sdX device. The volume label is metadata stored within the volume and is inherited by snapshot view volumes cut from the SC Series snapshot.

Volume labels or UUIDs can also be used in multipath environments. That being said, multipath /dev/mapper/mpathX device names (or multipath aliases a described in section 3.6.1) are persistent by default and will not change across reboots. Volume labels, UUIDs, or multipath device names can be used interchangeably for entries in the /etc/fstab file of a local Linux host.

If snapshot view volumes are used for recovery or relocation of volumes to an alternate Linux host, the use of volume labels or UUIDs is recommended because these values are uniquely identifying of the volume, in contrast to how multipath names may differ (depending on the configuration of the **/**etc/multipath.conf file).

2.5.1 Filesystem volume labels

Filesystem volume labels can be applied when creating a filesystem on the volume or by subsequently using differing commands. Different filesystems (for example, ext4 or xfs) have different filesystem metadata schemas and use different commands to view, manage, and change this data.

Filesystem volume labels are created in the /dev/disk/by-label/ folder. Entries in this folder are created as symbolic links to their respective devices in the /dev folder. The /dev/disk/by-label/ folder is managed dynamically and will not exist if none of the volumes on the Linux host have any volume labels applied.

The following examples demonstrate some of these concepts. The sample script in section 2.5.4 parses all the Linux multipath device names and presents its multipath UUID, any known filesystem type, filesystem volume label, and filesystem UUID values to STDOUT.

- To apply a filesystem and label at the same time:

```
# mkfs.ext4 -L My_ext4_vol /dev/sdX
# mkfs.xfs -L My_xfs_vol /dev/mapper/mpathX
```

- To apply a volume label to an existing filesystem (note the different commands used for different filesystem types):

```
# tune2fs -L My_ext4_vol /dev/sdX
# xfs_admin -L My_xfs_vol /dev/mapper/mpathX
```

- To remove a volume label from an existing filesystem (note the different commands used for different filesystem types):

```
# tune2fs -L ‟” /dev/sdX
# xfs_admin -L ‟--‟ /dev/mapper/mpathX
```

2.5.2 Universally unique identifier

UUID values exist and persist at multiple layers of volume metadata. UUIDs are created and used by the device-mapper drivers to uniquely identify and manage each multipath device in Linux. UUIDs are created within LVM metadata (if the volumes are managed by LVM). UUIDs are also created and managed within the filesystem layer to uniquely identify each filesystem volume. UUIDs are created dynamically during multipath definition, LVM, or filesystem creation. Although UUIDs can be changed within each of these layers independently of other layers, it is not recommended unless its intention is clear and well defined.

Any UUID entries are created in the /dev/disk/by-uuid/ folder. Entries in this folder are created as symbolic links to their respective devices in the /dev folder.

A sample script in section 2.5.4 parses all Linux multipath device names and presents its multipath UUID, any known filesystem type, filesystem volume label, and filesystem UUID values to STDOUT.

2.5.3 Persistent naming in /etc/fstab and /boot/grub2/grub.cfg

The **LABEL=** or **UUID=** syntax can be used to reference volumes in a variety of places including mount commands, entries in the /etc/fstab file and /boot/grub2/grub.cfg files or swap partitions. This provides the liberty of uniquely identifying the volumes regardless of their discovery device name designations, multipath configurations or predefined multipath aliases.

The following sample **/etc/fstab** output demonstrates using some of these concepts.

```
# cat /etc/fstab
[snip]
/dev/mapper/VolGroup00-home /home                   xfs    defaults          1 2
UUID=8284393c-18aa-46ff-9dc4-0357a5ef742d    swap         swap   defaults 0 0
LABEL=TestVol   /vol_001                            xfs    defaults,discard  0 0
[snip]
```

The following sample **/etc/grub2.conf** output demonstrates using some of these concepts.

```
# cat /boot/grub2/grub.cfg
[snip]
linux16 /vmlinuz-3.10.0-123.el7.x86_64 root=UUID=35bea1c0-ce32-42a4-8e36-
72fd5e77471d …
[snip]
```

Note: It is no longer recommended to manually edit the **/boot/grub/grub.cfg** file, and instead to use the **grubby** command line interface to achieve boot-level configuration changes. The grubby tool manual pages and help are available with the following commands, respectively: **man grubby** or **grubby --help**.

2.5.4 Volume label and UUID script

This sample script parses all devices identified as **vendor="COMPELNT",** determines if any filesystem has been applied to the volume, then extracts and displays any discovered disk label values, mpath-layer UUID, and disk-layer UUID values.

```
#!/bin/bash

MP=/usr/sbin/multipath
BLKID=/usr/sbin/blkid

for i in `${MP} -ll | grep COMPELNT | cut -d" " -f1 | sort -k1`
do
        echo "INFO: /dev/mapper/${i}"
        FsTyp=`${BLKID} /dev/mapper/${i} | awk -F" " '{print $(NF)}' | cut -d= -
f2 | cut -d\" -f2`
        if [ "${FsTyp}" = "" ]; then
                echo "WARN: No filesystem detected"
        else
                echo "Multipath Info:"
                /usr/sbin/dmsetup info /dev/mapper/${i} | grep UUID
                echo "Filesystem Details:"
                echo "Type: ${FsTyp}"
                case ${FsTyp} in
                        xfs)
                                Cmd="/usr/sbin/xfs_admin -lu"
                                ${Cmd} /dev/mapper/${i}
                                ;;
                        ext4|ext3)
                                Cmd="/usr/sbin/tune2fs -l"
                                ${Cmd} /dev/mapper/${i} | egrep 'volume|UUID'
                                ;;
                        *)
                                echo "WARN: Filesystem unknown"
                                ;;
                esac
        fi
        echo
done
```

Sample output from this script as shown in the following:

```
# ./get_UUID.sh
[snip]
INFO: /dev/mapper/vol_001
Multipath Info:
UUID: mpath-36000d31000006500000000000000017f2
Filesystem Details:
Type: xfs
label = "TestVol"
UUID = fbef5bfb-94b9-4827-a103-516045b9b608

INFO: /dev/mapper/vol_002
Multipath Info:
UUID: mpath-36000d31000006500000000000000017f3
Filesystem Details:
Type: LVM2_member
WARN: Filesystem unknown
[snip]
```

2.6 Using snapshot view volumes

The use of the XFS filesystem allows for simple, effective integration with SC Series snapshot and snapshot view volumes. An applied XFS filesystem residing on top of a volume creates, maintains, and manages XFS metadata on the volume including a volume UUID. This volume UUID assists Linux in uniquely identifying this volume regardless of the attached host.

Note: These methods can be applied to XFS file system management on any Linux host. If the snapshot view volume is presented to an alternate Linux host (with a UUID), then the volume mounts normally without needing any of the following methods.

The volume label and UUID of a known XFS filesystem can be displayed with the following command.

```
# xfs_admin -ul /dev/mapper/vol_001
UUID = fbef5bfb-94b9-4827-a103-516045b9b608
label = "TestVol"
```

This UUID on the filesystem is inherited by any snapshot or snapshot view volumes created from the original SC Series volume. As such, any snapshot view volumes cannot be automatically presented to and mounted on the originating host (even though the multipath device has its own unique ID). In the following example, a snapshot view volume is created from the XFS filesystem-based /dev/mapper/vol_001 device and mounted as /vol_001. This snapshot view volume is discovered, identified, and aliased as /dev/mapper/vol_004 using the /etc/multipath.conf file.

Note: The attempt to mount **/dev/mapper/vol_004** as **/vol_004** fails because the UUID is not unique and already in use on the Linux host.

```
# df -k
Filesystem                  1K-blocks     Used Available Use% Mounted on
[snip]
rhevm:/nfsroot              103213056 54833152  43137024  56% /Tools
/dev/mapper/vol_001          52418560  1939052  50479508   4% /vol_001
[snip]
```

Attempting to mount **/dev/mapper/vol_004** to **/vol_004** returns the following error.

```
# mount –o discard,sync /dev/mapper/vol_004 /vol_004
mount: wrong fs type, bad option, bad superblock on /dev/mapper/vol_004,
       missing codepage or helper program, or other error

       In some cases useful info is found in syslog - try
       dmesg | tail or so.
```

Attempting to **mount /dev/mapper/vol_004** to **/vol_004** along with the **nouuid** parameter allows the volume to be successfully mounted to /vol_004.

```
# mount –o discard,sync,nouuid /dev/mapper/vol_004 /vol_004; df –k

# df -k
Filesystem                  1K-blocks     Used Available Use% Mounted on
[snip]
rhevm:/nfsroot              103213056 54833152  43137024  56% /Tools
/dev/mapper/vol_001          52418560  1939052  50479508   4% /vol_001
/dev/mapper/vol_004          52418560  1939052  50479508   4% /vol_004
[snip]
```

Alternatively, the UUID value associated with **/dev/mapper/vol_004** can be changed permanently to remove the need for specifying the **nouuid** parameter with the mount command. This can be performed (applied to an unmounted filesystem) as shown in the following.

```
# xfs_admin –U generate /dev/mapper/vol_004
Clearing log and setting UUID
writing all SBs
new UUID = 26b4a32d-4a3a-405f-899c-7bb63087cc7b
```

2.7 Expanding XFS filesystem enabled volumes

With RHEL 7.x, volume and filesystem expansion in Linux is simple and effective. After the underlying SC Series volume has been increased in size, the filesystem residing on top of the volume can be resized to the full limit of the underlying volume or resized to a predetermined size as specified by the system administrator. The process for volume and XFS filesystem expansion is outlined in the following steps.

Note: The following volume expansion steps are only for those volumes which have an XFS filesystem applied. This is not applicable to volumes which have already been partitioned or LVM enabled.

1. Expand the SC Series volume.

   a. Right-click the SC Series volume.
   b. Select **Expand Volume**.
   c. Enter a new size for the volume.
   d. Click **OK** to execute the expansion.

2. Rescan the drive geometry for each path of the multipath device by executing:

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

3. Resize the multipath map by executing

   ```
   # multipathd -k"resize map <devicename>"
   ```

   Note the following:

   - The multipath device is **/dev/mapper/<devicename>.**
   - There is no space between the **–k** parameter and the command string.

4. Expand the XFS filesystem (in this example, to the boundary limit size of the underlying volume).

   ```
   # xfs_growfs -d /<mountpoint_name>
   ```

Note: Reducing the size of a volume (or its accompanying filesystem) on demand is not recommended in any environment. Alternate methods to achieving this result that greatly reduce the risk and potential impact toward data integrity include backup and recovery, relocating the contents of a volume to an alternate new, size-reduced SC Series volume, or mirroring and splitting volumes.

The rescan-scsi-bus.sh script (included in the sg3_utils.x86_64 package) can be used to rescan the SCSI bus to update the host's seen devices (after the device has been presented to the host).

The following shows the sample, truncated output from the **rescan-scsi-bus.sh** script showing a rescan of the SCSI bus against two host adapter's (adapter 10 and adapter 8) with no new device found or devices removed:

```
# rescan-scsi-bus.sh
Host adapter 10 (iscsi_tcp) found.
Host adapter 8 (iscsi_tcp) found.
Scanning SCSI subsystem for new devices
Scanning host 8 for  all SCSI target IDs, all LUNs
Scanning for device 8 0 0 1 ...
OLD: Host: scsi8 Channel: 00 Id: 00 Lun: 01
      Vendor: COMPELNT Model: Compellent Vol   Rev: 0702
      Type:   Direct-Access                    ANSI SCSI revision: 05
Scanning host 10 for  all SCSI target IDs, all LUNs
Scanning for device 10 0 0 1 ...
OLD: Host: scsi10 Channel: 00 Id: 00 Lun: 01
      Vendor: COMPELNT Model: Compellent Vol   Rev: 0702
      Type:   Direct-Access                    ANSI SCSI revision: 05
0 new device(s) found.
0 device(s) removed.
```

2.8 Removing volumes

Linux stores information about each volume presented to it. Even if a volume is in an unmapped state on the SC Series storage, Linux will retain information about that volume until the next reboot. If Linux is presented with a volume from the same target using the same LUN ID prior to any reboot, it will reuse the old data about that volume. This may result in misinformation and mismanagement of the volumes and potentially impact data integrity in the business environment.

It is recommended to always unmount, remove, and delete all volume information on Linux after the volume is deemed no longer in use. This management of the Linux volume metadata is non-destructive to any actual data stored on the volume itself.

The process for removing multipath volumes is outlined in the following steps.

1. Quiesce any I/O to the mounted volume.
2. Unmount the volume.
3. Edit **/etc/multipath.conf** and remove any syntax referencing this volume.
4. Reload the multipathd daemon (**systemctl restart multipathd.service**).
5. Remove the multipath backing device files by running the following script.
6. Remove any mappings to the volume(s) from SC Series storage.

The **rescan-scsi-bus.sh** script (included in the **sg3_utils.x86_64** package) with the **–r** flag can be used to rescan the SCSI bus to remove devices from the host.

To remove single volumes (not managed by multipath) use the following steps.

1. Quiesce any I/O to the mounted volume.
2. Unmount the volume.
3. From the command prompt, execute:

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

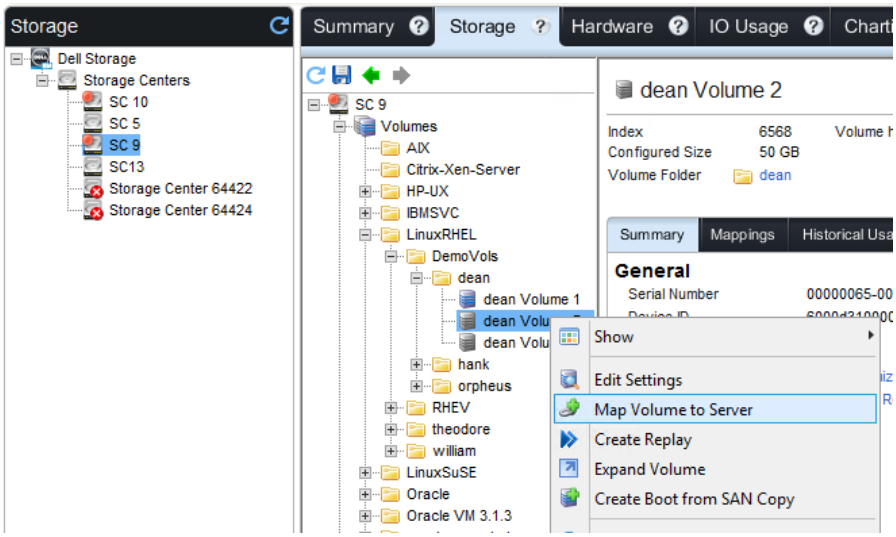
4. Remove any mappings to the volume(s) from SC Series storage.

