SUSE® Linux Enterprise Server

10 SP3/SP4

www.novell.com

January 19, 2011

SLES 10 SP3/SP4: Storage Administration Guide

A NOVELL BUSINESS

SLES 10 SP3/SP4: Storage Administration Guide

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. See the Novell International Trade Services Web page [`http://www.novell.com/info/exports/`] for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the latest online documentation for this and other Novell products, see the Novell Documentation Web page [`http://www.novell.com/documentation`].

Novell Trademarks

For Novell trademarks, see the Novell Trademark and Service Mark list [`http://www.novell.com/company/legal/trademarks/tmlist.html`].

Third-Party Materials

All third-party trademarks and copyrights are the property of their respective owners.

Contents

About This Guide

This guide provides information about how to manage storage devices on a SUSE Linux Enterprise Server 10 Support Pack 3 or 4 server. The guide also contains information on using the Enterprise Volume Management System (EVMS) 2.5.5 or later to manage devices. Related storage administration issues are also covered as noted below.

Audience

This guide is intended for system administrators.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation, or go to www.novell.com/documentation/feedback.html and enter your comments there.

Documentation Updates

For the most recent version of the *SUSE Linux Enterprise Server 10 Storage Administration Guide for EVMS*, visit the Novell Documentation Web site for SUSE Linux Enterprise Server 10 [http://www.novell.com/documentation/sles10].

Additional Documentation

For information about managing storage with the Linux Volume Manager (LVM), see the *SUSE Linux Enterprise Server 10 Installation and Administration Guide* [http://www.novell.com/documentation/sles10].

For information about iSNS (Internet Storage Name Service), see “iSNS” [http://www.novell.com/documentation/sles10/sles_admin/index.html?page=/documentation/sles10/sles_admin/data/cha_isns.html] in the *SUSE Linux Enterprise Server 10 Installation and Administration Guide* [http://www.novell.com/documentation/sles10].

1 Overview of EVMS

The Enterprise Volume Management System (EVMS) 2.5.5 management tool for Linux is an extensible storage management tool that integrates all aspects of volume management, such as disk partitioning, the Logical Volume Manager (LVM), the Multiple-Disk (MD) manager for software RAIDs, the Device Mapper (DM) for multipath I/O configuration, and file system operations.

1.1 Benefits of EVMS

EVMS provides the following benefits:

- An open source volume manager
- A plug-in framework for flexible extensibility and customization
- Plug-ins to extend functionality for new or evolving storage managers

- Support for foreign partition formats
- Cluster-aware

1.2 Plug-In Layers

EVMS abstracts the storage objects in functional layers to make storage management more user-friendly. The following table describes the current EVMS plug-in layers for managing storage devices and file systems:

Table 1.1 *EVMS Plug-In Layers*

Storage Managers	Description	Plug-Ins
Device	Manages the physical and logical devices	Device Mapper (DM)
Segment	Manages the partitioning of physical and logical devices into smaller segments of free space. Segment managers can be stacked. For example, a cluster segment can contain other storage objects or volumes.	Uses Device Mapper (DM) Segment managers include DOS, GPT, System/390 (S/390), Cluster, BSD, Mac, and BBR For more information, see Section 4.1, "Understanding Disk Segmentation" (page 41).
Regions	Manages the combination of multiple storage objects	LVM/LVM2 for containers and region, MD for RAIDs, and DM for multipath I/O
EVMS Features	Manages EVMS features	Drive linking (linear concatenation), Bad Block Relocation (BBR), and Snapshot

Storage Managers	Description	Plug-Ins
File System Interface Modules (FSIM)	Manages the interface between the file system managers and the segment managers	For information, see Section 1.3, “Supported File Systems” (page 3).
Cluster Manager Interface Modules	Manages the interface between the cluster manager and the file systems and devices	HeartBeat 2

1.3 Supported File Systems

EVMS supports the following Linux file systems:

- EXT3
- ReiserFS
- XFS
- OCFS2
- JFS
- EXT2
- Swap
- NTFS (read only)
- FAT (read only)

For more information about file systems supported in SUSE Linux Enterprise Server 10, see the *SUSE Linux Enterprise Server 10 Installation and Administration Guide.* [http://www.novell.com/documentation/sles10].

The Novell Storage Services (NSS) file system is also supported when used with the Novell Open Enterprise Server 2 for SUSE Linux Enterprise Server 10 SP1 (or later versions of OES 2 and SLES 10).

The *File System Primer* [http://wiki.novell.com/index.php/File_System_Primer] describes the variety of file systems available on Linux and which ones are the best to use for which workloads and data.

1.4 Terminology

EVMS uses the following terminology in the EVMS user interface:

Table 1.2 *EVMS Terms*

Term	Description
Sector	The lowest level that can be addressed on a block device.
Disk	A physical disk or a logical device.
Segment	An ordered set of physically contiguous sectors on a single device. It is similar to traditional disk partitions.
Region	An ordered set of logically contiguous sectors that might or might not be physically contiguous. The underlying mapping can be to logical disks, disk segments, or other storage regions.
Feature (Feature Object, EVMS Feature, EVMS Object)	A logically contiguous address space created from one or more disks, segments, regions, or other feature objects through the use of an EVMS feature.
Storage Object	Any storage structure in EVMS that is capable of being a block device. Disks, segments, regions, and feature objects are all storage objects.

Term	Description
Container (Storage Container)	A collection of devices that is managed as a single pool of storage. **Private Storage Container:** A storage container that is exclusively owned and accessed by only one server. **Cluster Storage Container:** A storage container managed by the Cluster Resource Manager. It is accessible to all nodes of a cluster. An administrator can configure the storage objects in the cluster container from any node in the cluster. Cluster containers can be private, shared, or deported. • **Private:** The cluster container is exclusively owned and accessed by only one particular node of a cluster at any given time. The ownership can be reassigned by failover policies or the administrator. • **Shared:** The cluster container is concurrently owned and accessed by all nodes of a cluster. Shared containers are preferred for distributed databases, clustered file systems, and cluster-aware applications that can coordinate safe access to shared volumes. • **Deported:** The cluster container is not owned or accessed by any node of the cluster.
Volume (Logical Volume)	A mountable storage object. Logical volumes can be EVMS volumes or compatibility volumes. • **EVMS Volume:** Volumes that contain EVMS metadata and support all EVMS features. Device nodes for EVMS volumes are stored in the `/dev/evms` directory. For example: `/dev/evms/my_volume` • **Compatibility Volume:** Volumes that are backward-compatible to other volume managers. They do not contain EVMS metadata and cannot support EVMS features.

1.5 Location of Device Nodes for EVMS Storage Objects

EVMS creates a unified namespace for the logical volumes on your system in the `/dev/evms` directory. It detects the storage objects actually present on a system, and creates an appropriate device node for each one, such as those shown in the following table.

Table 1.3 *Device Node Location*

Storage Object	Standard Location the Device Node	EVMS Location of the Device Node
A disk segment of disk	`/dev/sda5`	`/dev/evms/sda5`
A software RAID device	`/dev/md1`	`/dev/evms/md/md1`
An LVM volume	`/dev/lvm_group/lvm_volume`	`/dev/evms/lvm/lvm_group/lvm_volume`

Using EVMS to Manage Devices

2

This section describes how to configure EVMS as the volume manager of your devices.

- Section 2.1, “Configuring the System Device at Install to Use EVMS” (page 7)
- Section 2.2, “Configuring an Existing System Device to Use EVMS ” (page 20)
- Section 2.3, “Configuring LVM Devices to Use EVMS” (page 28)
- Section 2.4, “Using EVMS with iSCSI Volumes” (page 29)
- Section 2.5, “Using the ELILO Loader Files (IA-64)” (page 30)
- Section 2.6, “Starting EVMS” (page 30)
- Section 2.7, “Starting the EVMS Management Tools” (page 30)

2.1 Configuring the System Device at Install to Use EVMS

This section describes how to configure the system device during the Linux install to use EVMS as the volume manager instead of the current default of Linux Volume Manager (LVM).

- Section 2.1.1, “Before the Install” (page 8)

- Section 2.1.2, “During the Server Install” (page 10)
- Section 2.1.3, “After the Server Install” (page 16)

2.1.1 Before the Install

- Section “System Device” (page 8)
- Section “Device Size Limits” (page 9)
- Section “Data Loss Considerations for the System Device” (page 9)
- Section “Storage Deployment Considerations for the System Device” (page 10)

System Device

For the purposes of this install documentation, a system device is any device that contains the Linux `/boot`, `swap`, or root (`/`) partitions for your Linux computer.

The install instructions assume the following:

- All three system partitions are on the same physical disk.

  If you use different disks for any of the system partitions, make sure to modify the install instructions for your deployment scenario so that all of the system partitions are managed by EVMS.

- You must configure the boot partition within the BIOS-addressable space (such as 2 GB for x86 or 8 GB for x86-64) of the first disk recognized by the system.

  If this restriction is not required for your hardware, you can modify the location of the `/boot` partition as desired.

- Your system uses the Grub or LILO boot loaders.

  If you have an IA64 system, you must modify these install instructions to use the ELILO boot loader (`/boot/efi/elilo.conf`) instead.

WARNING

Whenever you manually alter the kernel or `initrd` on your system, make sure to run `/sbin/elilo` before shutting down the computer. If you leave out this step, your system might not be bootable.

Device Size Limits

Version 2.3 and later of `mdadm` supports component devices up to 4 TB in size each. Earlier versions support component devices up to 2 TB in size.

IMPORTANT

If you have a local disk, external disk arrays, or SAN devices that are larger than the supported device size, use a third-party disk partitioner to carve the devices into smaller logical devices.

You can combine up to 28 component devices to create the RAID array. The `md` RAID device you create can be up to the maximum device size supported by the file system you plan to use. For information about file system limits for SUSE Linux Enterprise Server 10, see “Large File System Support” in the *SUSE Linux Enterprise Server 10 Installation and Administration Guide*. [`http://www.novell.com/documentation/sles10`].

Data Loss Considerations for the System Device

This install requires that you delete the default partitioning settings created by the install, and create new partitions to use EVMS instead. This destroys all data on the disk.

WARNING

To avoid data loss, it is best to use the EVMS install option only on a new device.

If you have data volumes on the system device, take one or more of the following precautionary measures:

- Move the data volumes from the system device to another device.

- If you cannot move the volumes, make a backup copy of the data, so you can restore the data volumes later from a backup copy.

Storage Deployment Considerations for the System Device

By default, the YaST install for SUSE Linux Enterprise Server uses the Linux Volume Manager to manage the system device. The install procedures in this section describe how to install SUSE Linux Enterprise Server with EVMS as the volume manager of the system device. The instructions assume the following:

- You want to use EVMS to manage the system device.

- Only the system device is to be configured during the install.