2.9 Boot from SAN

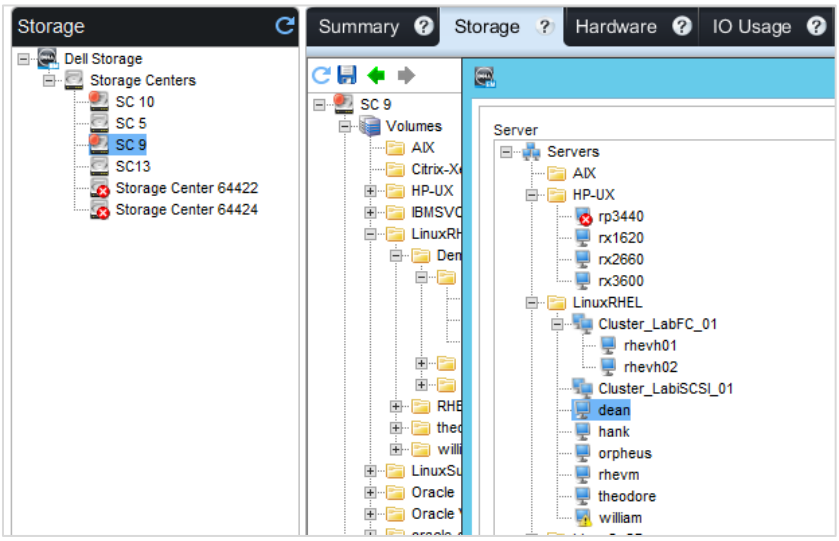
The ability to use SC Series volumes as bootable volumes in Linux allows system and storage administrators to further leverage the strengths of SC Series snapshot and snapshot view volume technologies. Two uses for snapshots of Linux boot volumes are: a backup/recovery mechanism, or to preserve the state of an operating system at a point in time prior to upgrades.

To use an SC Series volume as a bootable volume, the target volume needs to be presented to the target Linux host as **LUN ID 0**. A volume LUN ID 0 mapping with SC Series storage is performed as follows.

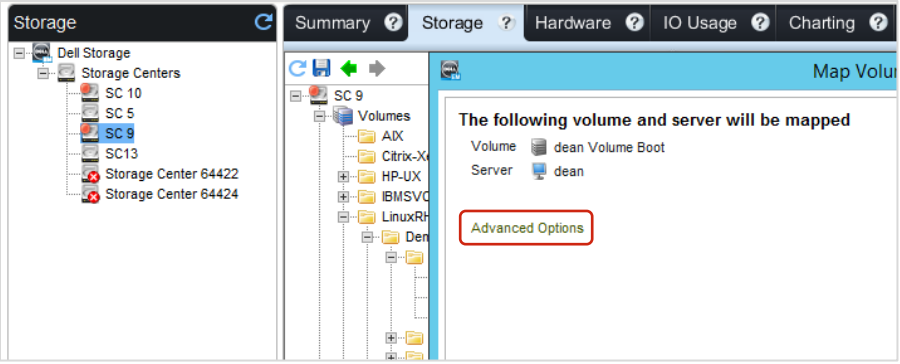
1. Right-click the volume and select **Map Volume to Server.**



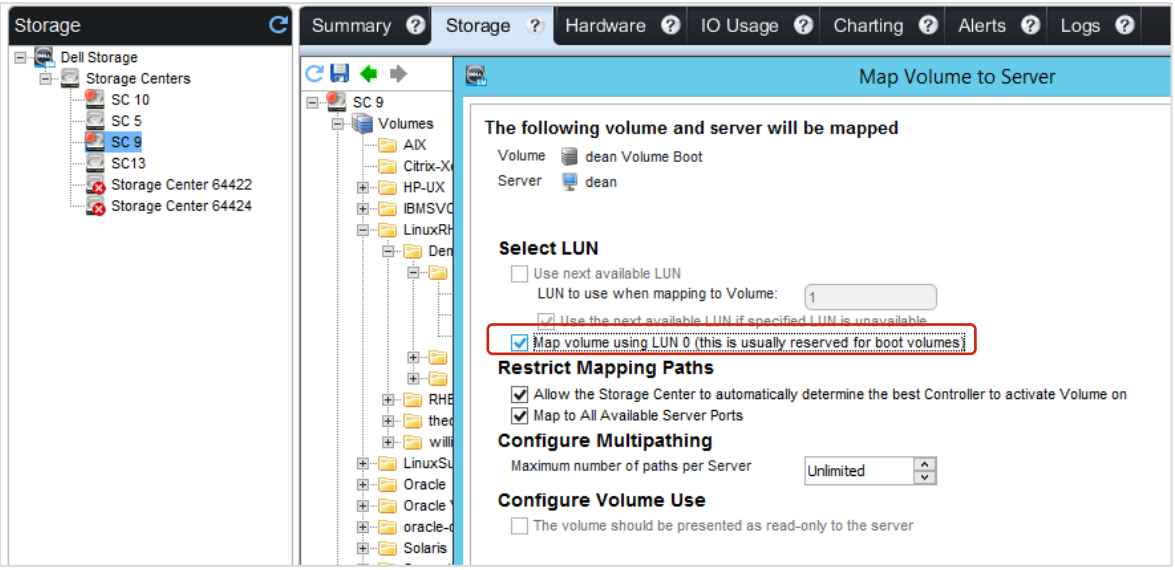
2. Identify the host where volume needs to be mapped and click **Next**.



3. Click **Advanced Options**.



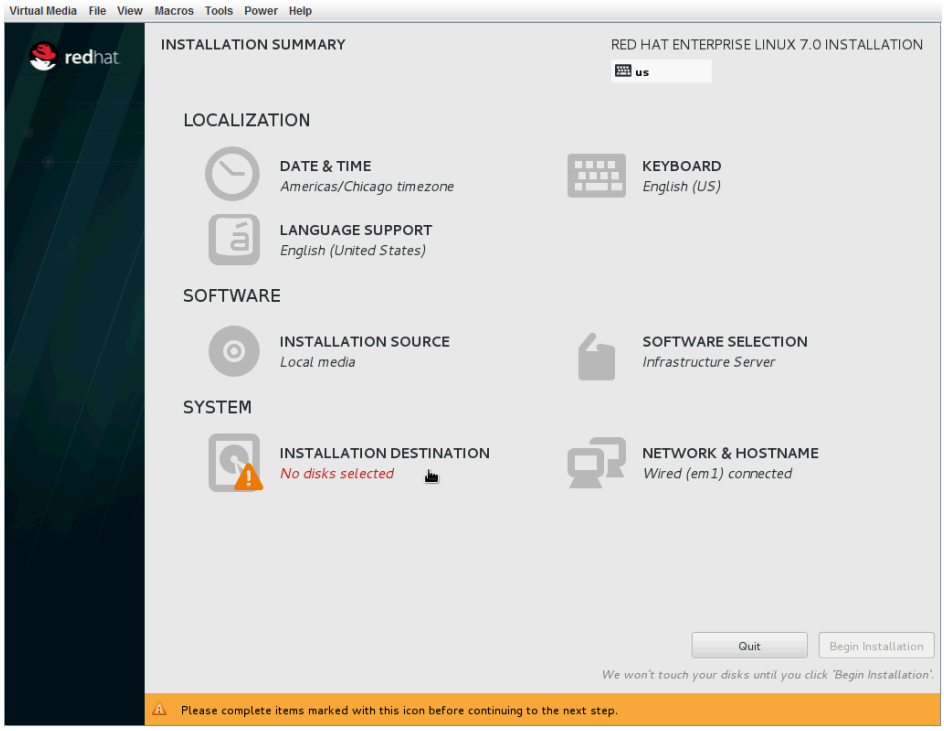
4. Select the **Map volume using LUN 0** check box and click **Finish** to map this volume.



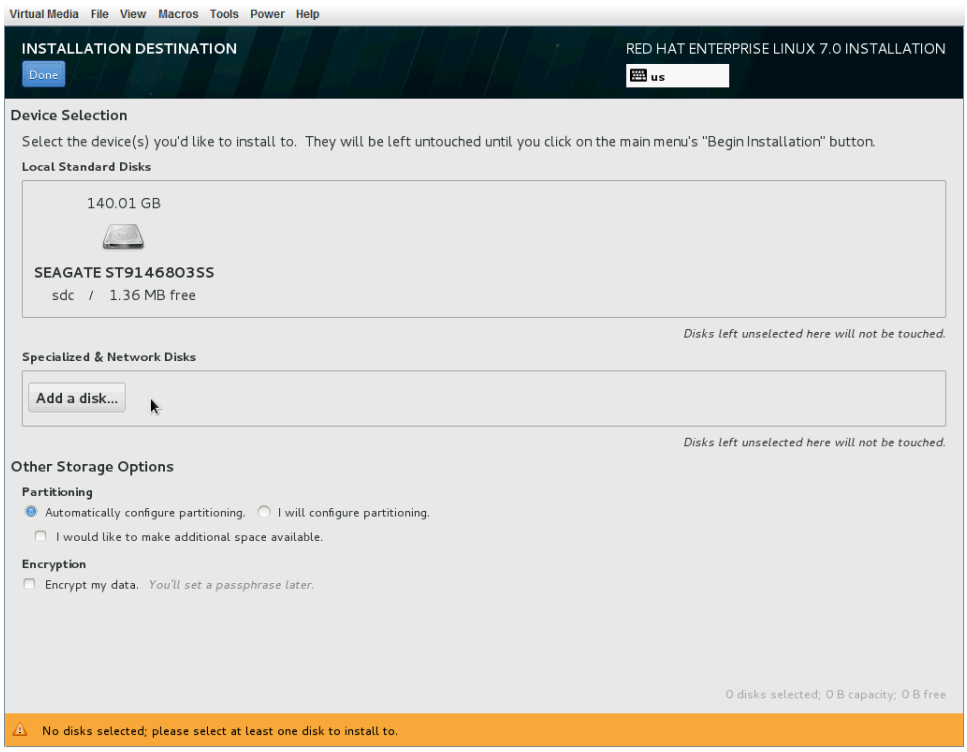
During the Linux host boot, the HBA BIOS boot process is interrupted and instructed to identify this SC Series volume as the preferred boot device in the boot device order.

During the Linux installation process, this SC Series volume is identified and selected as the installation target shown as follows.

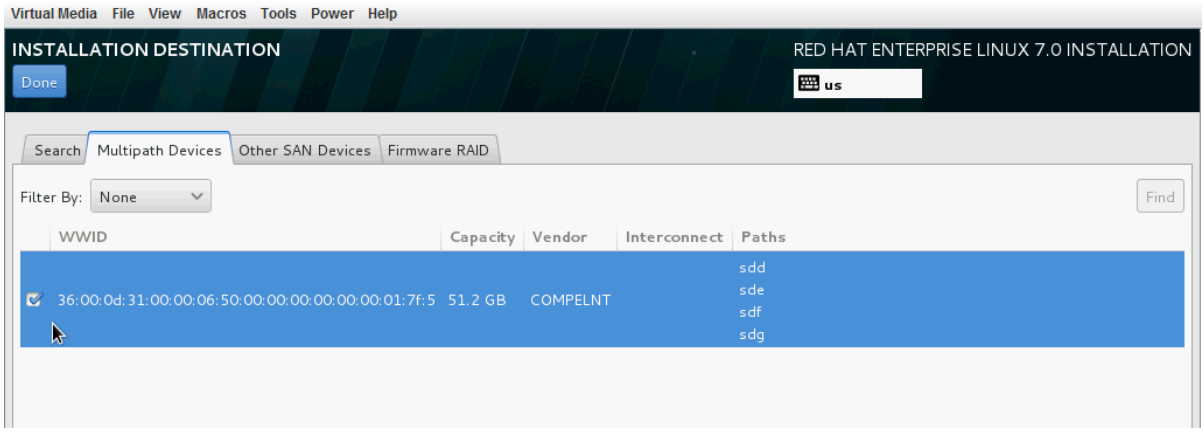
1. In the Installation main menu, click **Installation Destination**.



2. Click **Add a disk…**.

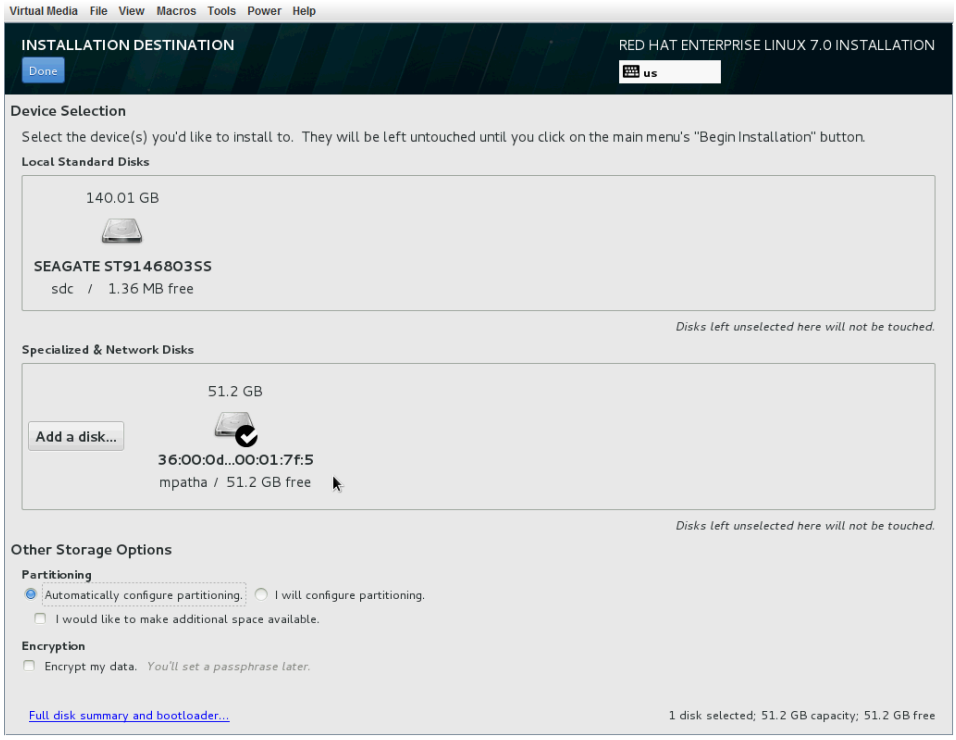


3. Identify the SC Series volume in the **Multipath Devices** tab, and select the check box accordingly.



4. Click **Done**.

   The SC Series volume shown is identified and selected as the single installation target for the Linux operating system.