- Other devices on the system are not configured during the install, or are attached to the server later. These additional devices are configured only after the system is operating and performing as expected.

2.1.2 During the Server Install

To install Linux with EVMS as the volume manager for your boot and system partitions, you must modify the Partitioning configuration in the Installation Settings.

WARNING

The following procedure destroys all data on the system device.

1. Begin the install, according to the instructions provided in Deployment [http://www.novell.com/documentation/sles10/sles_admin/data/part_setup.html] in the *SUSE Linux Enterprise 10 Installation and Administration Guide* [http://www.novell.com/documentation/sles10/sles_admin/data/bookinfo_book_sles_admin.html].

2. When the installation reaches the Installations Settings screen, delete the proposed LVM-based partioning solution. This deletes the proposed partitions and the partition table on the system device so that the device can be marked to use EVMS as the volume manager instead of LVM.

2a In the list of Installation Settings, select *Partitioning*.

2b In the Partitioning menu, select *Create Custom Partition Setup*, then click *Next*.

2c Select *Custom Partition - for Experts*, then click *Next* to open the *Expert Partitioner* dialog box.

2d Select *Expert > Delete Partition Table and Disk Label*, then click *Yes* twice to continue through the Warning advisories.

This deletes the recommended partitions and the partition table on the system disk.

3 Create a primary partition on the system disk to use as the boot partition:

3a Click *Create*.

3b From the list of devices, select the device you want to use for the boot partition, such as `/dev/hda`, then click *OK*.

If you have a single system disk, only one device is available, and you are not prompted to choose the device.

3c Select *Primary Partition*, then click *OK*.

3d Select *Format*, then select the native Linux file system you want to use, such as Ext3.

IMPORTANT

In a paravirtualized environment, use Ext2 as the file system for the boot device.

3e In *Size (End Value)* field, specify 200 MB or larger.

For example, to set the size at 300 MB, type `300M`.

3f Set the mount point to */boot*.

3g Click *OK*.

The partition appears as a logical device in the devices list, such as `/dev/hda1`.

4 Create a second primary partition on the system disk to use for both the swap and system volumes:

4a Click *Create*.

4b From the list of devices, select the device you want to use for the second primary partition, such as `/dev/hda`, then click *OK*.

If you have a single system disk, only one device is available and you are not prompted to choose the device.

4c Select *Primary Partition*, then click *OK*.

4d Select *Do Not Format*, then select *Linux LVM (0x8E)* from the list of file system IDs.

4e In *Size* (*End Value* field), set the cylinder End value to 5 GB or larger, depending on the combined partition size you need to contain your system and swap volumes.

IMPORTANT

Do not make the system partition larger than necessary. The remaining space on the system disk can be used to create NSS volumes or native Linux volumes that are managed by EVMS.

To determing how much space to use, consider the following recommendations:

- For your system volume, allow 2 GB (minimum) to 10 GB (recommended), depending on the OES services that you intend to install.
- If you intend to create additional NSS volumes on the same physical disk, you must leave unpartitioned space available.

- Set aside 128 MB or larger for the swap volume.

  Swap management is different for Linux kernel 2.4.10 and later. How much swap to add depends on the RAM size, the tasks that are planned for the system, and whether you want to make more virtual memory available than the RAM provides.

  Some swap (at least 128 MB) is good to have to minimize the risk of losing data when active processes run out of RAM space. Swap is not required for systems with more than 1 GB of RAM. You must have at least 1 GB of virtual memory (RAM plus swap) during the install, but if the swap is more than 2 GB, you might not be able to install on some machines.

- The total size should be the size you need for your system volume plus the size you need for your swap volume.

For example, if you have a 20 GB hard drive with 2 GB of RAM and plan to install all of the OES services on the system volume, your system partition should be at least 11 GB. The remaining 9 GB should remain as free unpartitioned space that can be used for NSS volumes or other Linux partitions that you might want to create later.

4f Click *OK*.

The partition appears as a logical device in the devices list, such as `/dev/hda2`.

5 Modify the volume management type from LVM to EVMS for the second primary partition you created in Step 4 (page 12):

5a At the bottom of the page, click *EVMS*.

Available partitions for EVMS appear as devices under `/dev/evms`, such as `/dev/evms/hda2`.

5b In the EVMS Configurator, select the LVM partition created in Step 4 (page 12), then click *Create Container*.

5c In the Create EVMS Container dialog box, select the partition, specify the container name (such as `system`), then click *Add Volume* to create the `lvm/`*`system`* container, where *`system`* is the container name.

5d Click *OK*.

The EVMS Configurator displays the `lvm/`*`system`* container you just created, its size, and free space.

6 Create the swap volume in the `lvm/`*`system`* container:

6a Select `lvm/system`, then click *Add*.

6b In the Create Logical Volume dialog box, select *Format*, then select *Swap* from the *File System* drop-down menu.

6c Specify `swap` as the volume name.

6d Specify 1 GB (recommended) for the swap volume.

The swap size should be 128 MB or larger, with a recommended size of 1 GB. For an explanation of this recommendation, see Step 4e (page 12).

6e Specify the mount point as *swap*.

6f Click *OK*.

7 Create the system volume in the `lvm/`*`system`* container:

7a Select `lvm/system`, then click *Add*.

7b In the Create Logical Volume dialog box, select *Format*, then select the file system to use from the *File System* drop-down menu, such as *Reiser or Ext3*.

7c In the Volume Name field, specify a volume name, such as `sys_lx`.

7d In the *Size* field, click Max to set the size of the system volume as the remaining space available in the `lvm/`*`system`* partition.

7e Specify the mount point as `/` (root volume).

7f Click *OK*.

8 Click *Next* to return to the list of devices.

Below is an example of the physical and logical devices that should be configured on your system. Your setup depends on the number of devices in the server and the sizes you choose for your partitions.

Device	Size	F	Type	Mount	Start	End	Used By
`/dev/hda`	149.0 GB		6Y160p0		0	19456	
`/dev/hda1`	305.9 MB	F	Linux Native (Reiser)	`/boot`	0	38	
`/dev/hda2`	20.0 GB		Linux LVM		39	2649	EVMS `lvm/system`
`/dev/hdb`	111.8 GB		SP1203N		0	14595	
`/dev/evms/lvm/system/sys_lx`	10.0 GB	F	EVMS	`/`	-	-	
`/dev/evms/lvm/system/swap`	1.0 GB	F	EVMS	`swap`	-	-	

9 Click *Next* to return to the Installation Settings page.

You can dismiss the message warning that you should not mix EVMS and non-EVMS partitions on the same device.

10 Continue with the SUSE Linux Enterprise Server installation.

IMPORTANT

After the install is complete, make sure to perform the mandatory post-install configuration of the related system settings to ensure that the system device functions properly under EVMS. Otherwise, the system fails to boot properly.

For information, see Section 2.1.3, “After the Server Install” (page 16).

2.1.3 After the Server Install

After the SUSE Linux Enterprise Server 10 install is complete, you must perform the following tasks to ensure that the system device functions properly under EVMS:

- Section “Edit the /etc/fstab File” (page 16)
- Section “Make a New initrd” (page 18)
- Section “Disable the boot.lvm and boot.md Services” (page 18)
- Section “Enable the boot.evms Service” (page 19)
- Section “Restart the Server” (page 19)
- Section “Verify the System Services” (page 19)

Edit the /etc/fstab File

When you boot the system, the kernel reads the `/etc/fstab` file to identify which file systems should be mounted and then mounts them. This file contains a table of file system information about the root (`/`), `/boot`, and `swap` partitions plus other partitions and file systems you want to mount.

The `/boot` partition is separate from the EVMS container where you placed the root (`/`) and `swap` partitions and is not managed by EVMS at this time. However, in the following steps, you disable `boot.lvm` and `boot.md`, then enable `boot.evms`. In effect, this forces EVMS to scan all the partitions at boot time, including the `/boot` partition, and it activates `/boot` under the `/dev/evms` directory. Therefore, this makes `/boot` a partition that is discovered by EVMS at startup, and requires that the device be listed under `/dev/evms` in the `fstab` file so it can be found when booting with `boot.evms`. You must edit the `/etc/fstab` file to modify the location of the `/boot` partition so it is under the `/dev/evms` directory.

In `fstab`, the entry for the boot device might present the boot device by the device node name (such as `/dev/sda1`) or by the UUID pathname (such as `/dev/disk/by-id/scsi-SServeRA_Drive_1_600BC00000-part1`). In ether case, that name for the boot device must be changed to include `evms` in the path, such as `/dev/evms/sda1`.

The procedure in this section shows how to change `/dev/`*`sda1`* to `/dev/evms/`*`sda1`*. Replace *`sda1`* with the device name of the device you used for your `/boot` partition.

IMPORTANT

When working in the `/etc/fstab` file, do not leave any stray characters or spaces in the file. This is a configuration file, and it is highly sensitive to such mistakes.

To modify the path of the boot device in the `/etc/fstab` file, complete the following procedure:

1 Open the `/etc/fstab` file in a text editor.

2 Locate the line that contains the `/boot` partition.

For example, if your `/boot` partition uses device `sda1` and the Reiser file system, look for a line similar to this:

```
/dev/sda1 /boot reiser defaults 1 1
```

3 In the *Device Name* column, modify the location of the `/boot` partition from `/dev` to `/dev/evms` so it can be managed by EVMS. Modify only the device name by adding `/evms`to the path:

```
/dev/evms/sda1 /boot reiser defaults 1 1
```

4 Save the file.

The changes do not take effect until the server is restarted. Do not restart at this time.

5 Continue with Section “Make a New initrd” (page 18).

Make a New initrd

1 Open a terminal console, and log in as the `root` user.

2 At the console prompt, enter

```
mkinitrd
```

This creates a new `initrd` file with the correct settings for the boot device. The changes do not take effect until the server is restarted. Do not restart at this time.

3 Continue with Section “Disable the boot.lvm and boot.md Services” (page 18).

Disable the boot.lvm and boot.md Services

Disable the `boot.lvm` and `boot.md` services so they do not run at boot time (runlevel B). EVMS now handles the boot.

1 In YaST, click *System* > *System Services (Runlevel*) > *Expert Mode*.

2 Select *boot.lvm*.

3 Click *Set/Reset* > *Disable the Service*.

4 Select *boot.md*.

5 Click *Set/Reset* > *Disable the Service*.

6 Click *Finish*, then click *Yes*.

The changes do not take effect until the server is restarted. Do not restart at this time.

7 Continue with Section “Enable the boot.evms Service” (page 19).

Enable the boot.evms Service

The`boot.evms` service should be enabled automatically after the install, but you should verify that it is enabled.

1 In YaST, click *System* > *System Services (Runlevel)* > *Expert Mode*.

2 Select *boot.evms*.

3 Click *Set/Reset* > *Enable the Service*.

   The *B runlevel* option is automatically selected.