5. Click **Done** to return to the installation main menu.

2.9.1 Capturing a boot from SAN snapshot volume

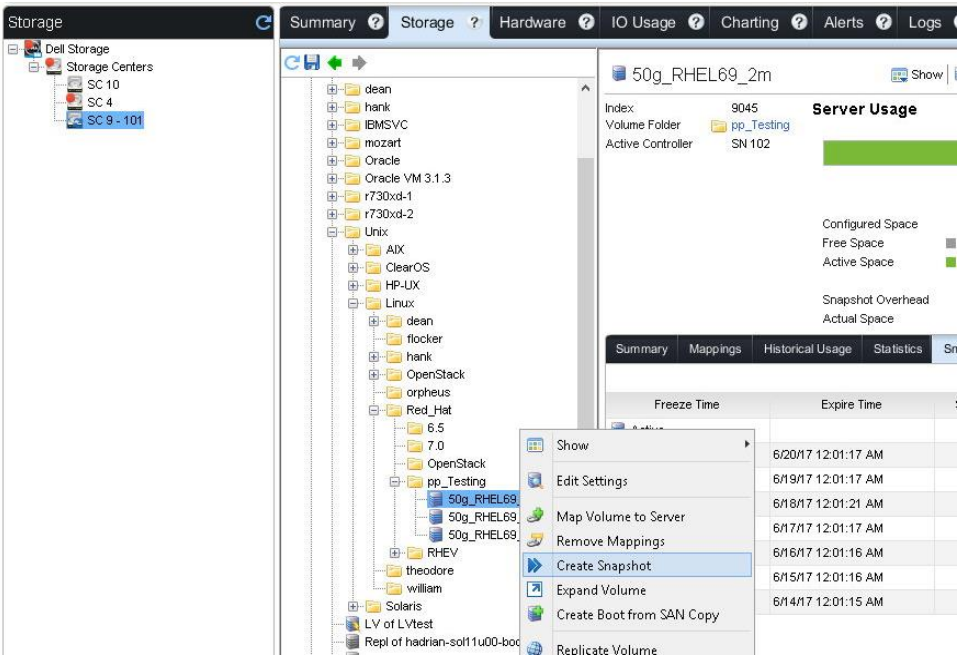
Capturing a snapshot volume of a boot from SAN volume is straightforward in SC Series storage. However, it is recommended to freeze any non-LVM-bound XFS volumes on the Linux host in order to quiesce any I/O before the snapshot volume is captured.

In this scenario (with a default Linux installation), the **/** and **/boot** filesystems are XFS-based, and frozen with the following commands.

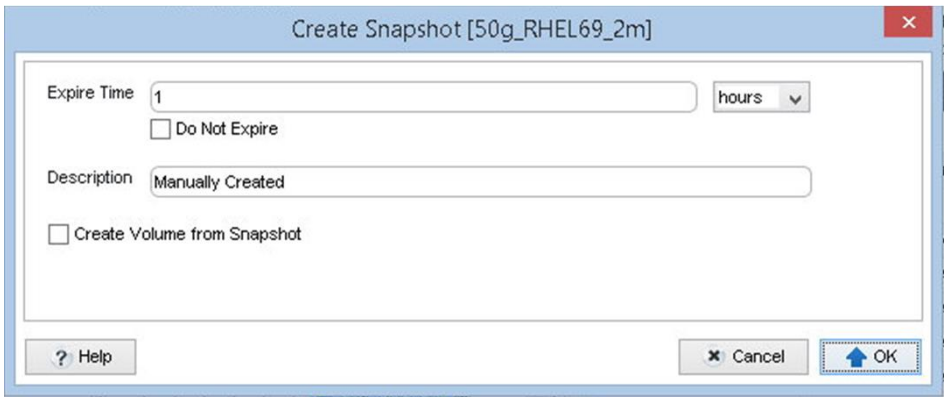
```
# xfs_freeze -f /
# xfs_freeze -f /boot
```

Capturing a snapshot volume of a boot from SAN volume with SC Series storage is performed as follows.

1. Right-click the volume and select **Create Snapshot**.



2. Give the volume a **1** week expiration time and click **OK** to create the snapshot.

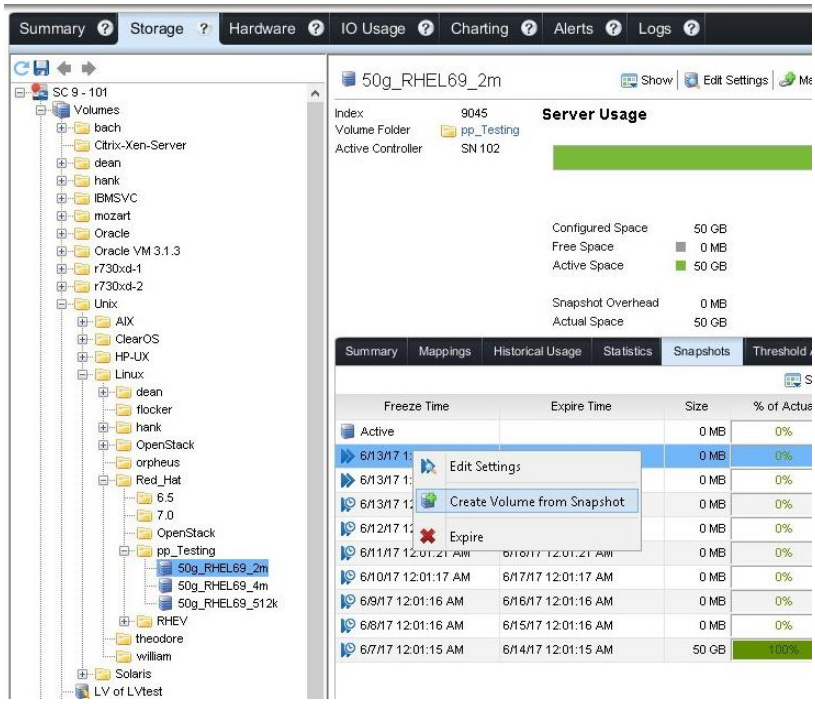


3. After the snapshot has been captured, the **/** and **/boot** filesystems are XFS-based and should be unfrozen with the following commands.

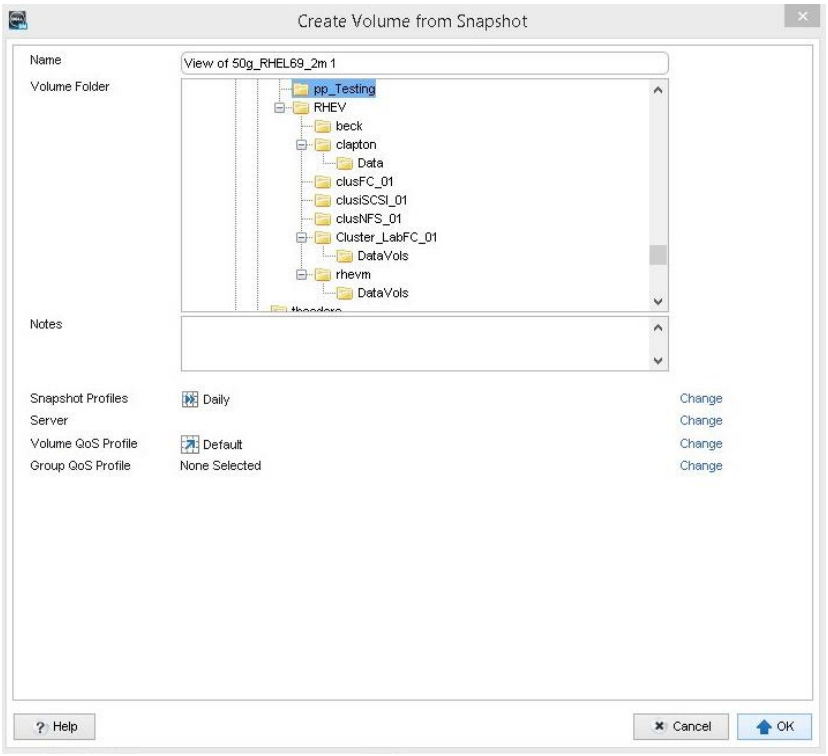
```
# xfs_freeze -u /
# xfs_freeze -u /boot
```

   A snapshot view volume from this captured snapshot is created as shown in step 4.

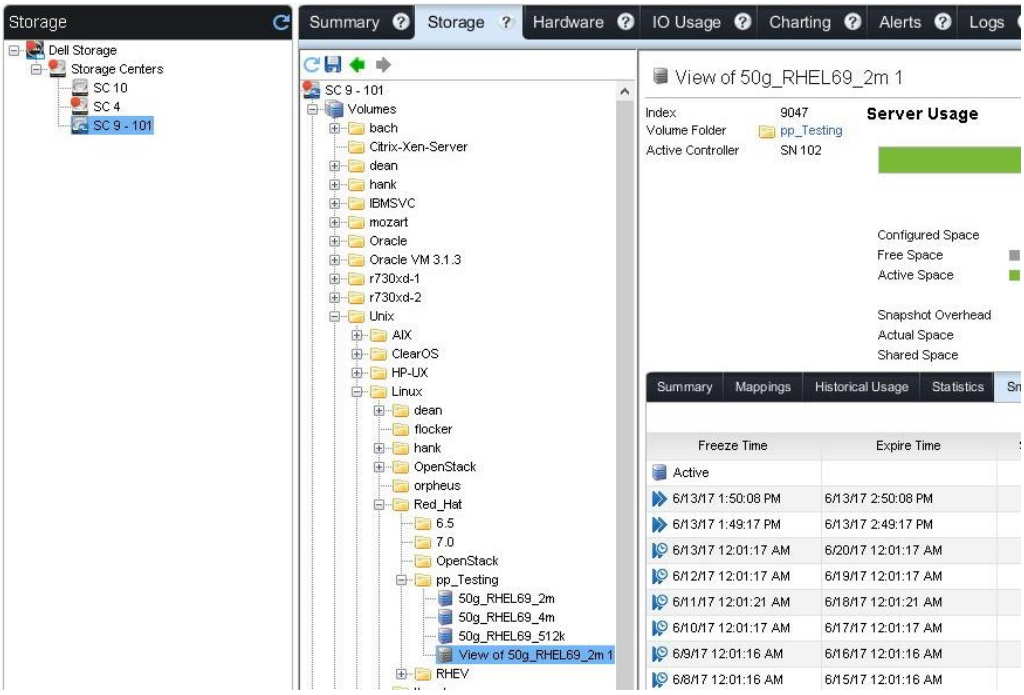
4. Right-click the desired snapshot in the volume **Snapshots** tab, and click **Create Volume from Snapshot**.



5. Associate this snapshot view volume to the appropriate server host object and click **OK**.



The snapshot view volume is shown in the **Volumes** tree, beneath the volume where it was created.



2.9.2 Recovering from a boot from SAN snapshot view volume

Snapshot view volumes of a boot from SAN volume, when presented back to the same host, are automatically bootable. This is because the volumes are now tracked using their unique and identifying UUID values inside of **/etc/fstab** and **/boot/grub2/grub.cfg** instead of using their system device file names. Additionally, the entries and aliases within **/etc/multipath/bindings** and **/etc/multipath/wwids** are also automatically updated to reflect any boot device changes.

Note: Ensure that the original boot from SAN volume is no longer mapped to the server object to prevent any boot time conflict when attempting to boot from the alternately mapped snapshot view volume. The snapshot view volume should also be mapped to the host as **LUN ID 0**.

2.9.3 Configuring the HBA BIOS

The HBA BIOS configuration on each Linux host should be configured to identify all SC Series storage controller World Wide Port Names (WWPNs) of all the active and alternate controllers. The configuration of the HBA BIOS in this manner ensures that any boot from SAN volumes will remain visible and accessible on all fabric-zoned controller paths in the event of any SC Series storage controller failover or other path failure events.

3 Server configuration

This section discusses the various configuration aspects of the I/O stack of a Linux host. The Linux I/O stack can be precisely configured to adapt to the needs of the environment in which the Linux host operates. The configuration aspects include tuning the HBA port retry count, queue depth, SCSI and iSCSI device timeout values, and more.

3.1 Fibre Channel and modprobe

The Linux modprobe facility provides a means to manage and configure the operating parameters of installed FC HBAs (such as QLogic and Emulex). The modprobe facility is configured by editing or creating text files within the /etc/modprobe.d directory.

3.1.1 Configuration

QLogic and Emulex HBAs are managed by configuration syntax that is written within individual text files, located in the **/etc/modprobe.d** directory. Create these files if they do not already exist. The configuration syntax for QLogic and Emulex hardware is discussed in detail in sections 0 and 3.8, respectively. If these files do not exist in a fresh RHEL 7.x installation, they can be created manually using any text editor.

- For QLogic:

  ```
  /etc/modprobe.d/qla2xxx.conf
  ```

- For Emulex:

  ```
  /etc/modprobe.d/lpfc.conf
  ```

3.1.2 Reloading modprobe and mkinitrd

After configuration changes are made, reload the modprobe facility for the new or updated configurations to take effect.