4 Click *Finish*, then click *Yes*.

   The changes do not take effect until the server is restarted.

   NOTE

   Effective in SUSE Linux Enterprise 10, the `/dev` directory is on `tmpfs`, and the device nodes are automatically re-created on boot. It is no longer necessary to modify the `/etc/init.d/boot.evms` script to delete the device nodes on system restart, as was required for previous versions of SUSE Linux.

5 Continue with Section "Restart the Server" (page 19).

Restart the Server

1 Restart the server to apply the post-install configuration settings.

2 On restart, if the system device does not appear, it might be because EVMS has not been activated. At the prompt, enter

```
evms_activate
```

Verify the System Services

After the post-install configuration is complete and you have restarted the server, make sure the server is operating as expected.

2.2 Configuring an Existing System Device to Use EVMS

If you have already installed Linux with a different volume manager for the system device (that is, the devices where you installed the `/boot`, `swap`, or root (`/`) partitions), you can optionally configure the device for EVMS at any time after the install.

If you do not configure the device to use EVMS, you must manage the device and all of its volumes with its current volume manager (the default is LVM), and free space on the device cannot be used for volumes you want to create using EVMS. Beginning with the Linux 2.6 kernel, a given device cannot be managed by multiple volume managers. However, you can have different volume managers for different devices.

The following procedures assume that you installed Linux with three partitions on a single SCSI device named `sda`:

```
/dev/sda1  reiserfs /boot

/dev/sda2  swap     swap

/dev/sda3  reiserfs /
```

IMPORTANT

Make sure to modify the following procedures as necessary for your specific setup.

- Section 2.2.1, “Disable the boot.lvm and boot.md Services” (page 21)
- Section 2.2.2, “Enable the boot.evms Service” (page 21)
- Section 2.2.3, “Edit the /etc/fstab File” (page 22)
- Section 2.2.4, “Edit the Boot Loader File” (page 23)
- Section 2.2.5, “Force the RAM Disk to Recognize the Root Partition” (page 26)
- Section 2.2.6, “Restart the Server” (page 27)
- Section 2.2.7, “Verify that EVMS Manages the Boot, Swap, and Root Partitions” (page 27)

2.2.1 Disable the boot.lvm and boot.md Services

You need to disable `boot.lvm` (handles devices for Linux Volume Manager) and `boot.md` (handles multiple devices in software RAIDs) so they do not run at boot time. In the future, you want `boot.evms` to run at boot time instead.

1 In YaST, click *System* > *Runlevel Editor* > *Expert Mode*.

2 Select *boot.lvm*.

3 Click *Set/Reset* > *Disable the Service*.

4 Select *boot.md*.

5 Click *Set/Reset* > *Disable the Service*.

6 Click *Finish*, then click *Yes*.

The changes do not take effect until the server is restarted. Do not restart at this time.

7 Continue with Section 2.2.2, “Enable the boot.evms Service” (page 21).

2.2.2 Enable the boot.evms Service

You need to enable the `boot.evms` service so that it boots devices when you restart the server.

1 In YaST, click *System* > *Runlevel Editor* > *Expert Mode*.

2 Select *boot.evms*.

3 Click *Set/Reset* > *Enable the Service*.

The *B runlevel* option is automatically selected.

4 Click *Finish*, then click *Yes*.

The changes do not take affect until the server is restarted. Do not restart at this time.

NOTE

Effective in SUSE Linux Enterprise 10, the `/dev` directory is on `tmpfs` and the device nodes are automatically re-created on boot. It is no longer necessary to modify the `/etc/init.d/boot.evms` script to delete the device nodes on system restart as was required for previous versions of SUSE Linux.

5 Continue with Section 2.2.3, "Edit the /etc/fstab File" (page 22).

2.2.3 Edit the /etc/fstab File

When you boot the system, the kernel reads the `/etc/fstab` file to identify which file systems should be mounted and then mounts them. This file contains a table of file system information about the `/boot`, `swap`, and root (`/`) partitions plus other partitions and file systems you want to mount.

You must edit the `/etc/fstab` file to modify the mount location of these three partitions so they are mounted under the `/dev/evms` directory. For example, change `/dev/`*`sda1`* to `/dev/evms/`*`sda1`*.

Although the `/boot` partition is not managed by EVMS, the `boot.evms` script forces EVMS to scan all the partitions at boot time, including the `/boot` partition, and it activates `/boot` under the `/dev/evms` directory. Therefore, this makes `/boot` a partition that is discovered by EVMS at startup, and requires that the device's path be listed under `/dev/evms` in the `fstab` file so it can be found when booting with `boot.evms`.

Make sure to replace *`sda1`*, *`sda2`*, and *`sda3`* with the device names you used for your partitions.

IMPORTANT

When working in the `/etc/fstab` file, do not leave any stray characters or spaces in the file. This is a configuration file, and it is highly sensitive to such mistakes.

1 Open the `/etc/fstab` file in a text editor.

2 Locate the line that contains the `/boot` partition.

For example, if your `/boot` partition uses device sda1 and the Reiser file system, look for a line similar to this:

```
/dev/sda1 /boot reiser defaults 1 1
```

3 In the *Device Name* column, modify the mount location of the `/boot` partition from `/dev` to `/dev/evms` so it can be managed by EVMS. Modify only the device name by adding `/evms` to the path:

```
/dev/evms/sda1 /boot reiser defaults 1 1
```

4 Repeat Step 2 (page 23) and Step 3 (page 23) to edit the Device Name entry in the lines for the `swap` and root (`/`) partitions.

For example, change `/dev/sda2` to `/dev/evms/sda2`, and change `/dev/sda3` to `/dev/evms/sda3`.

5 Save the file.

The changes do not take effect until the server is restarted. Do not restart at this time.

6 Continue with Section 2.2.4, “Edit the Boot Loader File” (page 23).

2.2.4 Edit the Boot Loader File

When you boot the system, the kernel reads the boot loader file for information about your system. For Grub, this is the `/boot/grub/menu.1st` file. For LILO, this is the `/etc/lilo.conf` file.

You must edit the boot loader file to modify the mount location of partitions so they are mounted under the `/dev/evms` directory. For example, change /dev/`sda1` to /dev/evms/`sda1`. Make sure to replace the path for all lines that contain device paths in the files. You can modify the boot loader file by editing fields in YaST, or use a text editor to modify the file directly.

IMPORTANT

When working in the boot loader file, do not leave any stray characters or spaces in the file. This is a configuration file, and it is highly sensitive to such mistakes.

Using YaST

To modify the boot loader file in the YaST Control Center:

1 Log in as the `root` user or equivalent.

2 In Yast, select *System* > *Boot Loader*.

3 Modify the boot loader image so that the root file system is mounted as `/dev/evms/` instead of `/dev/`.

3a Select the boot loader image file, then click *Edit*.

3b Edit the device path in the *Root Device* field.

For example, change the *Root Device* value from

`/dev/sda2`

to

`/dev/evms/sda2`

Replace `sda2` with the actual device on your machine.

3c Edit any device paths in the *Other Kernel Parameters* field.

3d Click *OK* to save the changes and return to the Boot Loader page.

4 Modify the failsafe image so that the failsafe root file system is mounted as `/dev/evms/` instead of `/dev/`.

4a Select the failsafe image file, then click *Edit*.

4b Edit the device path in the *Root Device* field.

4c Check the *Other Kernel Parameters* field and make changes if needed.

4d Click *OK* to save the change and return to the Boot Loader page.

5 Click *Finish*.

6 Continue with Section 2.2.5, “Force the RAM Disk to Recognize the Root Partition” (page 26).

Using a Text Editor

To edit the boot loader file in a text editor:

1 Log in as the `root` user or equivalent.

2 Open the boot loader file in a text editor.

For Grub, this is the `/boot/grub/menu.1st` file. For LILO, this is the `/etc/lilo.conf` file.

3 Locate the line that contains the `root=` parameter.

For example, if your root file system uses device sda1, look for a line similar to this:

```
kernel (sd0,0)/vmlinuz root=/dev/sda1 vga=0x31a splash=silent showopts
```

4 Modify the mount location from `/dev` to `/dev/evms` so it can be managed by EVMS.

For example, after the change, the line looks like this:

```
kernel (sd0,0)/vmlinuz root=/dev/evms/sda1 vga=0x31a splash=silent showopts
```

5 Repeat Step 3 (page 25) and Step 4 (page 25) to locate other lines in the file that need to be similarly modified.

6 Save the file.

The changes do not take effect until the server is restarted. Do not restart at this time.

7 Continue with Section 2.2.5, “Force the RAM Disk to Recognize the Root Partition” (page 26).

2.2.5 Force the RAM Disk to Recognize the Root Partition

The `mkinitrd(8)` command creates file system images for use as initial RAM disk (initrd) images. These RAM disk images are often used to preload the block device modules (SCSI or RAID) needed to access the root file system.You might need to force the RAM to update its device node information so that it loads the root (`/`) partition from the `/dev/evms` path.

NOTE

Recent patches to `mkinitrd` might resolve the need to do this task. For the latest version of mkinitrd, see *Recommended Updates for mkinitrd* [http://support.novell.com/techcenter/psdb/24c7dfbc3e0c183970b70c1c0b3a6d7d.html] at the Novell Technical Support Center.

1 At a terminal console prompt, enter the EVMS Ncurses command as the `root` user or equivalent:

```
evmsn
```

2 Review the output to verify that EVMS shows only the `/boot` and `swap` partitions as active in EVMS.

You should see the following devices mounted (with your own partition names, of course) for these two partitions:

```
/dev/evms/sda1
```

```
/dev/evms/sda2
```

3 At a terminal console prompt, enter the following to update the `initrd` image with the `/dev/evms` path information for the root (`/`) partition:

```
/sbin/mkinitrd -f evms
```

This does not take effect until you restart the server.

4 Continue with Section 2.2.6, “Restart the Server” (page 27).

2.2.6 Restart the Server

1 Restart the server to apply the post-install configuration settings.

When your system restarts, the kernel loads the `init-ramdisk`, which runs the EVMS tools to activate your volumes and mount your root file system. Then your boot scripts run the EVMS tools once more to make sure your `/dev/evms/` directory correctly reflects the current state of your volumes. Finally, the remaining EVMS volumes are mounted as specified in your `/etc/fstab` file. Everything else on your system should start up as you would normally expect.

2 Continue with Section 2.2.7, “Verify that EVMS Manages the Boot, Swap, and Root Partitions” (page 27).

2.2.7 Verify that EVMS Manages the Boot, Swap, and Root Partitions

1 At a terminal prompt, enter the EVMS Ncurses command as the `root` user or equivalent:

```
evmsn
```

2 Review the output to verify that EVMS shows the `/boot`, `swap`, and root (`/`) partitions as active in EVMS.

You should see the following devices mounted (with your own partition names, of course) under the `/dev/evms` directory:

```
/dev/evms/sda1

/dev/evms/sda2

/dev/evms/sda3
```

2.3 Configuring LVM Devices to Use EVMS

Use the following post-installation procedure to configure data devices (not system devices) to be managed by EVMS. If you need to configure an existing system device for EVMS, see Section 2.2, “Configuring an Existing System Device to Use EVMS ” (page 20).

1 In a terminal console, run the EVMSGUI by entering the following as the `root` user or equivalent:

```
evmsgui
```

2 In the *Volumes* panel, review the names that EVMS reports as compatibility volumes, find the devices that represent the devices you want to manage using EVMS, then write down the names for future reference.

For example, `/dev/sdb1`.

3 In a text editor, edit the `/etc/fstab` file to use the EVMS volume names.

For example, change the following entry for an LVM2 volume from this

```
/dev/sdb1 / reiserfs defaults 1 2
```

to this

```
/dev/evms/lvm2/sdb1 / reiserfs defaults 1 2
```

IMPORTANT

Make sure not to leave any stray characters or spaces in the line.

With these changes, each time your system boots, your file system is mounted using EVMS as the volume manager.

4 Update the boot scripts as follows:

- The command `evms_activate` must be run from your boot scripts in order to activate your volumes so they can be mounted.

- If you run software-RAID (`boot.md`) or LVM (`boot.lvm`) boot files in your boot scripts, and if you are moving all devices to EVMS, remove or disable those commands.

5 If you have not already done so, enable the `boot.evms` service.

For information, see Section “Enable the boot.evms Service” (page 19).

6 Restart your system.

2.4 Using EVMS with iSCSI Volumes

If your EVMS devices, RAIDs, and volumes use storage devices from an iSCSI SAN, make sure that your system starts iSCSI before EVMS so that the SAN and its disks are available to EVMS on system startup. iSCSI must be started and running before any disks or volumes on the iSCSI SAN can be accessed. If EVMS starts before iSCSI, EVMS cannot see or access the devices in the iSCSI SAN to mount the storage objects they contain, so the EVMS devices, RAIDs, and volumes might not be visible or accessible.

If EVMS starts before iSCSI on your system so that your EVMS devices, RAIDs, and volumes are not visible or accessible, you must correct the order in which iSCSI and EVMS are started. Enter the `chkconfig` command at the Linux server console of every server that is part of your iSCSI SAN.

- At a terminal console prompt, enter either

```
chkconfig evms on
```

or

```
chkconfig boot.evms on
```

This ensures that EVMS and iSCSI start in the proper order each time your servers restart.

2.5 Using the ELILO Loader Files (IA-64)

On a SUSE Linux Enterprise Server boot device EFI System Partition, the full paths to the loader and configuration files are:

```
/boot/efi/SuSE/elilo.efi
```

```
/boot/efi/SuSE/elilo.conf
```

When configuring partitioning during the install on IA64 systems, set the file system type for the `/boot` partition to `vfat`, then choose *Fstab Options* and set the *Arbitrary* option value to `umask=077` to ensure that the partition is accessible only to administrators.

WARNING

Whenever you manually alter the kernel or `initrd` on your system, make sure to run `/sbin/elilo` before shutting down the computer. If you leave out this step, your system might not be bootable.

2.6 Starting EVMS

If EVMS does not start during the system boot, you must activate it manually.

1 Open a terminal console, then log in as the `root` user or equivalent.

2 At the terminal console prompt, enter

```
evms_activate
```

2.7 Starting the EVMS Management Tools

Use the following procedure to start the EVMS management tools.

IMPORTANT

When you are done, make sure to exit the EVMS UI tool. When it is running, the EVMS UI tool locks the EVMS engine, potentially blocking other EVMS actions from taking place.

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Enter one of the following commands to open the desired EVMS UI:

Command	Description
`evmsgui`	Starts the graphical interface for EVMS GUI. For information about features in this interface, see ”EVMS GUI” [http://evms.sourceforge.net/user_guide/#GUI] in the *EVMS User Guide* at the EVMS project on SourceForge.net.
`evmsn`	Starts the text-mode interface for EVMS Ncurses. For information about features in this interface, see the “EVMS Ncurses Interface” [http://evms.sourceforge.net/user_guide/#NCURSES] in the *EVMS User Guide* at the EVMS project on SourceForge.net.
`evms`	Starts the EVMS commandline interpreter (CLI) interface. For information about command options, see “EVMS Command Line Interpreter” [http://evms.sourceforge.net/user_guide/#COMMANDLINE] in the *EVMS User Guide* at the EVMS project on SourceForge.net.

To stop `evmsgui` from running automatically on restart:

1 Close `evmsgui`.

2 Do a clean shutdown (not a restart).

3 Start the server.

When the server comes back up, `evmsgui` is not automatically loaded on restart.

Using UUIDs to Mount Devices 3

This section describes the optional use of UUIDs instead of device names to identify file system devices in the boot loader file and the /etc/fstab file.

3.1 Naming Devices with udev

In the Linux 2.6 and later kernel, udev provides a userspace solution for the dynamic /dev directory, with persistent device naming. As part of the hotplug system, udev is executed if a device is added or removed from the system.

A list of rules is used to match against specific device attributes. The udev rules infrastructure (defined in the /etc/udev/rules.d directory) provides stable names for all disk devices, regardless of their order of recognition or the connection used for the device. The udev tools examine every appropriate block device that the kernel creates to apply naming rules based on certain buses, drive types, or file systems. For information about how to define your own rules for udev, see *Writing udev Rules* [http://reactivated.net/writing_udev_rules.html].

Along with the dynamic kernel-provided device node name, udev maintains classes of persistent symbolic links pointing to the device in the /dev/disk directory, which is further categorized by the by-id, by-label, by-path, and by-uuid subdirectories.

NOTE

Other programs besides udev, such as LVM or md, might also generate UUIDs, but they are not listed in /dev/disk.

3.2 Understanding UUIDs

A UUID (Universally Unique Identifier) is a 128-bit number for a file system that is unique on both the local system and across other systems. It is a randomly generated with system hardware information and time stamps as part of its seed. UUIDs are commonly used to uniquely tag devices.

- Section 3.2.1, “Using UUIDs to Assemble or Activate File System Devices” (page 34)

- Section 3.2.2, “Finding the UUID for a File System Device” (page 35)

3.2.1 Using UUIDs to Assemble or Activate File System Devices

The UUID is always unique to the partition and does not depend on the order in which it appears or where it is mounted. With certain SAN devices attached to the server, the system partitions are renamed and moved to be the last device. For example, if root (/) is assigned to /dev/sda1 during the install, it might be assigned to /dev/sdg1 after the SAN is connected. One way to avoid this problem is to use the UUID in the boot loader and /etc/fstab files for the boot device.

The device ID assigned by the manufactuer for a drive never changes, no matter where the device is mounted, so it can always be found at boot. The UUID is a property of the filesystem and can change if you reformat the drive. In a boot loader file, you typically specify the location of the device (such as `/dev/sda1` or `/dev/evms/sda1`) to mount it at system boot. The boot loader can also mount devices by their UUIDs and administrator-specified volume labels. However, if you use a label and file location, you cannot change the label name when the partition is mounted.

You can use the UUID as criterion for assembling and activating software RAID devices. When a RAID is created, the `md` driver generates a UUID for the device, and stores the value in the `md` superblock.

3.2.2 Finding the UUID for a File System Device

You can find the UUID for any block device in the `/dev/disk/by-uuid` directory. For example, a UUID looks like this:

```
e014e482-1c2d-4d09-84ec-61b3aefde77a
```

3.3 Using UUIDs in the Boot Loader and /etc/fstab File (x86)

After the install, you can optionally use the following procedure to configure the UUID for the system device in the boot loader and `/etc/fstab` files for your x86 system.

1 Install the SUSE Linux Enterprise Server for x86 with no SAN devices connected.

2 After the install, boot the system.

3 Open a terminal console as the `root` user or equivalent.

4 Navigate to the `/dev/disk/by-uuid` directory to find the UUID for the device where you installed `/boot`, `/root`, and `swap`.

  4a At the terminal console prompt, enter

```
cd /dev/disk/by-uuid
```

4b List all partitions by entering

```
ll
```

4c Find the UUID, such as

```
e014e482-1c2d-4d09-84ec-61b3aefde77a —> /dev/sda1
```

5 Edit `/boot/grub/menu.1st` file, using the Boot Loader option in YaST2 or using a text editor.

For example, change

```
kernel /boot/vmlinuz root=/dev/sda1
```

to

```
kernel /boot/vmlinuz
root=/dev/disk/by-uuid/e014e482-1c2d-4d09-84ec-61b3aefde77a
```

IMPORTANT

Make a copy of the original boot entry, then modify the copy. If you make a mistake, you can boot the server without the SAN connected, and fix the error.

If you use the Boot Loader option in YaST, there is a defect where it adds some duplicate lines to the boot loader file when you change a value. Use an editor to remove the following duplicate lines:

```
color white/blue black/light-gray

default 0

timeout 8

gfxmenu (sd0,1)/boot/message
```

When you use YaST to change the way that the root (/) device is mounted (such as by UUID or by label), the boot loader configuration needs to be saved again to make the change effective for the boot loader.

6 As the `root` user or equivalent, do one of the following to place the UUID in the `/etc/fstab` file:

- Open YaST to *System* > *Partitioner*, select the device of interest, then modify *Fstab Options*.
- Edit the `/etc/fstab` file to modify the system device from the location to the UUID.

  For example, if the root (/) volume has a device path of `/dev/sda1` and its UUID is `e014e482-1c2d-4d09-84ec-61b3aefde77a`, change line entry from

  ```
  /dev/sda1   /            reiserfs   acl,user_xattr        1 1
  ```

  to

  ```
  UUID=e014e482-1c2d-4d09-84ec-61b3aefde77a   /  reiserfs   acl,user_xattr
          1 1
  ```

  IMPORTANT

  Make sure to make a backup copy of the `fstab` file before you begin, and do not leave stray characters or spaces in the file.

3.4 Using UUIDs in the Boot Loader and /etc/fstab File (IA64)

After the install, use the following procedure to configure the UUID for the system device in the boot loader and `/etc/fstab` files for your IA64 system. IA64 uses the EFI BIOS. Its file system configuration file is `/boot/efi/SuSE/elilo.conf` instead of `/etc/fstab`.

1 Install the SUSE Linux Enterprise Server for IA64 with no SAN devices connected.

2 After the install, boot the system.

3 Open a terminal console as the `root` user or equivalent.

4 Navigate to the `/dev/disk/by-uuid` directory to find the UUID for the device where you installed `/boot`, `/root`, and `swap`.