For local boot systems, it is recommended to unmount all SAN volumes and reload the module. The module should be unloaded from memory before it is reloaded as follows.

```
# modprobe –r qla2xxx
# modprobe qla2xxx
```

Replace **qla2xxx** with **lpfc** if working with Emulex hardware; SAN volumes can be remounted subsequently.

For configuration persistence, the initramfs-*.img file needs to be rebuilt so that the new configurations are incorporated during boot time. The following methods demonstrate rebuilding the initramfs-*.img file. This process overwrites the same file in its existing location. It is recommended to back up the existing **initramfs-*.img** file before applying this procedure. The two commands can be used as follows

```
# cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).img.$(date
+%Y%m%d-%H%m)
# dracut --force
```

Ensure that the GRUB entry in **/boot/grub2/grub.cfg** points to the correct **initramfs-*.img** file and reboot the system.

Note: It is no longer recommended to manually edit the **/boot/grub/grub.cfg** file. Use the **grubby** command line interface to achieve boot-level configuration changes. The grubby tool manual pages and help are available with the following commands, respectively: **man grubby** or **grubby --help**.

3.2 iSCSI

RHEL 7.x introduces an updated software implementation of the open-iSCSI stack (based on RFC3720). This new iSCSI stack resides in the kernel memory space instead of user space.

iSCSI is mature technology that allows organizations to scale into the realm of enterprise storage while leveraging their existing infrastructure. This section discusses the configuration, use, and implementation of the open-iSCSI stack only. For more advanced implementations of iSCSI software (which may include custom iSCSI HBA drivers and offload engines), consult with the associated vendor documentation.

3.2.1 Configuration

The Linux host being configured requires an Ethernet port that can communicate with the iSCSI ports on the SC Series storage. It is recommend that a dedicated port/VLAN is used for this purpose.

One of the more important considerations when configuring iSCSI is the network path. Due consideration is recommended to determine the confidentiality, security, and latency required for the iSCSI traffic. These needs will define and determine the network topology of the iSCSI architecture (such as dedicated physical ports or VLAN, multipath, and redundancy).

In an ideal scenario, iSCSI traffic is separated and isolated from routine network traffic by the use of dedicated ports, switches, and infrastructure. If the physical topology is constrained, it is recommended to separate and isolate iSCSI traffic by the use of VLAN subnets. It is also recommended to always use iSCSI in a multipath configuration to create path redundancy.

If VLAN subnets are not possible, two further options should be explored:

- Route traffic at the network layer by defining static routes.
- Route traffic at the iSCSI level through configuration.

The following demonstration walks through the process of requesting targets from the SC Series storage discovery IP from the Linux host, logging in to these targets, and setting up an iSCSI-based server object on the SC Series storage. Additional methods, like scanning for newly discovered volumes, are discussed in section 2.1.

In this demonstration, only the em2 interface is configured to the iSCSI VLAN as shown in the following ifconfig and netstat output. In production use, it is recommended to configure more than one iSCSI interface for each Linux host, using dm-multipath for path, redundancy, and I/O queue management.

1. Configure the network interface.

```
# ifconfig -a
[snip]
em2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.83.21  netmask 255.255.0.0  broadcast 10.10.255.255
        ether 14:fe:b5:c9:95:7f  txqueuelen 1000  (Ethernet)
        RX packets 12670048  bytes 1031419587 (983.6 MiB)
        RX errors 0  dropped 34235  overruns 0  frame 0
        TX packets 59  bytes 9454 (9.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
[snip]

# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt
Iface
0.0.0.0         172.16.16.1     0.0.0.0         UG        0 0          0
em1
10.10.0.0       0.0.0.0         255.255.0.0     U         0 0          0
em2
172.16.16.0     0.0.0.0         255.255.240.0   U         0 0          0
em1
224.0.0.0       -               255.255.255.0   !         - -          - -
255.255.255.255 -               255.255.255.255 !H        - -          - -
```

2. With the em2 interface configured, and knowing that the SC Series discovery IP addresses are **10.10.97.2** and **10.10.130.2**, accordingly (as identified from the SC Series management GUI), validate that these IP addresses can be pinged.

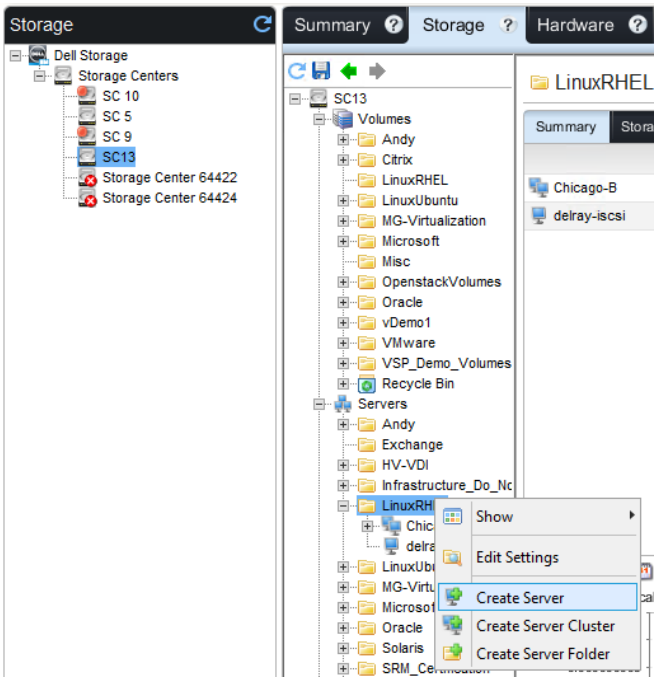
```
# ping 10.10.97.2
PING 10.10.97.2 (10.10.97.2) 56(84) bytes of data.
64 bytes from 10.10.97.2: icmp_seq=1 ttl=64 time=0.289 ms
64 bytes from 10.10.97.2: icmp_seq=2 ttl=64 time=0.353 ms
[snip]

# ping 10.10.130.2
PING 10.10.130.2 (10.10.130.2) 56(84) bytes of data.
64 bytes from 10.10.130.2: icmp_seq=1 ttl=64 time=0.394 ms
64 bytes from 10.10.130.2: icmp_seq=2 ttl=64 time=0.318 ms
[snip]
```

3. Issue the following commands to request SC Series target IQNs and request the Linux host to login to these targets.

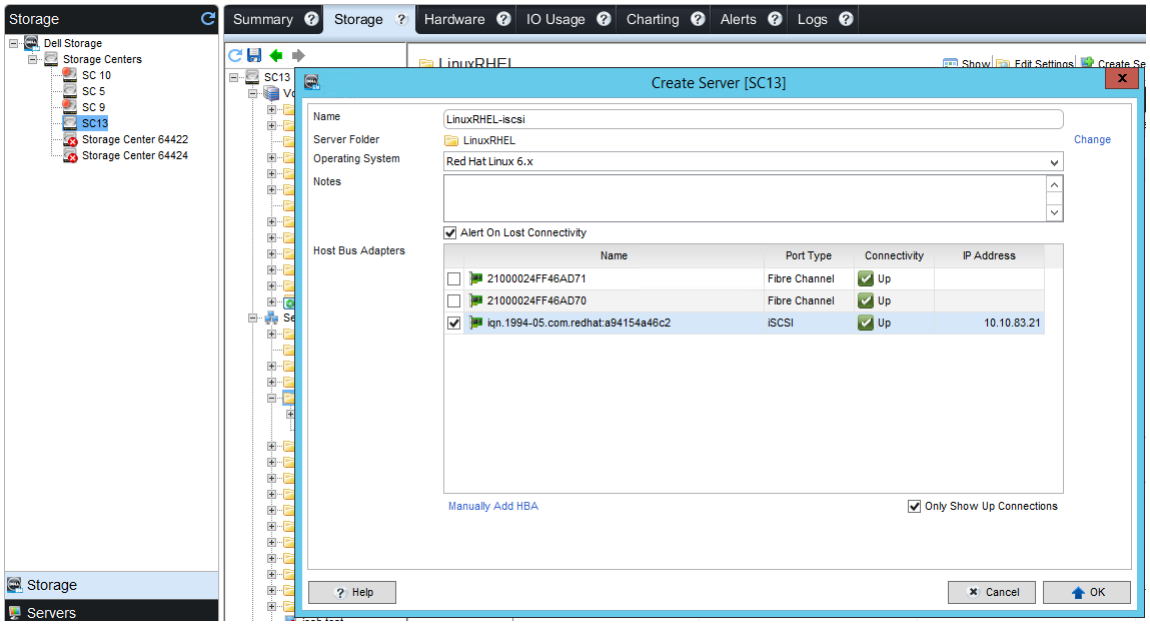
```
# iscsiadm -m discovery -t sendtargets -p 10.10.97.2:3260
10.10.97.1:3260,0 iqn.2002-03.com.compellent:5000d3100002bb2f
10.10.97.1:3260,0 iqn.2002-03.com.compellent:5000d3100002bb30
10.10.97.1:3260,0 iqn.2002-03.com.compellent:5000d3100002bb32
10.10.97.1:3260,0 iqn.2002-03.com.compellent:5000d3100002bb3b

# iscsiadm -m discovery -t sendtargets -p 10.10.130.2:3260
10.10.130.1:3260,0 iqn.2002-03.com.compellent:5000d3100002bb31
10.10.130.1:3260,0 iqn.2002-03.com.compellent:5000d3100002bb39
10.10.130.1:3260,0 iqn.2002-03.com.compellent:5000d3100002bb3a
10.10.130.1:3260,0 iqn.2002-03.com.compellent:5000d3100002bb3c

# iscsiadm -m node --login
Logging in to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb2f, portal: 10.10.97.1,3260] (multiple)
Logging in to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb30, portal: 10.10.97.1,3260] (multiple)
Logging in to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb32, portal: 10.10.97.1,3260] (multiple)
Logging in to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb3b, portal: 10.10.97.1,3260] (multiple)
Logging in to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb31, portal: 10.10.130.1,3260] (multiple)
Logging in to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb39, portal: 10.10.130.1,3260] (multiple)
Logging in to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb3a, portal: 10.10.130.1,3260] (multiple)
Logging in to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb3c, portal: 10.10.130.1,3260] (multiple)
Login to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb2f, portal: 10.10.97.1,3260] successful.
Login to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb30, portal: 10.10.97.1,3260] successful.
Login to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb32, portal: 10.10.97.1,3260] successful.
Login to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb3b, portal: 10.10.97.1,3260] successful.
Login to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb31, portal: 10.10.130.1,3260] successful.
Login to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb39, portal: 10.10.130.1,3260] successful.
Login to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb3a, portal: 10.10.130.1,3260] successful.
Login to [iface: default, target: iqn.2002-
03.com.compellent:5000d3100002bb3c, portal: 10.10.130.1,3260] successful.
```

4. On the SC Series storage, set up the new iSCSI server object.

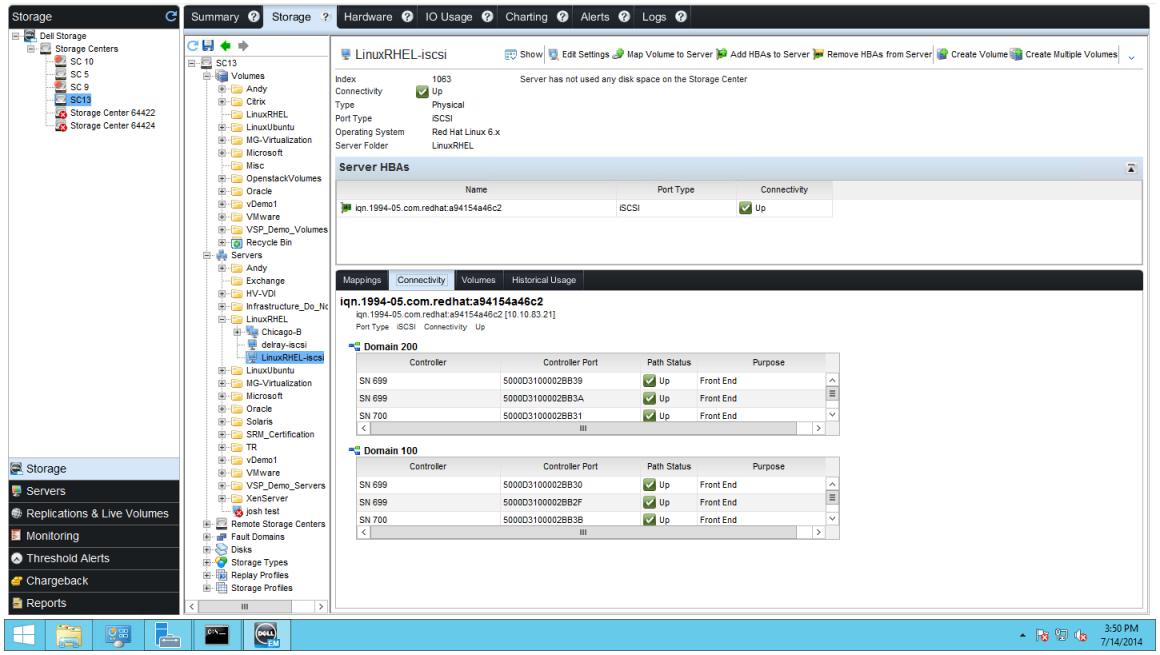
   a. Right-click the appropriate **Servers** tree folder and click **Create Server**.



   b. Name the server object, set the **Operating System** to **Red Hat Linux 6.x**, select the IQN checkbox that corresponds to the Linux host IP address (in this case, 10.10.83.21), and click **OK**.



The new **LinuxRHEL-iscsi** server object is shown in the **LinuxRHEL** folder and is ready to have volumes mapped to it.



3.2.2 iSCSI timeout

In single-path environments, configure the iSCSI daemon to wait and queue I/O for an amount of time sufficient to accommodate proper failure recovery. For example, a SAN fabric failure or SC Series storage array failover event can take between 5 and 60 seconds to complete. It is recommended to configure the iSCSI software initiator to queue I/O for up to 60 seconds before starting to fail I/O requests.