4a At the terminal console prompt, enter

```
cd /dev/disk/by-uuid
```

4b List all partitions by entering

```
ll
```

4c Find the UUID, such as

```
e014e482-1c2d-4d09-84ec-61b3aefde77a —> /dev/sda1
```

5 Edit the boot loader file, using the Boot Loader option in YaST2.

For example, change

```
root=/dev/sda1
```

to

```
root=/dev/disk/by-uuid/e014e482-1c2d-4d09-84ec-61b3aefde77a
```

6 Edit the `/boot/efi/SuSE/elilo.conf` file to modify the system device from the location to the UUID.

For example, change

```
/dev/sda1   /   reiserfs   acl,user_xattr        1 1
```

to

```
UUID=e014e482-1c2d-4d09-84ec-61b3aefde77a   /  reiserfs   acl,user_xattr
      1 1
```

IMPORTANT

Make sure to make a backup copy of the `/boot/efi/SuSE/elilo.conf` file before you begin, and do not leave stray characters or spaces in the file.

3.5 Additional Information

For more information about using udev(8) for managing devices, see “Dynamic Kernel Device Management with udev” [http://www.novell.com/documentation/sles10/sles_admin/data/cha_udev.html] in the *SUSE Linux Enterprise Server 10 Installation and Administration Guide*.

For more information about udev(8) commands, see its man page. Enter the following at a terminal console prompt:

```
man 8 udev
```

4 Managing EVMS Devices

This section describes how to initialize a disk for EVMS management by adding a segment management container to manage the partitions that you later add to the disk.

4.1 Understanding Disk Segmentation

In EVMS, you initialize a disk by assigning a segment manager to it. The segment manager creates metadata for the disk and exposes its free space so you can subdivide it into one or multiple segments (also called partitions).

4.1.1 Segment Managers

The most commonly used segment manager is the DOS Segment Manager. The following table describes the segment managers available in EVMS.

Table 4.1 *EVMS Segment Managers*

Segment Manager	Description
DOS	The standard MS-DOS disk partitioning scheme. It is the most commonly used partitioning scheme for Linux, NetWare, Windows, OS/2, BSD, Solaris X86, and UnixWare.
GPT (Globally Unique Identifier (GUID) Partitioning Table)	A partitioning scheme used for IA-64 platforms, as defined in the Intel *Extensible Firmware Interface (EIF) Specification*. It is not compatible with DOS, Windows, or OS/2 systems. The GUID is also known as Universally Unique Identifier (UUID). The GPT combines time and space descriptors to create this unique 128-bit tag for the disk and its segments.
S/390	A partitioning scheme used exclusively for System/390 mainframes.
Cluster	A partitioning scheme for high-availability clusters. It provides a GUID for the disk, creates an EVMS container for the shared cluster devices, and specifies a node ID for the node that owns the device and the cluster ID.
BSD	A partitioning scheme for BSD UNIX.
MAC	A partitioning scheme for Mac-OS partitions.

4.1.2 Disk Segments

After you initialize the disk by adding a segment manager, you see metadata and free space segments on the disk. You can then create one or multiple data segments in a disk segment.

Table 4.2 *Disk Segment Types*

Segment Type	Description
Metadata	A set of contiguous sectors that contain information needed by the segment manager.
Free Space	A set of contiguous sectors that are unallocated or not in use. Free space can be used to create a segment.
Data	A set of contiguous sectors that has been allocated from a disk. The segment might be in use for a volume or a software RAID.

4.2 Initializing Disks

You must initialize new disks and disks that you want to reformat. After the disk is initialized, you can subdivide, or carve, the device into one or more disk segments for your file systems.

- Section 4.2.1, “Before You Begin” (page 43)
- Section 4.2.2, “Guidelines” (page 44)
- Section 4.2.3, “Adding a Segment Manager” (page 45)

4.2.1 Before You Begin

If you use large disks or disk arrays, use the vendor’s tools to carve them into the sizes that are usable for the management tools you plan to use. For example, the `md` driver recognizes disks only up to 2 TB in size, so the limit also applies to the `md` plug-in for EVMS. Software RAID devices you create with EVMS can be larger than 2 TB, of course, because the md driver plug-in manages the disks underneath that storage structure.

When you boot the server, EVMS scans and recognizes all devices it manages. If you add a new device to the server or create a device using mkfs, EVMS automatically mounts it on reboot under /dev/evms as a compatibility volume, such as /dev/evms/sdb.

IMPORTANT

If you cannot find a new disk, device, or volume, look under /dev/evms in a file browser, or look for compatibility volumes in the Volumes Manager in the EVMS GUI (evmsgui).

4.2.2 Guidelines

Consider the following guidelines when initializing a disk:

- EVMS might allow you to create segments without first adding a segment manager for the disk, but it is best to explicitly add a segment manager to avoid problems later.

  IMPORTANT

  You must add a Cluster segment manager if you plan to use the devices for volumes that you want to share as cluster resources.

- When you initialize an existing disk that is already formatted, the process of adding a segment manager destroys all data on the disk. If you want to keep the data on the disk, make sure to back up the data before you begin this process.

- For existing disks on the system or disks that you move from another system, you must delete any existing volume management structures, and remove any segment managers. This removes the device’s metadata and data, and destroys all data on the disk.

WARNING

Do not initialize the device that contains your current system disk or any device that contains the `/boot`, `swap`, or root (/) volumes.

- If a new disk does not show up in the list of *Available Objects*, look for it in the *Volumes* list to see if the disk shows up as a compatibility volume. For example, a new disk `sdb` would show up as `/dev/evms/sdb`. Delete it from the Volumes list to force the disk to show up in *Available Objects*, then create segments as desired.

4.2.3 Adding a Segment Manager

Use the following procedure to assign a segment manager to device for servers using x86, x64, and IA64 controllers. This option is not available for S390 platforms, so simply continue with configuring software RAIDs or file system partitions, as desired.

WARNING

Adding a segment manager initializes the disk, completely removing all the segments it contains. All the data stored on the device is lost.

1 If the disk has any existing volume management structures or an existing segment manager, remove them.

   1a Select *Actions* > *Delete* > *Volume* to view the Volumes list.

   1b Select any existing volume management structures on the device, then click *Delete*.

   1c Select *Actions* > *Remove* > *Segment Manager from Storage Object.*

   1d Select the type of Segment Manager in use, then click *Next*.

   1e Select the device, then click Remove.

2 If the disk is a new one that is listed as a compatibility volume in the Volumes list, delete it as a compatibility volume.

2a Select *Actions* > *Delete* > *Volume* to view the Volumes list.

2b Select the device, then click *Delete*.

3 Add the Segment Manager.

3a In the list of *Availability Objects*, select the device, then click *Actions* > *Add* > *Segment Manager to Storage Object*.

3b From the list, select one of the following types of segment manager, then click *Next*.

- *DOS Segment Manager* (the most common choice)
- *GPT Segment Manager* (for IA-64 platforms)
- *Cluster Segment Manager* (available only if it is a viable option for the selected disk)

3c Select the device from the list of *Plugin Acceptable Objects*, then click *Next*.

3d If required, specify the disk type as Linux.

3e Click *Add* to create the segment management container for the disk, then click *OK* to dismiss the confirmation message.

4.3 Removing the Segment Manager from a Device

1 If the disk has any existing volume management structures, remove them.

1a Select *Actions* > *Delete* > *Volume* to view the Volumes list.

1b Select any existing volume management structures on the device, then click *Delete*.

2 Select *Actions* > *Remove* > *Segment Manager from Storage Object*.

3 Select the type of Segment Manager in use, then click *Next*.

4 Select the device, then click *Remove*.

4.4 Creating Disk Segments (or Partitions)

1 In EVMS, select *Actions* > *Create* > *Segment* to see a list of segment managers.

2 From the list, select the segment manager for the device you want to manage, then click *Next*.

- *DOS Segment Manager* (the most common choice)
- *GPT Segment Manager* (for IA-64 platforms)
- *Cluster Segment Manager* (available only if it is a viable option for the selected disk)

For information about these and other segment managers available, see Section 4.1.1, “Segment Managers” (page 42).

3 Select the storage object that you want to segment, then click *Next*.

4 Complete the required configuration options for the segment, and modify default values as desired.

- **Size (MB):** Specify the amount of space (in MB) that you want to use. Use the arrows or type a value. The interface corrects the value to the lower or upper size limit if you specify a size that is too small or that exceeds the amount of free space available.
- **Offset (Sectors):** Specify the number of sectors to skip before beginning this partition if you want to leave free space in front of it.

- **Partition Type:** From the drop-down list, select *Linux* (default), *Linux Swap*, *Linux LVM*, *NTFS*, *HPFS*, *FAT16*, or *Other Partition Type*.
- **Partition Type ID:** This value changes automatically based on the *Partition Type* value, except for the Other Partition Type option, where you must manually enter a value.
- **Bootable:** Click *Yes* to make a primary partition active so that you can boot from it, or click *No* to make it unbootable. *No* is the only option if you are creating a logical partition.
- **Primary Partition:** Click *Yes* for a primary partition, or click *No* for a logical partition.

Required settings are denoted in the page by an asterisk (*). All required fields must be completed to make the *Create* button active.

5 Click *Create* to create the segment.

6 Verify that the new segment appears in the Segment list.

4.5 Configuring Mount Options for Devices

The following table describes the *Fstab Options* that are configurable in YaST. The values are written to the `/etc/fstab` file and are applied upon reboot.

Table 4.3 *Fstab Options in YaST*

Fstab Option	Description
Mount by	• *Device name* (K) (default, such as /dev/sda2) • *Volume label* (L) Make sure to also specify a value in *Volume Label*. • *UUID* (U)

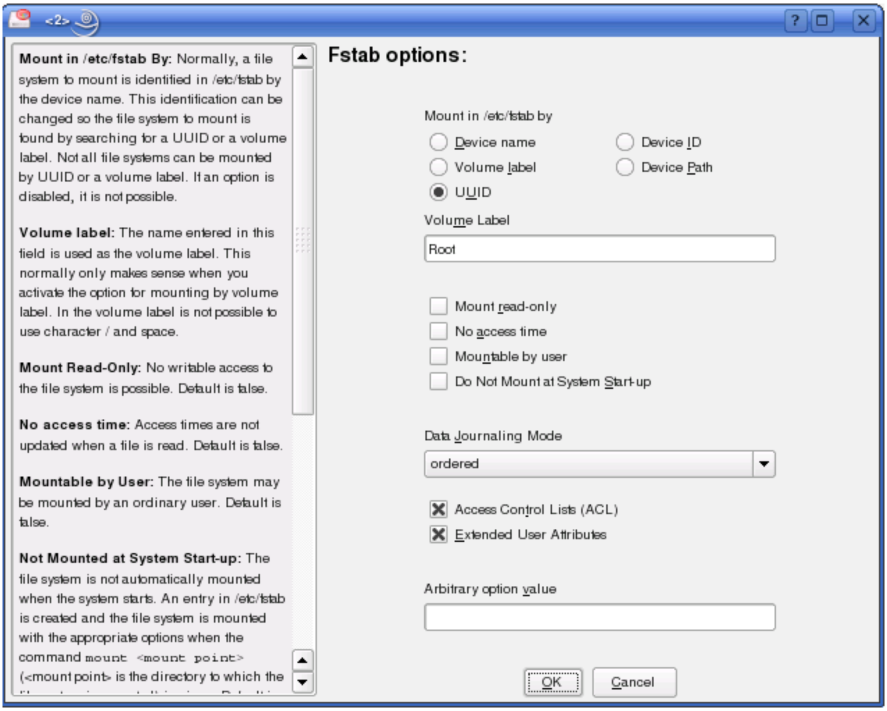
Fstab Option	Description
	For information about why you might want to discover partitions and devices by UUID, see Section 3.2.1, “Using UUIDs to Assemble or Activate File System Devices” (page 34). • *Device ID* (I) • *Device Path* (P)
Volume label	A useful name to help you easily identify the volume on the server. By default, this field is empty.
Mount read-only	Select the check box to enable this option. It is deselected (disabled) by default. If this option is enabled, files and directories cannot be modified or saved on the volume.
No access time	Select the check box to enable this option. It is deselected (disabled) by default. By default, the Linux `open(2)` command updates the access time whenever a file is opened. The *No Access Time* option disables the updating of access time, so that reading a file does not update its access time. Enabling the *No Access Time* option allows you to back up a volume without modifying the access times of its files.
Mountable by user	Select the check box to enable this option. It is deselected (disabled) by default. If this option is enabled, the volume can be mounted by any user; `root` privileges are not required.
Do Not Mount at System Start-up	Select the check box to enable this option. It is deselected (disabled) by default.

Fstab Option	Description
	The system volumes such as /boot, swap, and root (/) should all be mounted at system start. For other volumes, enable this option for a volume if you want to mount it manually later using the mount command at a terminal console prompt.
Data journaling mode	For journaling file systems, select the preferred journaling mode: • **Ordered:** Writes data to the file system, then enters the metadata in the journal. This is the default. • **Journal:** Writes data twice; once to the journal, then to the file system. • **Writeback:** Writes data to the file system and writes metadata in the journal, but the writes are performed in any order.
Access Control LIsts (ACL)	Select this option to enable access control lists on the file system. It is enabled by default.
Extended user attributes	Select this option to enable extended user attributes on the file system. It is enabled by default.
Arbitrary option value	Specify any mount option that is legal for the Mount Options column for a device entry in the /etc/fstab file. Use a comma with no spaces to separate multiple options.

You can modify these values for each entry by editing the /etc/fstab file, or use the following procedure to modify the mount options for a volume in the /etc/fstab file from YaST.

1 Open YaST, then click *System* > *Partitioning*.

2 Select the device you want to modify, then click *Fstab Options*.



Mount in /etc/fstab By: Normally, a file system to mount is identified in /etc/fstab by the device name. This identification can be changed so the file system to mount is found by searching for a UUID or a volume label. Not all file systems can be mounted by UUID or a volume label. If an option is disabled, it is not possible.

Volume label: The name entered in this field is used as the volume label. This normally only makes sense when you activate the option for mounting by volume label. In the volume label is not possible to use character / and space.

Mount Read-Only: No writable access to the file system is possible. Default is false.

No access time: Access times are not updated when a file is read. Default is false.

Mountable by User: The file system may be mounted by an ordinary user. Default is false.

Not Mounted at System Start-up: The file system is not automatically mounted when the system starts. An entry in /etc/fstab is created and the file system is mounted with the appropriate options when the command `mount <mount point>` (<mount point> is the directory to which the

Fstab options:

Mount in /etc/fstab by

○ Device name
○ Device ID
○ Volume label
○ Device Path
● UUID

Volume Label

Root

□ Mount read-only
□ No access time
□ Mountable by user
□ Do Not Mount at System Start-up

Data Journaling Mode

ordered

☒ Access Control Lists (ACL)
☒ Extended User Attributes

Arbitrary option value

3 Modify the settings as desired, then click *OK* to accept your changes.

4.6 What's Next

If multiple paths exist between your host bus adapters (HBAs) and the storage devices, configure multipathing for the devices before creating software RAIDs or file system volumes on the devices. For information, see Chapter 5, *Managing Multipath I/O for Devices* (page 53).

If you want to configure software RAIDs, do it before you create file systems on the devices. For information, see Chapter 6, *Managing Software RAIDs with EVMS* (page 111).

Managing Multipath I/O for Devices

5

This section describes how to manage failover and path load balancing for multiple paths between the servers and block storage devices.

5.1 Understanding Multipathing

5.1.1 What Is Multipathing?

Multipathing is the ability of a server to communicate with the same physical or logical block storage device across multiple physical paths between the host bus adapters in the server and the storage controllers for the device, typically in Fibre Channel (FC) or iSCSI SAN environments. You can also achieve multiple connections with direct attached storage when multiple channels are available.

5.1.2 Benefits of Multipathing

Linux multipathing provides connection fault tolerance and can provide load balancing across the active connections. When multipathing is configured and running, it automatically isolates and identifies device connection failures, and reroutes I/O to alternate connections.

Typical connection problems involve faulty adapters, cables, or controllers. When you configure multipath I/O for a device, the multipath driver monitors the active connection between devices. When the multipath driver detects I/O errors for an active path, it fails over the traffic to the device’s designated secondary path. When the preferred path becomes healthy again, control can be returned to the preferred path.

5.2 Planning for Multipathing

5.2.1 Guidelines for Multipathing

Use the guidelines in this section when planning your multipath I/O solution.

Prerequisites

- Multipathing is managed at the device level.

- The storage array you use for the multipathed device must support multipathing. For more information, see Section 5.2.9, “Supported Storage Arrays for Multipathing” (page 61).

- You need to configure multipathing only if multiple physical paths exist between host bus adapters in the server and host bus controllers for the block storage device. You configure multipath for the logical device as seen by the server.

Vendor-Provided Multipath Solutions

For some storage arrays, the vendor will provide its own multipathing software to manage multipathing for the array’s physical and logical devices. In this case, you should follow the vendor’s instructions for configuring multipathing for those devices.

Disk Management Tasks

Perform the following disk management tasks before you attempt to configure multipathing for a physical or logical device that has multiple paths:

- Use third-party tools to carve physical disks into smaller logical disks.

- Use third-party tools to partition physical or logical disks. If you change the partitioning in the running system, the Device Mapper Multipath (DM-MP) module does not automatically detect and reflect these changes. DM-MP must be reinitialized, which usually requires a reboot.

- Use third-party SAN array management tools to create and configure hardware RAID devices.

- Use third-party SAN array management tools to create logical devices such as LUNs. Logical device types that are supported for a given array depend on the array vendor.

Software RAIDs

The Linux software RAID management software runs on top of multipathing. For each device that has multiple I/O paths and that you plan to use in a software RAID, you must configure the device for multipathing before you attempt to create the software RAID device. Automatic discovery of multipathed devices is not available. The software RAID is not aware of the multipathing management running underneath.

High-Availability Solutions

High-availability solutions for clustering typically run on top of the multipathing server. For example, the Distributed Replicated Block Device (DRBD) high-availability solution for mirroring devices across a LAN runs on top of multipathing. For each device that has multiple I/O paths and that you plan to use in a DRDB solution, you must configure the device for multipathing before you configure DRBD.

Volume Managers

Volume managers such as LVM2 and EVMS run on top of multipathing. You must configure multipathing for a device before you use LVM2 or EVMS to create segment managers and file systems on it.

Virtualization Environments

When using multipathing in a virtualization environment, the multipathing is controlled in the host server environment. Configure multipathing for the device before you assign it to a virtual guest machine.

5.2.2 Using Multipathed Devices Directly or in EVMS

If you want to use the entire LUN directly (for example, if you are using the SAN features to partition your storage), you can simply use the `/dev/disk/by-id/xxx` names directly for `mkfs`, `fstab`, your application, etc.

If the `user_friendly_names` option or `alias` option is enabled in the `/etc/multipath.conf` file, you can optionally use the `/dev/mapper/mpathn` device name because this name is aliased to the devices ID. Some limitations apply; for information, see Section “Configuring User-Friendly Names or Alias Names in /etc/multipath.conf” (page 79).

5.2.3 Using LVM2 on Multipath Devices

By default, LVM2 does not recognize multipathed devices. To make LVM2 recognize the multipathed devices as possible physical volumes, you must modify /etc/lvm/lvm.conf. It is important to modify it in a way that it does not scan and use the physical paths, but only accesses the multipath I/O storage through the multipath I/O layer.

To modify/etc/lvm/lvm.conf for multipath use:

1 Open the /etc/lvm/lvm.conf file in a text editor.

2 Change the filter and types entry in /etc/lvm/lvm.conf as follows:

```
filter = [ "a|/dev/disk/by-id/.*|", "r|.*|" ]
```

This allows LVM2 to scan only the by-id paths and reject everything else.

3 If you are also using LVM2 on non-multipathed devices, make the necessary adjustments in the filter and types entries to suit your setup. Otherwise, the other LVM devices are not visible with a pvscan after you modify the lvm.conf file for multipathing.

You want only those devices that are configured with LVM to be included in the LVM cache, so make sure you are specific about which other non-multipathed devices are included by the filter.

For example, if your local disk is /dev/sda and all SAN devices are /dev/sdb and above, specify the local and multipathing paths in the filter as follows:

```
filter = [ "a|/dev/sda.*|", "a|/dev/disk/by-id/.*|", "r|.*|" ]
types = [ "device-mapper", 253 ]
```

4 Save the file.

5 Add dm-multipath to /etc/sysconfig/kernel:INITRD_MODULES.

6 Make a new initrd to ensure that the Device Mapper Multipath services are loaded with the changed settings. Enter

```
mkinitrd -f mpath
```

7 Reboot the server to apply the changes.

5.2.4 Using mdadm with Multipath Devices

The `mdadm` tool requires that the devices be accessed by the ID rather than by the device node path. Therefore, the `DEVICE` entry in `/etc/mdadm.conf` should be set as follows:

```
DEVICE /dev/disk/by-id/*
```

This is the default handling in SUSE Linux Enterprise Server 10 and later.

5.2.5 Using --noflush with Multipath Devices

The option --noflush should always be used when running on multipath devices.