Note: The following configuration settings affect iSCSI connection timeouts.

To adjust this timeout value for a single-path environment, open **/etc/iscsi/iscsid.conf** and configure the following parameter to 60 seconds instead of the default value of 10 seconds.

```
node.conn[0].timeo.noop_out_timeout = 60
```

Note: The following configuration settings affect iSCSI connection timeouts.

When used in a multipath environments, configure the iSCSI daemon to fail a path in 5 seconds. This minimizes the latency of waiting for a single path to recover. Since I/O is managed by dm-multipath, it will automatically resubmit any I/O requests to alternating active iSCSI paths. If all paths are down, dm-multipath will queue the I/O until a path becomes available. This approach enables an environment to sustain failures at both the network and storage layers.

To adjust the following timeout values, for a multipath enabled environment, open **/etc/iscsi/iscsid.conf** and configure the following parameters:

- To control how often a NOP-Out request is sent to each target, configure the following parameter to 5 seconds instead of the default value of 10 seconds.

  ```
  node.conn[0].timeo.noop_out_interval = 5
  ```

- To control the timeout value for the NOP-Out request, configure the following parameter to 5 seconds instead of the default value of 15 seconds.

  ```
  node.conn[0].timeo.noop_out_timeout = 5
  ```

- To control the timeout value for how long the iSCSI layer should wait before failing any commands to queried paths, configure the following parameter to 5 seconds versus the default value of 15 seconds.

  ```
  node.session.timeo.replacement_timeout = 5
  ```

The replacement_timeout command controls the wait time of a session re-establishment before failing the pending SCSI commands. This includes commands that the SCSI layer error handler is passing up to a higher level (such as multipath) or to an application if multipath is not active.

The NOP-Out section dictates that if a network problem is detected, the running commands are failed immediately. The exception is if the SCSI layer error handler is running. To check if the SCSI error handler is running, run the following command.

```
# iscsiadm –m session –P 3
Host Number: X State: Recovery
```

When the SCSI error handler is running, commands will not be failed until the seconds of the node.session.timeo.replacement_timeout parameter are modified.

In multipath iSCSI environments where paths are managed by dm-multipath, configure iSCSI to monitor for problems on the SAN (by sending out a NOP-Out request to each target) every 5 seconds. In addition, set these requests to time out every 5 seconds. This manages and minimizes any latency in waiting for any single path to recover by automatically resubmitting I/O requests to an alternate and still active iSCSI path.

The iSCSI configuration parameters NOP-Out and its associated time out value are managed by these two entries in the **/etc/iscsi/iscsid.conf** file. Both entries default to 5 in RHEL 7.x.

```
node.conn[0].timeo.noop_out_interval = 5
node.conn[0].timeo.noop_out_timeout = 5
```

In multipath iSCSI use cases, this additional parameter in the /etc/iscsi/iscsd.conf file should be configured from its default value of 120 to 5. This will dictate that the iSCSI layer should wait up to 5 seconds before failing any commands to this path, thereby minimizing any latency and I/O wait.

```
node.conn[0].timeo.replacement_timeout = 5
```

3.2.3 Configuring /etc/fstab

iSCSI is dependent on an operational network, and volumes added to /etc/fstab need to be designated as network-dependent. In other words, do not attempt to mount an iSCSI volume until the network layer services have completed the startup and the network is operational. The example below demonstrates how to create this network dependency to the iSCSI mount using the **_netdev** mount option in the **/etc/fstab** file.

```
# cat /etc/fstab
[snip]
/dev/mapper/VolGroup00-home /home                   xfs   defaults          1 2
/dev/mapper/VolGroup00-swap swap                    swap  defaults          0 0
LABEL=iSCSI_Vol      /mnt/iSCSI_Vol                 ext4  defaults,_netdev  0 0
[snip]
```

3.3 Serial-attached SCSI

The Dell SCv2000 Series product line offers serial-attached SCSI (SAS) front-end connectivity and coincides with the launch of SCOS 6.6.x. The Dell SCv2000 Series supports the use of Dell EMC 12Gbps SAS HBAs on the target hosts.

SAS connectivity to a Linux host requires specific configuration schema in the /etc/multipath.conf file.

3.3.1 SAS drivers

SAS drivers are preloaded into certain Linux kernels (RHEL 6.5 or newer, and RHEL 7.x). The existence of the SAS drivers can be validated with the following commands. As a best practice, validate the driver versions with the Dell EMC Storage Compatibility Matrix and use the latest supported driver as indicated.

```
# lsmod | grep sas
mpt3sas               188001  4
scsi_transport_sas     35588  1 mpt3sas
raid_class              4388  1 mpt3sas
megaraid_sas           96205  2

# modinfo mpt3sas | grep description
description:    LSI MPT Fusion SAS 3.0 Device Driver
```

3.3.2 SAS /etc/multipath.conf

Add the following device schema to the **/etc/multipath.conf** file that is used exclusively for SAS-connected Linux hosts. This schema defines the configuration parameters for all devices identified by **vendor="COMPELNT"** and **product="Compellent Vol".** This schema is typically added after the defaults schema and before the **blacklist_exceptions** schema.

```
devices {

      device {
          vendor COMPELNT
          product "Compellent Vol"
          path_checker tur
          prio alua
          path_selector "service-time 0"
          path_grouping_policy group_by_prio
          no_path_retry 24
          hardware_handler "1 alua"
          failback immediate
          rr_weight priorities
      }

}
```

3.3.3 FC/iSCSI and SAS

The use of a merged FC/iSCSI and SAS-connected environment on the same Linux host is not validated nor supported. Maintain and operate a SAS-connected Linux host separately from other Linux hosts using FC/iSCSI connectivity.

The SAS-specific **multipath.conf** configuration schema can be merged with a FC/iSCSI-based **/etc/multipath.conf** file.

3.3.4 Identify SAS devices on Linux

This sample script parses the Linux /sys filesystem and displays the contents of the files located and identified as **host_sas_address**. The contents of this file represents the device name of the installed SAS HBA.

In the following instance, two SAS HBA cards are identified with their device names shown accordingly. This script works for both RHEL 6.x and RHEL 7.x Linux hosts.

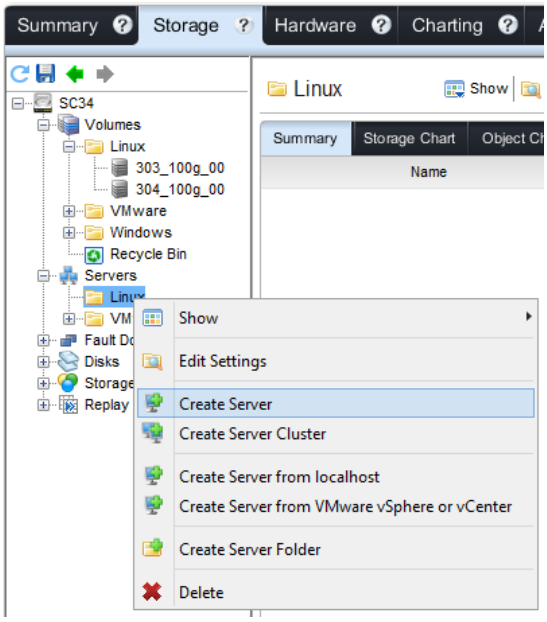
```
# for i in `find /sys/devices -name host_sas_address`; do echo "=== $i"; cat $i;
echo; done
===
/sys/devices/pci0000:40/0000:40:01.0/0000:41:00.0/host1/scsi_host/host1/host_sas
_address
0x544a842007902800

===
/sys/devices/pci0000:40/0000:40:03.0/0000:42:00.0/host2/scsi_host/host2/host_sas
_address
0x544a84200792ce00
```

3.3.5 Identify SAS devices on Dell SCv2000

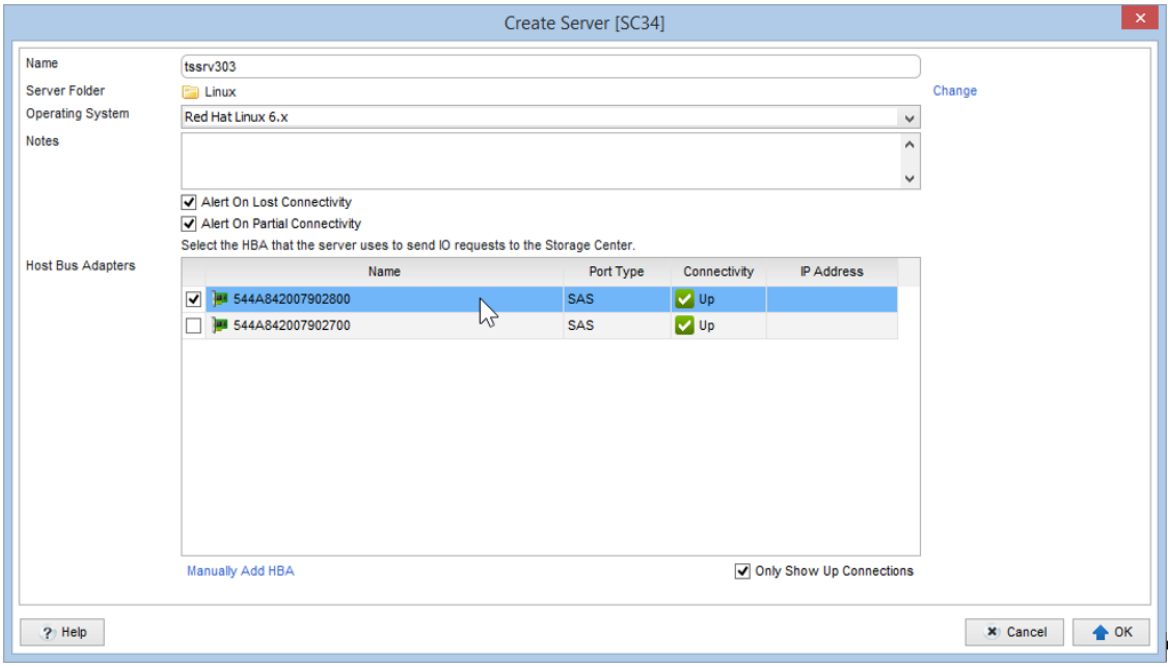
SAS devices are exposed to the Dell SCv2000 through their singular device names instead of their multiple SAS World-Wide Port Names (WWPNs).

The following sequence of screenshots outlines the procedure to create a new server object on a Dell SCv2000 storage array.

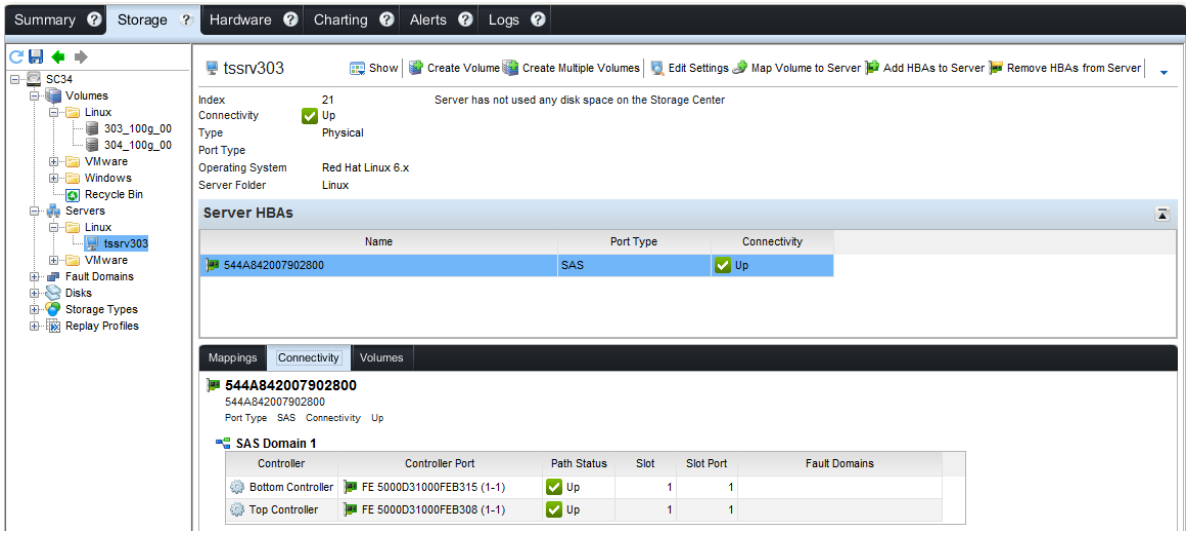
1. Right-click in the **Servers** folder tree and select **Create Server**.



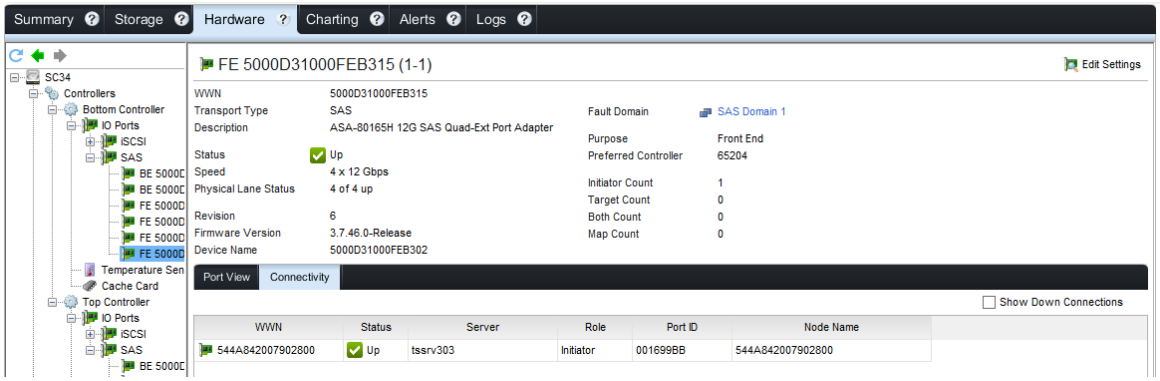
2. Name the new server object, select the proper **Operating System** (Red Hat Linux 6.x or 7.x), select the identified SAS device name corresponding to the Linux host, and click **OK**.