For example, in scripts where you perform a table reload, you use the --noflush option on resume to ensure that any outstanding I/O is not flushed, because you need the multipath topology information.

```
load
resume --noflush
```

5.2.6 SAN Timeout Settings When the Root Device Is Multipathed

A system with root (`/`) on a multipath device might stall when all paths have failed and are removed from the system because a `dev_loss_tmo` time-out is received from the storage subsystem (such as Fibre Channel storage arrays).

If the system device is configured with multiple paths and the multipath `no_path_retry` setting is active, you should modify the storage subsystem's `dev_loss_tmo` setting accordingly to ensure that no devices are removed during an all-paths-down scenario. We strongly recommend that you set the `dev_loss_tmo` value to be equal to or higher than the `no_path_retry` setting from multipath.

The recommended setting for the storage subsystem's `dev_los_tmo` is:

```
<dev_loss_tmo> = <no_path_retry> * <polling_interval>
```

where the following definitions apply for the multipath values:

- `no_path_retry` is the number of retries for multipath I/O until the path is considered to be lost, and queuing of IO is stopped.

- `polling_interval` is the time in seconds between path checks.

Each of these multipath values should be set from the `/etc/multipath.conf` configuration file. For information, see Section 5.4.5, “Creating and Configuring the /etc/multipath.conf File” (page 76).

5.2.7 Partitioning Multipath Devices

SUSE Linux Enterprise Server 10

In SUSE Linux Enterprise Server 10, the `kpartx` software is used in the `/etc/init.d/boot.multipath` to add symlinks to the `/dev/dm-*` line in the `multipath.conf` configuration file for any newly created partitions without requiring a reboot. This triggers `udevd` to fill in the `/dev/disk/by-*` symlinks. The main benefit is that you can call `kpartx` with the new parameters without having to reboot the server.

SUSE Linux Enterprise Server 9

In SUSE Linux Enterprise Server 9, it is not possible to partition multipath I/O devices themselves. If the underlying physical device is already partitioned, the multipath I/O device reflects those partitions and the layer provides `/dev/disk/by-id/<name>p1 ... pN` devices so you can access the partitions through the multipath I/O layer. As a consequence, the devices need to be partitioned prior to enabling multipath I/O. If you change the partitioning in the running system, DM-MP does not automatically detect and reflect these changes. The device must be reinitialized, which usually requires a reboot.

5.2.8 Supported Architectures for Multipath I/O

The multipathing drivers and tools in SUSE Linux Enterprise Server 10 support all seven of the supported processor architectures: IA32, AMD64/EM64T, IPF/IA64, p-Series (32-bit/64-bit), z-Series (31-bit and 64-bit).

5.2.9 Supported Storage Arrays for Multipathing

The multipathing drivers and tools in SUSE Linux Enterprise Server 10 support most storage arrays. The storage array that houses the multipathed device must support multipathing in order to use the multipathing drivers and tools. Some storage array vendors provide their own multipathing management tools. Consult the vendor's hardware documentation to determine what settings are required.

- Section "Storage ArraysThat Are Automatically Detected for Multipathing" (page 61)
- Section "Tested Storage Arrays for Multipathing Support" (page 63)
- Section "Storage Arrays that Require Specific Hardware Handlers" (page 63)

Storage ArraysThat Are Automatically Detected for Multipathing

The `multipath-tools` package automatically detects the following storage arrays:

3PARdata VV
Compaq HSV110
Compaq MSA1000
DDN SAN MultiDirector
DEC HSG80
EMC CLARiiON CX
EMC Symmetrix
FSC CentricStor

Hewlett Packard (HP) A6189A
HP HSV110
HP HSV210
HP Open-
Hitachi DF400
Hitachi DF500
Hitachi DF600
IBM 3542
IBM ProFibre 4000R
NetApp
SGI TP9100
SGI TP9300
SGI TP9400
SGI TP9500
STK OPENstorage DS280
Sun StorEdge 3510
Sun T4

In general, most other storage arrays should work. When storage arrays are automatically detected, the default settings for multipathing apply. If you want non-default settings, you must manually create and configure the `/etc/multipath.conf` file. For information, see Section 5.4.5, "Creating and Configuring the /etc/multipath.conf File" (page 76).

Hardware that is not automatically detected requires an appropriate entry for configuration in the `DEVICES` section of the `/etc/multipath.conf` file. In this case, you must manually create and configure the configuration file. For information, see Section 5.4.5, "Creating and Configuring the /etc/multipath.conf File" (page 76).

Consider the following caveats:

- Not all of the storage arrays that are automatically detected have been tested on SUSE Linux Enterprise Server. For information, see Section "Tested Storage Arrays for Multipathing Support" (page 63).

- Some storage arrays might require specific hardware handlers. A hardware handler is a kernel module that performs hardware-specific actions when switching path groups and dealing with I/O errors. For information, see Section "Storage Arrays that Require Specific Hardware Handlers" (page 63).

- After you modify the `/etc/multipath.conf` file, you must run `mkinitrd` to re-create the INITRD on your system, then reboot in order for the changes to take effect.

Tested Storage Arrays for Multipathing Support

The following storage arrays have been tested with SUSE Linux Enterprise Server:

EMC
Hitachi
Hewlett-Packard/Compaq
IBM
NetApp
SGI

Most other vendors' storage arrays should also work. Consult your vendor's documentation for guidance. For a list of the default storage arrays recognized by the `multipath-tools` package, see Section "Storage ArraysThat Are Automatically Detected for Multipathing" (page 61).

Storage Arrays that Require Specific Hardware Handlers

Storage arrays that require special commands on failover from one path to the other or that require special nonstandard error handling might require more extensive support. Therefore, the Device Mapper Multipath service has hooks for hardware handlers. For example, one such handler for the EMC CLARiiON CX family of arrays is already provided.

IMPORTANT

Consult the hardware vendor's documentation to determine if its hardware handler must be installed for Device Mapper Multipath.

The `multipath -t` command shows an internal table of storage arrays that require special handling with specific hardware handlers. The displayed list is not an exhaustive list of supported storage arrays. It lists only those arrays that require special handling and that the `multipath-tools` developers had access to during the tool development.

IMPORTANT

Arrays with true active/active multipath support do not require special handling, so they are not listed for the `multipath -t` command.

A listing in the `multipath -t` table does not necessarily mean that SUSE Linux Enterprise Server was tested on that specific hardware. For a list of tested storage arrays, see Section "Tested Storage Arrays for Multipathing Support" (page 63).

5.3 Multipath Management Tools

The multipathing support in SUSE Linux Enterprise Server 10 is based on the Device Mapper Multipath module of the Linux 2.6 kernel and the `multipath-tools` userspace package. You can use `mdadm` to view the status of multipathed devices.

- Section 5.3.1, "Device Mapper Multipath Module" (page 64)
- Section 5.3.2, "Multipath I/O Management Tools" (page 67)
- Section 5.3.3, "Using mdadm for Multipathed Devices" (page 68)
- Section 5.3.4, "The Linux multipath(8) Command" (page 69)

5.3.1 Device Mapper Multipath Module

The Device Mapper Multipath (DM-MP) module provides the multipathing capability for Linux. DM-MP is the preferred solution for multipathing on SUSE Linux Enterprise Server 10. It is the only multipathing option shipped with the product that is completely supported by Novell and SUSE.

DM-MP features automatic configuration of the multipathing subsystem for a large variety of setups. Configurations of up to 8 paths to each device are supported. Configurations are supported for active/passive (one path active, others passive) or active/active (all paths active with round-robin load balancing).

The DM-MP framework is extensible in two ways:

- Using specific hardware handlers. For information, see Section “Storage Arrays that Require Specific Hardware Handlers” (page 63).

- Using more sophisticated load-balancing algorithms than round-robin

The user-space component of DM-MP takes care of automatic path discovery and grouping, as well as automated path retesting, so that a previously failed path is automatically reinstated when it becomes healthy again. This minimizes the need for administrator attention in a production environment.

DM-MP protects against failures in the paths to the device, and not failures in the device itself. If one of the active paths is lost (for example, a network adapter breaks or a fiber-optic cable is removed), I/O is redirected to the remaining paths. If the configuration is active/passive, then the path fails over to one of the passive paths. If you are using the round-robin load-balancing configuration, the traffic is balanced across the remaining healthy paths. If all active paths fail, inactive secondary paths must be waked up, so failover occurs with a delay of approximately 30 seconds.

If a disk array has more than one storage processor, make sure that the SAN switch has a connection to the storage processor that owns the LUNs you want to access. On most disk arrays, all LUNs belong to both storage processors, so both connections are active.

NOTE

On some disk arrays, the storage array manages the traffic through storage processors so that it presents only one storage processor at a time. One processor is active and the other one is passive until there is a failure. If you are connected to the wrong storage processor (the one with the passive path) you might not see the expected LUNs, or you might see the LUNs but get errors when trying to access them.

Table 5.1 *Multipath I/O Features of Storage Arrays*

Features of Storage Arrays	Description
Active/passive controllers	One controller is active and serves all LUNs. The second controller acts as a standby. The second controller also presents the LUNs to the multipath component so that the operating system knows about redundant paths. If the prima-

Features of Storage Arrays	Description
	ry controller fails, the second controller takes over, and it serves all LUNs. In some arrays, the LUNs can be assigned to different controllers. A given LUN is assigned to one controller to be its active controller. One controller does the disk I/O for any given LUN at a time, and the second controller is the standby for that LUN. The second controller also presents the paths, but disk I/O is not possible. Servers that use that LUN are connected to the LUN's assigned controller. If the primary controller for a set of LUNs fails, the second controller takes over, and it serves all LUNs.
Active/active controllers	Both controllers share the load for all LUNs, and can process disk I/O for any given LUN. If one controller fails, the second controller automatically handles all traffic.
Load balancing	The Device Mapper Multipath driver automatically load balances traffic across all active paths.
Controller failover	When the active controller fails over to the passive, or standby, controller, the Device Mapper Multipath driver automatically activates the paths between the host and the standby, making them the primary paths.
Boot/Root device support	Multipathing is supported for the root (`/`) device in SUSE Linux Enterprise Server 10 and later. The host server must be connected to the currently active controller and storage processor for the boot device. The `/boot` partition must be on a separate, non-multipathed partition. Otherwise, no boot loader is written. For information, see Section 5.7, "Configuring Multipath I/O for the Root Device" (page 98).

Device Mapper Multipath detects every path for a multipathed device as a separate SCSI device. The SCSI device names take the form `/dev/sd`*`N`*, where *`N`* is an autogenerated letter for the device, beginning with a and issued sequentially as the devices are created, such as /dev/sda, /dev/sdb, and so on. If the number of devices exceeds 26, the letters are duplicated so that the next device after /dev/sdz will be named /dev/sdaa, /dev/sdab, and so on.