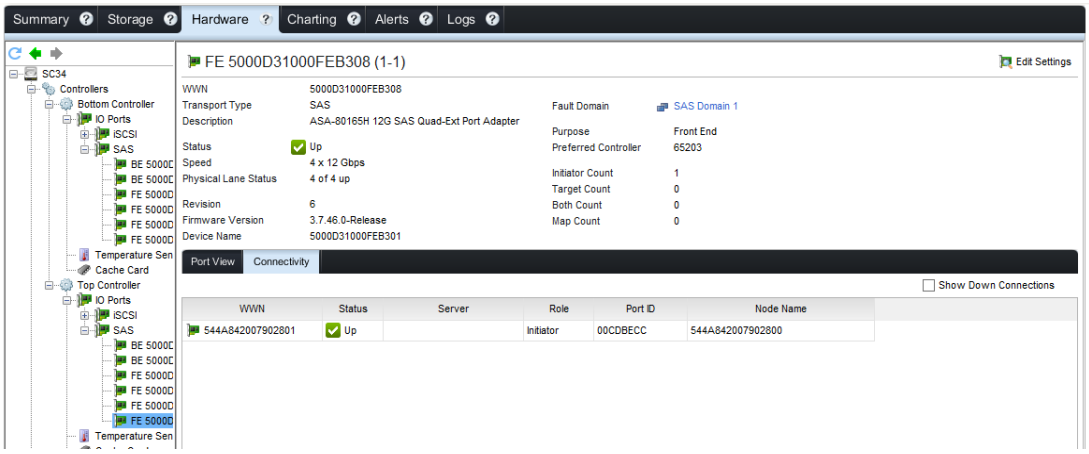
3. The new server object (tssrv303) is created in the **Servers** folder tree, with the **Connectivity** pane displaying that the Linux host has established connectivity to both bottom and top controllers of the SC Series array.



4. Select the SAS controller ending with ***FEB315** in the **Hardware** pane to display the actual SAS WWPN ending with ***2800** in the **Connectivity** pane.



5. Select the SAS controller ending with ***FEB308** in the **Hardware** pane to display the actual SAS WWPN ending with ***2801** in the **Connectivity** pane.



3.3.6 Configured multipath

A properly configured SAS volume will return the following multipath –ll output. This SC Series volume is discovered as a multipath ALUA-capable volume, where each path is capable of I/O. The path represented by **prio=50** (active/optimized path) is used for all active I/O requests. The path represented by **prio=1** (standby path) is a highly available, redundant path and is used when the active/optimized path becomes unavailable.

```
# multipath -ll
Compelnt_0016 (36000d31000feb3000000000000000016) dm-3 COMPELNT,Compellent Vol
size=100G features='1 queue_if_no_path' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 1:0:0:1 sdb 8:16 active ready running
`-+- policy='service-time 0' prio=1 status=enabled
  `- 1:0:1:1 sdc 8:32 active ready running
Compelnt_001a (36000d31000feb300000000000000001a) dm-4 COMPELNT,Compellent Vol
size=100G features='1 queue_if_no_path' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 1:0:1:2 sdd 8:48 active ready running
`-+- policy='service-time 0' prio=1 status=enabled
  `- 1:0:2:2 sde 8:64 active ready running
```

3.3.7 SAS queue depth

Currently supported SAS HBAs default to a queue depth of 254 for every volume. This value should be left at its factory default value unless directed by support or application specific directives.

```
# lsscsi -L | egrep 'COMPELNT|depth'
 [1:0:0:1]    disk    COMPELNT Compellent Vol    0606  /dev/sdb
  queue_depth=254
[1:0:1:1]    disk    COMPELNT Compellent Vol    0606  /dev/sdc
  queue_depth=254
```

3.3.8 Boot from SAS

Boot from SAS functionality is not validated nor supported in use with Dell EMC 12Gbps SAS HBAs on Linux.

3.4 Managing queue depth

There are two locations to configure queue depth for Fibre Channel HBAs. The first location is within the BIOS configuration menu of the HBA. This configuration menu is typically accessed during boot or using the tools provided by the HBA vendor. The queue depth can also be configured using the HBA configuration files that are managed with the Linux modprobe facility within the operating system. If the queue depth value is different in these two locations, the lesser of the two values takes precedence. It is recommended to configure the BIOS to a high value (with 8Gb QLogic HBAs, this value is known as Execution Throttle and defaults to 65535) and subsequently manage the HBA queue depth with its respective operating system configuration file. A queue depth value of 128 is a recommended starting value. Evaluate, test, and adjust the value accordingly to meet individual environment needs.

- For QLogic:

  ```
  options qla2xxx ql2xmaxqdepth=<value>
  ```

- For Emulex:

  ```
  options lpfc lpfc_hba_queue_depth=<value> lpfc_lun_queue_depth=<value>
  ```

The values inside these files can be stacked with other additional key value pairs as needed to define a complete configuration set.

3.5 SCSI device timeout

The SCSI device timeout of discovered SC Series storage volumes defaults to 30 seconds. Verify the setting with the following command.

```
# cat /sys/block/sdX/device/timeout
30
```

This SCSI device timeout value does not need to be typically changed unless instructed or recommended to do so by support or application specific directives. If necessary to change this default value, the following steps are recommended.

To modify the timer that starts the SCSI error handler, run the following command.

```
# echo X > /sys/block/sdX/device/timeout
```

X is measured in seconds. Depending on the Linux distribution, this can also be achieved by modifying the respective udev rule.

To modify the **udev** rule, edit the **/etc/udev/rules.d/60-raw.rules** file and append the following lines.

```
ACTION=="add", SUBSYSTEMS=="scsi", ATTRS{vendor}=="COMPELNT*",
ATTRS{model}=="Compellent Vol*", RUN+="/bin/sh -c 'echo 60
>/sys$DEVPATH/device/timeout'"
```

To make the above udev rule take effect immediately on the configured devices of the host, run the following command.

```
# udevadm trigger –action=add && sleep 2 && multipath –r > /dev/null
```

The above command triggers an **add** to occur within udev which calls the previously configured rule to run against any SC Series volumes found on the host. The shell will then sleep for 2 seconds, allowing the new udev rule to fully complete before reloading the multipathd service to force any multipath enabled devices to take on the timeout of their respective child paths. The **&&** portion of the command instructs the shell to only execute the next commands if the previous command completes successfully.

3.6 /etc/multipath.conf and SC Series volume definition

The SC Series storage device definitions are incorporated into the kernel. Use /etc/multipath.conf within its native syntax constraints as shown in the following. In this example, all other vendor devices are blocked while the SC Series storage devices are identified in the blacklist_exceptions clause by means of their unique WWID values. This /etc/multipath.conf configuration applies only in FC and iSCSI implementations; the SAS /etc/multipath.conf configuration is discussed in section 3.3.2 and should be used instead in SAS implementations.

```
# cat /etc/multipath.conf
# multipath.conf written by anaconda

defaults {
        find_multipaths yes
        user_friendly_names yes
}

blacklist {
        devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
        devnode "^hd[a-z]"
        devnode "^dcssblk[0-9]*"
        device {
                vendor "DGC"
                product "LUNZ"
        }
        device {
                vendor "IBM"
                product "S/390.*"
        }
        # don't count normal SATA devices as multipaths
        device {
                vendor  "ATA"
        }
        # don't count 3ware devices as multipaths
        device {
                vendor  "3ware"
        }
        device {
                vendor  "AMCC"
        }
        # nor highpoint devices
        device {
                vendor  "HPT"
        }
        wwid "20080519"
        wwid "20080519"
        device {
                vendor iDRAC
                product Virtual_CD
        }
        device {
```

```
                vendor PLDS
                product DVD-ROM_DS-8D3SH
        }
        #wwid "*"
}

blacklist_exceptions {
        device {
                vendor "COMPELNT"
                product "Compellent Vol"
        }
}

multipaths {
    multipath {

        wwid "36000d31000006500000000000000017f2"

           alias "vol_001"

        uid 0
        gid 0
        mode 0600
    }
    multipath {

        wwid "36000d31000006500000000000000017f3"

           alias "vol_002"

        uid 0
        gid 0
        mode 0600
    }
    multipath {

        wwid "36000d31000006500000000000000017f4"
  alias "vol_003"
        uid 0
        gid 0
        mode 0600
    }
}
```

An example of the **multipath -ll** command output is shown as follows.

```
[root@dean ~]# multipath -ll
boot_drive (36000d31000035e000000000000000059) dm-0 COMPELNT,Compellent Vol
size=146G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=1 status=active
  |- 1:0:1:0 sdc 8:32 active ready running
  |- 1:0:2:0 sde 8:64 active ready running
  |- 4:0:1:0 sdb 8:16 active ready running
  `- 4:0:2:0 sdd 8:48 active ready running
[snip]
```

Note: With RHEL 7.x, multipath defaults to service-time 0 instead of the round-robin 0 policy; RHEL 6.x releases continues to default to round-robin 0. The service-time 0 policy states, "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", and is recommended to be left in this state unless instructed by support or application specific directives. The service-time 0 policy compensates for any unbalanced I/O patterns and improves throughput. With more balanced I/O patterns, service-time 0 operates akin to the round-robin 0 policy. This policy works with Fibre Channel, iSCSI, and SAS protocols.

3.6.1 Multipath aliases

In the prior example /etc/multipath.conf file, SC Series storage WWID values are further identified by an alias key value (vol_XXX) within the each respective multipath clause. Multipath aliases serve to better identify the function of the volume, and also maintain device name persistence across reboots and reconfiguration. This means that multipath aliases are consistent and safe for use in scripting, mount commands, /etc/fstab syntax, and more.

If multipath aliases are not defined in /etc/multipath.conf, volumes will default to their default naming scheme of /dev/mapper/mpathX while also maintaining persistence across reboots. Designating aliases is recommended and extremely useful when associating multipath device names with more descriptive labels (for example, business function or usage).

After defining aliases or updating any clauses inside **/etc/multipath.conf**, restart the multipathd daemon as follows.

```
# systemctl restart multipathd.service
```

With RHEL 7.x, the older command syntax may still be used to pass the command constructs into the new systemd service management framework.

```
# service multipathd restart
```

3.6.2 Multipath queueing

The compiled defaults of presented and discovered SC Series storage volumes feature the 1 queue_if_no_path option that instructs multipath to automatically queue and retry all I/O requests should all paths become unavailable. This is observed from the following sample output, and should be left in its default state.

```
# multipath -ll
[snip]
vol_001 (36000d31000006500000000000000017f2) dm-3 COMPELNT,Compellent Vol
size=50G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=1 status=active
  |- 1:0:3:1 sdf 8:80  active ready running
  |- 1:0:5:1 sdk 8:160 active ready running
  |- 4:0:4:1 sdg 8:96  active ready running
  `- 4:0:5:1 sdj 8:144 active ready running
[snip]
```

The exception to using this default is in business or application scenarios where I/O is not queued but instead quickly failed if all paths become unavailable. In these circumstances, I/O queueing can be disabled in /etc/multipath.conf either within the defaults clause, to specific devices within the multipath clause, or applied at runtime.

Within the **defaults** clause, insert the following into **/etc/multipath.conf**, then restart the multipath service.

```
devices {
    device {
        vendor "COMPELNT"
        product "^Compellent Vol"
        features 0
        no_path_retry fail
    }
```

Multipath queueing applies to all I/O requests for all transports including Fibre Channel, iSCSI, and SAS presented SC Series volumes.

To implement this to specific device at runtime, issue the following command for the desired multipath device.

```
# dmsetup message /dev/mapper/vol_001 0 "fail_if_no_path"
```

3.7 Multipath environments

During an SC Series **path** or **controller failover** event with the SC Series storage operating **in legacy port mode**, any ports experiencing failure will trigger their respective WWPN identities to momentarily disappear from the SAN fabric before relocating these WWPN identities to a reserve port within the same fault domain on the alternate active controller.

During an SC **path failover** event with SC Series storage operating **in virtual port mode**, the WWPN identities of any NPIV ports experiencing failure will be relocated to another active NPIV port within the same fault domain on the same controller. During a **controller failover** event, the WWPN identities of any NPIV ports experiencing failure will be relocated to the active NPIV ports within the same fault domain on the alternate active controller.

In either failover scenario, the SC Series storage may take anywhere from five up to 60 seconds to propagate these changes through the SAN fabric.

In order to mitigate I/O disruption in multipath scenarios, it is recommended to instruct the HBA code to wait up to 5 seconds before marking a port as down or failed. This minimizes I/O latency to the I/O or application stack above it by quickly relocating I/O requests to alternate and still active HBA paths through the SAN fabric. If all of the paths are down, multipathd will start queuing I/O until one or more paths have recovered. This allows the SC Series array sufficient time to relocate the WWPN identities of the failed ports to an active ports and propagate the changes through the SAN fabric. The configuration syntax for a multipath environment is outlined in the section below.

In SAS-connected environments, paths from both controllers are presented to the connected host (active/optimized and standby), however only the active/optimized path is used for all active I/O at any one time. When the active/optimized path becomes unavailable, the SC Series array will dynamically determine which one of the remaining standby paths will assume the role of the active/optimized path and continue to stream active I/O to the new active/optimized path.

3.7.1 SC Series volume PortDown timeout

These settings dictate how long a Linux system waits before destroying a connection after being losing its connectivity with the port. Configure the following in the **qla2xxx.conf** or **lpfc.conf** file accordingly.

- For QLogic:

  ```
  options qla2xxx qlport_down_retry=5
  ```

- For Emulex:

  ```
  options lpfc lpfc_devloss_tmo=5
  ```

3.7.2 Verifying parameters

To verify that the configuration changes have taken effect, apply the following commands.

- For QLogic:

  ```
  # cat /sys/module/qla2xxx/parameters/qlport_down_retry
  5
  ```

- For Emulex:

  ```
  # cat /sys/class/scsi_host/hostX/lpfc_devloss_tmo
  5
  ```

3.8 Single-path environments

During an SC Series **path** or **controller** failover event with the SC Series storage operating in **legacy port mode**, any ports experiencing failure will trigger their respective WWPN identities to momentarily disappear from the SAN fabric before relocating these WWPN identities to a reserve port within the same fault domain on the alternate active controller.

During an SC Series **path** failover event with SC Series storage operating in **virtual port mode**, the WWPN identities of any NPIV ports experiencing failure will be relocated to another active NPIV port within the same fault domain on the same controller. During a **controller** failover event, the WWPN identities of any NPIV

ports experiencing failure will be relocated to the active NPIV ports within the same fault domain on the alternate active controller.

In order to mitigate I/O disruption in single-pathed scenarios, instruct the HBA driver code to wait up to 60 seconds before marking a port as down or failed. This allows the SC Series storage sufficient time to relocate the WWPN of the failed ports to active ports and to propagate the changes through the SAN fabric. The configuration syntax for a single path environment is shown in the following subsections.

3.8.1 PortDown timeout

These settings dictate how long a Linux system waits before destroying a connection after losing its connectivity with the port. Configure the following in the **qla2xxx.conf** or **lpfc.conf** file accordingly.

- For QLogic:

  ```
  options qla2xxx qlport_down_retry=60
  ```

- For Emulex:

  ```
  options lpfc lpfc_devloss_tmo=60
  ```

3.8.2 Verifying parameters

To verify that the configuration changes have taken effect, apply the following commands.

- For QLogic:

  ```
  # cat /sys/module/qla2xxx/parameters/qlport_down_retry
  60
  ```

- For Emulex:

  ```
  # cat /sys/class/scsi_host/hostX/lpfc_devloss_tmo
  60
  ```

4 Performance considerations

This section provides general information and guidance pertaining to some of the more common performance tuning options and variables available to Linux. This information is not intended to be all encompassing and the values used should not be considered final. This section provides a starting point that Linux and storage administrators can use to fine tune a Linux installation and achieve optimal performance.

Prior to making any changes to the following parameters, establish a good understanding of the current environment and I/O workload. There are numerous methods to accomplish this, including evaluating the perception of the system or storage administrators based on day-to-day experience with supporting the environment. To analyze your environment, the Dell Performance Analysis Collection Kit (DPACK) is a free toolkit that is obtained by sending an email to: DPACK_Support@Dell.com.

Some general guidelines to keep in mind for performance tuning with Linux are:

- Performance tuning is as much an art as it is a science. Since there are a number of variables that impact performance (I/O in particular), specific values cannot be recommended for every environment. Begin with a few variables and add more variables or layers as the system is tuned. For example, start with single path, tune, and then add multipath.
- Make one change at a time and then test, measure, and assess the impact on performance with a performance monitoring tool before making subsequent changes.
- As a best practice, make sure the original settings are recorded so the changes can be reverted to a known state if needed.
- Apply system tuning principles (such as failover) in a non-production environment first (when able) and validate the changes with as many environmental conditions as possible before propagating these changes into production environments.
- If performance needs are being met with the current configuration settings, it is generally a best practice to leave the settings alone to avoid introducing changes that may make the system less stable.
- An understanding of the differences between block- and file-level data should be established in order to effectively target the tunable settings for the most effective impact on performance. Although the SC Series array is a block-based storage device, the support for the iSCSI transport mechanism introduces performance considerations that are typically associated with network- and file-level tuning.
- When validating whether a change is having an impact on performance, leverage the charting feature of Dell Storage Manager to track the performance. In addition, be sure to make singular changes between iterations in order to better track what variables have the most impact (positive or negative) on I/O performance.

4.1 Tuning profiles

RHEL 7.x introduced a new set of tools to assist administrators and storage professionals with tuning RHEL hosts. This is achieved using two new commands, **tuned** and **tuned-adm**, which manage a set of predefined, performance-tuned profiles. The following output presents the active tuning profile (default) as well as the list of alternative tuning profiles that can be applied. It is recommended to use the default tuning profile, **throughput-performance,** with any SC Series storage implementation. The discussion of each tuning profile and its merits are outside the scope of this paper, however further discussion of this topic can be found in online Red Hat documentation.

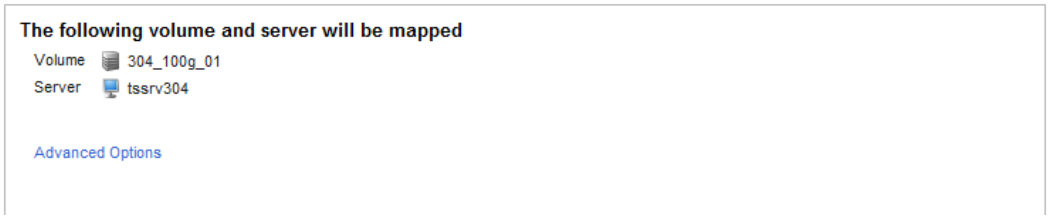
```
# tuned-adm active
Current active profile: throughput-performance

# tuned-adm list
Available profiles:
- balanced
- desktop
- latency-performance
- network-latency
- network-throughput
- powersave
- sap
- throughput-performance
- virtual-guest
- virtual-host
```

4.2 Using multiple volumes

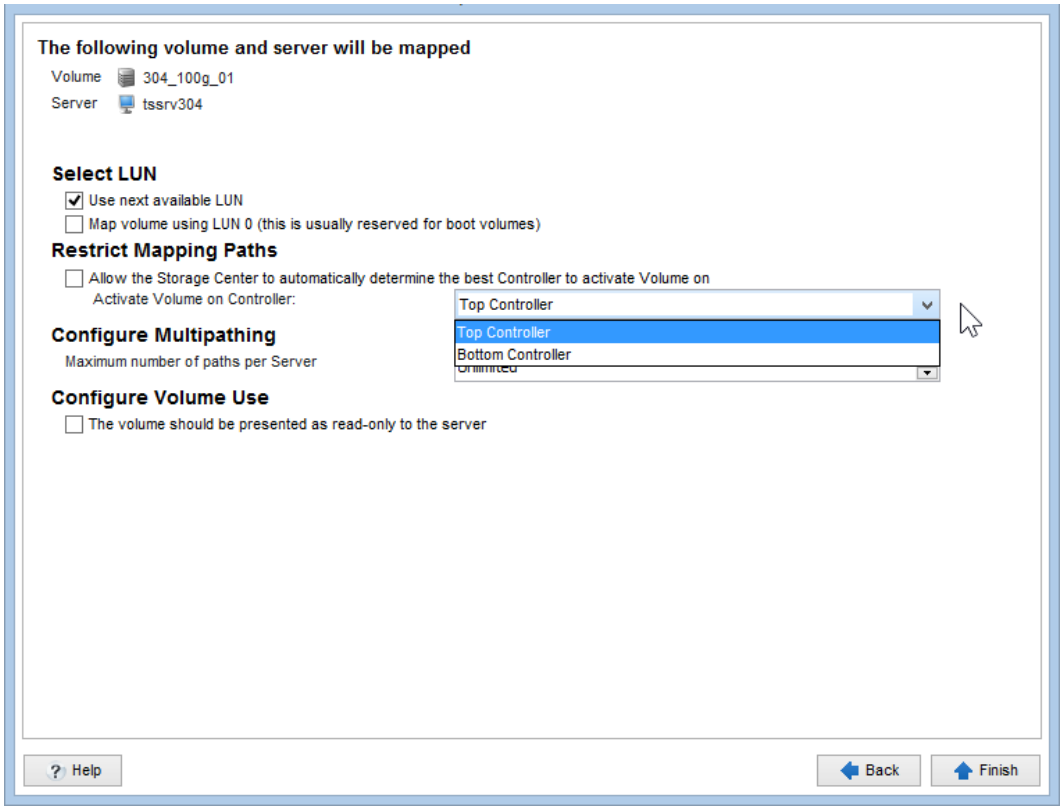
A volume is only active on one SC Series controller at any one time. Therefore, where possible, distribute volume workload evenly across both SC Series storage controllers to most effectively leverage simultaneous I/O processing. A larger number of smaller-sized volumes will often result in better performance than fewer larger-sized volumes. From a Linux perspective, having multiple target volumes can result in performance improvements by leveraging the kernel to process I/O in parallel to addressing multiple paths and SCSI devices.

In SAS-connected environments, paths from both controllers are presented to the connected host (active/optimized and standby), however only the active/optimized path is used for all active I/O at any one time. When the active/optimized path becomes unavailable, the SC Series array will dynamically determine which one of the remaining standby paths will assume the role of the active/optimized path and continue to stream active I/O to the new active/optimized path. This is accomplished by explicitly pinning volumes to a different controller when mapping these volumes to the server object. This feature is accessible using the Advanced Options on the mapping dialog shown as follows.

1. Click the **Advanced Options** link.



2. Uncheck **Restrict Mapping Paths** and select the controller to pin the volume to for mapping the server object.



4.3 Understanding HBA queue depth

Queue depth refers to the number of pending I/O requests. Modifying this value can lead to an improvement in I/O performance in certain workloads. Generally, increasing queue depth can increase throughput, but consideration should be taken because increasing this value can also lead to higher I/O latency. Different applications may benefit from increasing this value, such as environments where the bulk of I/O is small reads and writes. In environments defined by lower IOPS requirements, but needing higher throughput, this is achieved by lowering this queue depth setting until optimal levels of performance are achieved.

This value can be changed in the HBA firmware or in the Linux kernel module for the HBA. Keep in mind that if these two settings have different values, the lower value takes precedence. A good strategy to consider is setting the HBA firmware to the highest number allowable and then tuning this value downward from within the Linux kernel module.

Consult section 3 for details on modifying this value for the particular vendor HBA in use.

4.4 SCSI UNMAP/TRIM

The application and use of SCSI UNMAP/TRIM in filesystem use cases can be very effective towards storage and cost management in the business enterprise. However, consideration should also be made in regard to how this function is implemented.

The discard mount parameter would enable the filesystem to perform real-time, on-demand SCSI UNMAP commands to the SC Series array. In high-I/O-based application landscapes, this may introduce unnecessarily high levels of I/O control traffic as well as raised CPU loads, causing an increase I/O latency and potentially impacting business critical I/O subsequently.

An alternative to this implementation is using the **fstrim** command. The fstrim command is part of the util-linux package, and allows a one-time request to discard used blocks from a mounted filesystem. This command can be scripted (shown in the following sample script), injected into a scheduled cron job, and then applied to a set of mounted filesystems in batch during a time of day that would have less impact to business critical functions.

```
#!/bin/bash

FSTrim=/usr/bin/fstrim

MntPoints="u01 u02 u03 u04 u05"

for i in ${MntPoints}
do
        echo "INFO: Applying ${FSTrim} to mount point ${i}"
        ${FSTrim} -v /${i}
done
```

4.5 SCSI device queue variables

Several Linux SCSI device queue settings can be applied to tune performance. The more common ones are listed in sections 4.5.1 through 4.5.4, with a brief explanation of what each parameter does with regard to I/O. These values are found in the /sys/block/dm-X/queue directory (multipath devices) and /sys/block/sdX/queue directory (block devices) and should be modified for each path device for the intended multipath volume.

4.5.1 I/O scheduler

The /sys/block/dm-X/queue/scheduler parameter and its contents define the I/O scheduler in use by the Linux kernel for SCSI (sd) devices. Some application vendors (such as Oracle) provide specific recommendations for I/O scheduler to use in order to achieve optimal performance with the application platform. By default in RHEL 7.x, this variable is set to deadline as denoted by the [ ] brackets within the file. The deadline scheduler can be further configured using the sysfs tunable subparameters (such as fifo_batch, read_expire, and write_expire) that are discussed in the RHEL 7 Performance Tuning Guide.

```
# cat /sys/block/dm-X/queue/scheduler
noop [deadline] cfq
```

This parameter can be dynamically changed by performing the following command. However, it is recommended to leave the scheduler as **deadline** with all SC Series storage implementations. The scheduler value is also dynamically adjusted according to the use of RHEL 7.x tuning profiles as previously discussed.

```
# echo cfq > /sys/block/dm-X/queue/scheduler
# cat /sys/block/dm-X/queue/scheduler noop deadline [cfq]
```

The scheduler, if changed, applies only to the current running instance of the operating system and only for this specific /dev/sdX device where it is applied. A script could be used to make this change persistent on all required SCSI (sd) devices (on a device specific basis) during boot time. Alternatively, this change can also be applied system wide during boot time by appending the **elevator=** key value option to the end of the kernel string inside of the **/boot/grub2/grub.cfg** boot configuration file as shown in the following.

```
linux16 /vmlinuz-0-rescue-c417dfc159fc4450ac2cc28137506041 root=UUID=35bea1c0-
ce32-42a4-8e36-72fd5e77471d ro rd.lvm.lv=VolGroup00/root
vconsole.font=latarcyrheb-sun16 rd.lvm.lv=VolGroup00/swap crashkernel=auto
vconsole.keymap=us rhgb quiet elevator=deadline
```

4.5.2 read_ahead_kb

This parameter is used when the kernel detects it is sequentially reading from a block device and defines how many kilobytes of I/O the Linux kernel will read. Modifying this value can have a noticeable effect on performance in heavy sequential read workloads. The RHEL 7.x default value for this parameter is 4096 for each block device. This default configuration state is a good starting point.

4.5.3 nr_requests

The nr_requests value is used by the Linux kernel to define the depth of the request queue and is often used in conjunction with changes to the HBA queue depth configuration. The RHEL 7.x default value is 128. Increasing this value sets the I/O subsystem to a larger threshold where it will continue scheduling requests. This keeps the I/O subsystem moving in one direction longer, and can result in more efficient handling of disk I/O. It is recommended as a starting point to increase this value to 1024, then measure, assess, and adjust according to the performance results observed and achieved.

4.5.4 max_sectors_kb

The max_sectors_kb value determines the number of kilobytes that the block layer will allow and accept consistent with the filesystem request. This value must remain equal to or smaller than the maximum allowable size dictated by the hardware layer. It is recommended with SC Series storage, to configure this value to 2048 as a starting point in alignment with the SC Series 2MB default page size, then measure, assess, and adjust according to the performance results observed and achieved.

This configuration is applied to a single block device as shown in the following example. In multipath environments, apply this command to all paths that constitute the multipath device.

```
# echo 2048 > /sys/block/sdX/queue/max_sectors_kb
```

Multipath devices inherit their configuration settings from the block-level device path configuration.

The following example is applied to all SC Series storage devices.

```
# cat >> /etc/udev/rules.d/50-compelnt.rules <<EOT
ACTION=="add|change", SUBSYSTEM=="block", ENV{ID_VENDOR}=="COMPELNT",
ENV{ID_MODEL}=="Compellent Vol", RUN+="/bin/sh -c '/bin/echo 2048 >
/sys%p/queue/max_sectors_kb'"
EOT
```

This udev rule triggers with the discovery of newly presented SC Series storage volumes. To apply this rule to existing SC Series storage volumes run the following command.

```
# udevadm trigger --verbose --subsystem-match=block --property-
match=ID_VENDOR="COMPELNT" && sleep 3 && systemctl reload multipathd.service >
/dev/null
```

```
The increase of the max_sectors_kb value needs to work in conjunction with other
configurable parameters within the I/O stack to enable seamless end to end large
block I/O; some of these additional parameters include but is not limited to
sg_tablesize and the underlying hardware driver layer and its respective
throughput limits.
```

Note: With RHEL 7.4, the max_sectors_kb value can also be configured using the /etc/multipath.conf file within the defaults clause or within each device multipath clause. The configuration value will be applied to all path devices as well as the multipath device.

4.6 iSCSI considerations

Tuning iSCSI is an effort in both Ethernet network and block-level tuning. Evaluate the common Ethernet kernel tunable parameters in order to determine the settings that provide the optimal performance gain with iSCSI. The use of Jumbo frames can lead to improved iSCSI performance when used with 1Gb/10Gb Ethernet. Like Fibre Channel, iSCSI changes should be made individually, incrementally, and evaluated against multiple workload types in order to fully understand the effects on overall performance. In other words, tuning iSCSI is often more time consuming because of the block-level subsystem tuning considerations in addition to network (Ethernet) tuning. A solid understanding of the various Linux subsystem layers involved is necessary to effectively tune the system.

Kernel parameters that can be tuned for performance are found in the /proc/sys/net/core and /proc/sys/net/ipv4 kernel parameters. Once optimal values are determined, permanently set these in the /etc/sysctl.conf file. Like most other modern operating system platforms, Linux can efficiently auto-tune TCP buffers. However by default, some of the settings are conservatively low. Experimenting with the following kernel parameters can lead to improved network performance, and subsequently improve iSCSI performance.

- TCP Max Buffer Sizes:
  - net.core.rmem_max
  - net.core.wmem_max
- Linux Auto-tuning buffer limits:
  - net.ipv4.tcp_rmem
  - net.ipv4.tcp_wmem
- net.ipv4.tcp_window_scaling
- net.ipv4.tcp_timestamps
- net.ipv4.tcp_sack

5 Useful tools

The following native Linux tools can be used to identify and correlate volumes to their respective Storage Center devices.

5.1 The lsscsi command

The lsscsi command is a tool that parses information from the /proc and /sys pseudo filesystems into human readable output. The lsscsi tool is installed using the following command.

```
# yum –y install lsscsi
```

The lsscsi output is shown in the following example, using the grep command to display only SC Series devices. The first column displays the [host:channel:target:lun] designation for each volume. The host number corresponds to the local HBA hostX device file that the volume is mapped to. The channel number is the SCSI bus address and is always zero (0). The target number correlates to the SC Series front-end ports (targets). The lun number represents the LUN ID of the volume on the SC Series controller where it is mapped.

```
# lsscsi | grep COMPELNT | sort -k7
[snip]
[1:0:0:1]    disk    COMPELNT Compellent Vol   0605  /dev/sdb
[1:0:2:1]    disk    COMPELNT Compellent Vol   0605  /dev/sdc
[1:0:3:1]    disk    COMPELNT Compellent Vol   0605  /dev/sdd
[1:0:3:2]    disk    COMPELNT Compellent Vol   0605  /dev/sde
[1:0:5:1]    disk    COMPELNT Compellent Vol   0605  /dev/sdf
[1:0:5:2]    disk    COMPELNT Compellent Vol   0605  /dev/sdg
[4:0:0:1]    disk    COMPELNT Compellent Vol   0605  /dev/sdh
[4:0:2:1]    disk    COMPELNT Compellent Vol   0605  /dev/sdi
[4:0:4:1]    disk    COMPELNT Compellent Vol   0605  /dev/sdj
[4:0:4:2]    disk    COMPELNT Compellent Vol   0605  /dev/sdk
[4:0:5:1]    disk    COMPELNT Compellent Vol   0605  /dev/sdl
[4:0:5:2]    disk    COMPELNT Compellent Vol   0605  /dev/sdm
[snip]
```

5.2 The scsi_id command

The scsi_id command (located in the /usr/lib/udev folder) can be used to report the WWID of volumes. This WWID can then be used to correlate volumes to their respective SC Series storage devices.

This script applies the scsi_id command and correlates Linux device names (/dev/sdX) to their respective WWID values.

```
#!/bin/bash

SCSIID=/usr/lib/udev/scsi_id

#OSMajor=`cat ${ReleaseFile} | awk '{print $7}' | cut -d. -f1`
OSMajor=`uname -r | awk -F. '{print $4}'`

echo "INFO: OS Major Rev. ${OSMajor} detected!"

if [ "${OSMajor}" = "el7" -o "${OSMajor}" = "el6" ]; then
        {
        if [ "${OSMajor}" = "el6" ]; then
        {
                SCSIID=/sbin/scsi_id
        }
        for i in `cat /proc/partitions | awk {'print $4'} | grep sd`
        do
                echo "Device: $i WWID: `${SCSIID} --page=0x83 --whitelisted --
device=/dev/$i`"
        done
        } | sort -k4 $1
elif [ "${OSMajor}" = "el5" ]; then
        {
        for i in `cat /proc/partitions | awk {'print $4'} | grep sd`
        do
                echo "Device: $i WWID: `${SCSIID} -g -u -s /block/$i`"
        done
        } | sort -k4 $1
else
        echo "WARN: OSMajor parameter of unknown value, Exiting"
        exit 1
fi
```

Sample output from this script is shown as follows.

```
# ./get_WWID.sh
INFO: OS Major Rev. el7 detected!
[snip]
Device: sdd WWID: 36000d31000006500000000000000017f2
Device: sdf WWID: 36000d31000006500000000000000017f2
Device: sdj WWID: 36000d31000006500000000000000017f2
Device: sdl WWID: 36000d31000006500000000000000017f2
[snip]
```

These WWID values in turn, correlate to the Serial Number value of the SC Series storage devices.

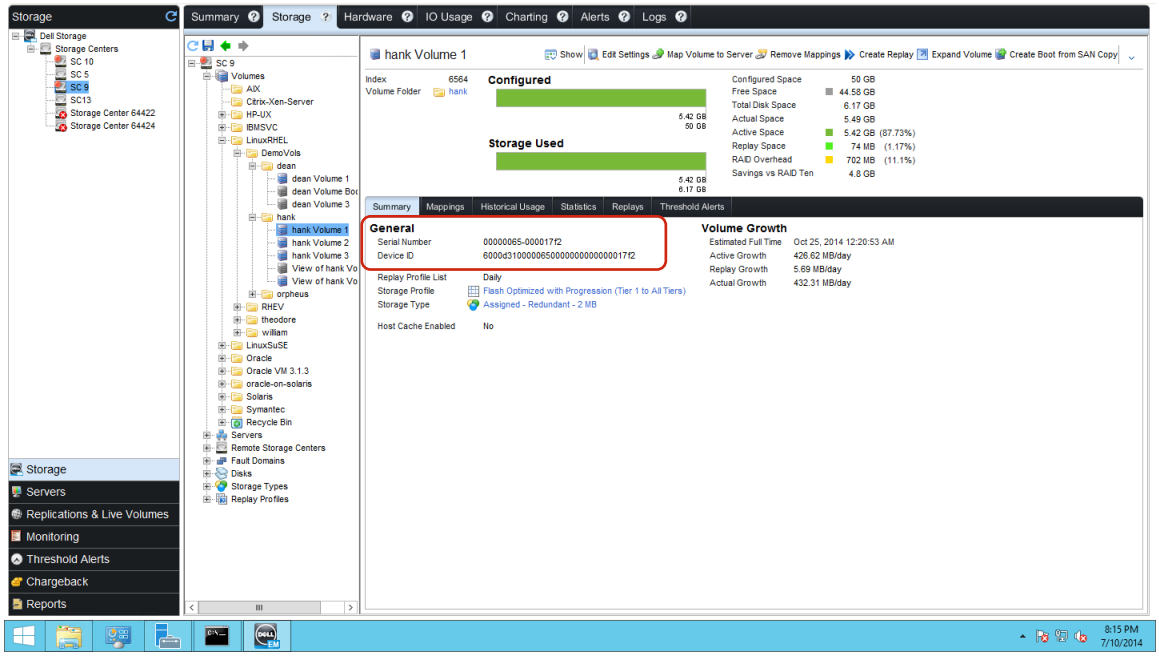


Figure 1 Serial Number field

5.3 The dmsetup command

The dmsetup command can be used to view, manage, and change the metadata of logical volumes that are managed by the device-mapper driver. In the following example, the **dmsetup** command is used to display the **Name** and **UUID** values of this SC Series volume (the UUID value is a concatenation of the **mpath-** prefix to the volume serial number value).

```
# dmsetup info /dev/mapper/vol_001
Name:              vol_001
State:             ACTIVE
Read Ahead:        256
Tables present:    LIVE
Open count:        0
Event number:      0
Major, minor:      253, 4
Number of targets: 1
UUID: mpath-36000d31000006500000000000017f2
```

5.4 The dmesg command

The dmesg command is useful for discovering which device names are assigned to recently discovered volumes. The following output demonstrates the discovery and assignment of a new SC Series volume to the /dev/sdh device file.

```
# dmesg
[snip]
[1203830.267568] scsi 4:0:0:1: Direct-Access     COMPELNT Compellent Vol   0605
PQ: 0 ANSI: 5
[1203830.267996] sd 4:0:0:1: Attached scsi generic sg8 type 0
[1203830.268089] sd 4:0:0:1: [sdh] 20971520 512-byte logical blocks: (10.7
GB/10.0 GiB)
[1203830.268093] sd 4:0:0:1: [sdh] 4096-byte physical blocks
[1203830.268847] sd 4:0:0:1: [sdh] Write Protect is off
[1203830.268858] sd 4:0:0:1: [sdh] Mode Sense: 8f 00 00 08
[1203830.269033] sd 4:0:0:1: [sdh] Write cache: disabled, read cache: enabled,
doesn't support DPO or FUA
[snip]
```

5.5 The /proc/scsi/scsi file

The /proc/scsi/scsi file can provide additional detail about volumes and targets on a Linux host.

```
# cat /proc/scsi/scsi
Attached devices:
[snip]
Host: scsi1 Channel: 00 Id: 00 Lun: 01
  Vendor: COMPELNT Model: Compellent Vol   Rev: 0605
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi1 Channel: 00 Id: 02 Lun: 01
  Vendor: COMPELNT Model: Compellent Vol   Rev: 0605
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi1 Channel: 00 Id: 03 Lun: 01
  Vendor: COMPELNT Model: Compellent Vol   Rev: 0605
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi1 Channel: 00 Id: 03 Lun: 02
  Vendor: COMPELNT Model: Compellent Vol   Rev: 0605
  Type:   Direct-Access                    ANSI  SCSI revision: 05
[snip]
```

6 Dell Storage REST API

The Dell Storage REST API interface is available with the installation of Dell Storage Manager (DSM) 2015 R3 or newer. The REST API is recommended when intending to interact with SC Series storage managed by the DSM installation, either programmatically or in a command line interface. For more information, refer to the *Dell Storage Representational State Transfer (REST) API Cookbook*.

The use of the Compellent Command Utility (CompCU) has been deprecated.

A Configuration details

Component	Description
Operating system	Red Hat Enterprise Linux 7.3 (Maipo)
Driver version	Driver version = 8.04.00.04.06.3-k-debug BIOS version = 3.00
Firmware version	Firmware version = 5.06.05 (90d5)
Application	NA
Cabling	QLE2562 8Gb Dual Port PCIe FC HBA Dell EMC 12Gbps SAS HBA
Server	Dell PowerEdge™ R630 x 2
Storage	Dell SC8000, SCOS 6.6.x, virtual port mode Dell SCv2x00, SCOS 6.6.x, virtual port mode
Switch	Dell PowerConnect™ 5500 Series

B Technical support and resources

Dell.com/support is focused on meeting customer needs with proven services and support.

Dell TechCenter is an online technical community where IT professionals have access to numerous resources for Dell EMC software, hardware, and services.

Storage Solutions Technical Documents on Dell TechCenter provide expertise that helps to ensure customer success on Dell EMC storage platforms.

B.1 Additional resources

- Dell Storage Manager 2016 R3 Administrator's Guide: http://topics-cdn.dell.com/pdf/dell-compellent-sc4020_administrator%20guide_en-us.pdf
- Dell Storage Manager 2016 R3 Web UI Administrator's Guide: http://topics-cdn.dell.com/pdf/dell-compellent-sc4020_Administrator%20Guide5_en-us.pdf
- Dell Storage Center SC9000 Owner's Manual: http://topics-cdn.dell.com/pdf/storage-sc9000_Owner's%20Manual_en-us.pdf
- Red Hat Enterprise Linux 7.x System Administrator's Guide: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/pdf/System_Administrators_Guide/Red_Hat_Enterprise_Linux-7-System_Administrators_Guide-en-US.pdf
- Red Hat Enterprise Linux Document Portal: https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/
- Red Hat Labs (beta): https://access.redhat.com/labs/