If multiple paths are not automatically detected, you can configure them manually in the /etc/multipath.conf file. The multipath.conf file does not exist until you create and configure it. For information, see Section 5.4.5, "Creating and Configuring the /etc/multipath.conf File" (page 76).

5.3.2 Multipath I/O Management Tools

The multipath-tools user-space package takes care of automatic path discovery and grouping. It automatically tests the path periodically, so that a previously failed path is automatically reinstated when it becomes healthy again. This minimizes the need for administrator attention in a production environment.

Table 5.2 *Tools in the multipath-tools Package*

Tool	Description
multipath	Scans the system for multipathed devices and assembles them.
multipathd	Waits for maps events, then executes multipath.
devmap-name	Provides a meaningful device name to udev for device maps (devmaps).
kpartx	Maps linear devmaps to partitions on the multipathed device, which makes it possible to create multipath monitoring for partitions on the device.

The file list for a package can vary for different server architectures. For a list of files included in the multipath-tools package, go to the *SUSE Linux Enterprise Server Technical SpecificationsPackage Descriptions* Web page [http://www.novell.com/products/server/techspecs.html], find your architecture and select *Packages Sorted by Name*, then search on "multipath-tools" to find the package list for that architecture.

You can also determine the file list for an RPM file by querying the package itself: using the `rpm -ql` or `rpm -qpl` command options.

- To query an installed package, enter

  ```
  rpm -ql <package_name>
  ```

- To query a package not installed, enter

  ```
  rpm -qpl <URL_or_path_to_package>
  ```

To check that the `multipath-tools` package is installed, do the following:

- Ensure that the `multipath-tools` package is installed by entering the following at a terminal console prompt:

  ```
  rpm -q multipath-tools
  ```

  If it is installed, the response repeats the package name and provides the version information, such as:

  ```
  multipath-tools-04.7-34.23
  ```

  If it is not installed, the response reads:

  ```
  package multipath-tools is not installed
  ```

5.3.3 Using mdadm for Multipathed Devices

In SUSE Linux Enterprise Server 10, Udev is the default device handler, and devices are automatically known to the system by the Worldwide ID instead of by the device node name. This resolves problems in previous releases where `mdadm.conf` and `lvm.conf` did not properly recognize multipathed devices.

Just as for LVM2, `mdadm` requires that the devices be accessed by the ID rather than by the device node path. Therefore, the `DEVICE` entry in `/etc/mdadm.conf` should be set as follows:

```
DEVICE /dev/disk/by-id/*
```

This is the default handling in SUSE Linux Enterprise Server 10 and later, as noted above.

To verify that mdadm is installed:

- Ensure that the mdadm package is installed by entering the following at a terminal console prompt:

  ```
  rpm -q mdadm
  ```

  If it is installed, the response repeats the package name and provides the version information. For example:

  ```
  mdadm-2.6-0.11
  ```

  If it is not installed, the response reads:

  ```
  package mdadm is not installed
  ```

For information about modifying the /etc/lvm/lvm.conffile, see Section 5.2.3, “Using LVM2 on Multipath Devices” (page 58).

5.3.4 The Linux multipath(8) Command

Use the Linux multipath(8) command to configure and manage multipathed devices.

Syntax

General syntax for the multipath(8) command:

```
multipath [-v verbosity] [-d] [-h|-l|-ll|-f|-F] [-p failover | multibus |
group_by_serial | group_by_prio| group_by_node_name ]
```

Options

multipath
: Configure all multipath devices.

multipath *devicename*
: Configures a specific multipath device.

  Replace *devicename* with the device node name such as /dev/sdb (as shown by udev in the $DEVNAME variable), or in the major:minor format.

multipath -f

Selectively suppresses a multipath map, and its device-mapped partitions.

multipath -d

Dry run. Displays potential multipath devices, but does not create any devices and does not update device maps.

multipath -v2 -d

Displays multipath map information for potential multipath devices in a dry run. The -v2 option shows only local disks. This verbosity level prints the created or updated multipath names only for use to feed other tools like kpartx.

There is no output if the devices already exists and there are no changes. Use `multipath -ll` to see the status of configured multipath devices.

multipath -v2 *`devicename`*

Configures a specific potential multipath device and displays multipath map information for it. This verbosity level prints only the created or updated multipath names for use to feed other tools like `kpartx`.

There is no output if the device already exists and there are no changes. Use `multipath -ll` to see the status of configured multipath devices.

Replace *`devicename`* with the device node name such as `/dev/sdb` (as shown by `udev` in the $DEVNAME variable), or in the `major:minor` format.

multipath -v3

Configures potential multipath devices and displays multipath map information for them. This verbosity level prints all detected paths, multipaths, and device maps. Both wwid and devnode blacklisted devices are displayed.

multipath -v3 *`devicename`*

Configures a specific potential multipath device and displays information for it. The -v3 option shows the full path list. This verbosity level prints all detected paths, multipaths, and device maps. Both wwid and devnode blacklisted devices are displayed.

Replace *`devicename`* with the device node name such as `/dev/sdb` (as shown by `udev` in the $DEVNAME variable), or in the `major:minor` format.

multipath -ll
Display the status of all multipath devices.

multipath -ll *devicename*
Displays the status of a specified multipath device.

Replace *devicename* with the device node name such as /dev/sdb (as shown by udev in the $DEVNAME variable), or in the major:minor format.

multipath -F
Flushes all unused multipath device maps. This unresolves the multiple paths; it does not delete the devices.

multipath -F *devicename*
Flushes unused multipath device maps for a specified multipath device. This unresolves the multiple paths; it does not delete the device.

Replace *devicename* with the device node name such as /dev/sdb (as shown by udev in the $DEVNAME variable), or in the major:minor format.

multipath -p [failover | multibus | group_by_serial | group_by_prio | group_by_node_name]
Sets the group policy by specifying one of the group policy options that are described in Table 5.3, "Group Policy Options for the multipath -p Command" (page 71):

Table 5.3 *Group Policy Options for the multipath -p Command*

Policy Option	Description
failover	One path per priority group. You can use only one path at a time.
multibus	All paths in one priority group.
group_by_serial	One priority group per detected SCSI serial number (the controller node worldwide number).
group_by_prio	One priority group per path priority value. Paths with the same priority are in the same priority group. Priorities are determined by callout programs specified as a global, per-

Policy Option	Description
	controller, or per-multipath option in the `/etc/multipath.conf` configuration file.
group_by_node_name	One priority group per target node name. Target node names are fetched in the`/sys/class/fc_transport/target*/node_name` location.

5.4 Configuring the System for Multipathing

5.4.1 Preparing SAN Devices for Multipathing

Before configuring multipath I/O for your SAN devices, prepare the SAN devices, as necessary, by doing the following:

- Configure and zone the SAN with the vendor’s tools.
- Configure permissions for host LUNs on the storage arrays with the vendor’s tools.
- Install the Linux HBA driver module. Upon module installation, the driver automatically scans the HBA to discover any SAN devices that have permissions for the host. It presents them to the host for further configuration.

NOTE

Ensure that the HBA driver you are using does not have native multipathing enabled.

See the vendor's specific instructions for more details.

- After the driver module is loaded, discover the device nodes assigned to specific array LUNs or partitions.
- If the SAN device will be used as the root device on the server, modify the timeout settings for the device as described in Section 5.2.6, "SAN Timeout Settings When the Root Device Is Multipathed" (page 59).

If the LUNs are not seen by the HBA driver, `lsscsi` can be used to check whether the SCSI devices are seen correctly by the operating system. When the LUNs are not seen by the HBA driver, check the zoning setup of the SAN. In particular, check whether LUN masking is active and whether the LUNs are correctly assigned to the server.

If the LUNs are seen by the HBA driver, but there are no corresponding block devices, additional kernel parameters are needed to change the SCSI device scanning behavior, such as to indicate that LUNs are not numbered consecutively. For information, see *Options for SCSI Device Scanning* [http://support.novell.com/techcenter/sdb/en/2005/06/drahn_scsi_scanning.html] in the Novell Support Knowledgebase.

5.4.2 Partitioning Multipathed Devices

Partitioning devices that have multiple paths is not recommended, but it is supported.

SUSE Linux Enterprise Server 10

In SUSE Linux Enterprise Server 10, you can use the `kpartx` tool to create partitions on multipathed devices without rebooting. You can also partition the device before you attempt to configure multipathing by using the Partitioner function in YaST2 or by using a third-party partitioning tool.

SUSE Linux Enterprise Server 9

In SUSE Linux Enterprise Server 9, if you want to partition the device, you should configure its partitions before you attempt to configure multipathing by using the Partitioner function in YaST2 or by using a third-party partitioning tool. This is necessary because partitioning an existing multipathed device is not supported. Partitioning operations on multipathed devices fail if attempted.

If you configure partitions for a device, DM-MP automatically recognizes the partitions and indicates them by appending p1 to p*n* to the device's ID, such as

/dev/disk/by-id/*26353900f02796769p1*

To partition multipathed devices, you must disable the DM-MP service, partition the normal device node (such as /dev/sdc), then reboot to allow the DM-MP service to see the new partitions.

5.4.3 Configuring the Server for Multipathing

The system must be manually configured to automatically load the device drivers for the controllers to which the multipath I/O devices are connected within the INITRD. Therefore add the needed driver module to the variable INITRD_MODULES in the file /etc/sysconfig/kernel.

For example, if your system contains a RAID controller accessed by the cciss driver and multipathed devices connected to a QLogic controller accessed by the driver qla2xxx, this entry would look like:

```
INITRD_MODULES="cciss"
```

Because the QLogic driver is not automatically loaded on start-up, add it here:

```
INITRD_MODULES="cciss qla23xx"
```

After having changed /etc/sysconfig/kernel, you must re-create the INITRD on your system with the command mkinitrd, then reboot in order for the changes to take effect.

When you are using LILO as a boot manager, reinstall it with the command `/sbin/lilo`. No further action is required if you are using GRUB.

5.4.4 Adding multipathd to the Boot Sequence

Use either of the methods in this section to add multipath I/O services (`multipathd`) to the boot sequence.

- Section "YaST" (page 75)
- Section "Command Line" (page 75)

YaST

1. In YaST, click *System* > *System Services (Runlevel)* > *Simple Mode*.
2. Select *multipathd*, then click *Enable*.
3. Click *OK* to acknowledge the service startup message.
4. Click *Finish*, then click *Yes*.

   The changes do not take affect until the server is restarted.

Command Line

1. Open a terminal console, then log in as the `root` user or equivalent.
2. At the terminal console prompt, enter

   ```
   insserv multipathd
   ```

5.4.5 Creating and Configuring the /etc/multipath.conf File

The /etc/multipath.conf file does not exist unless you create it. The /usr/share/doc/packages/multipath-tools/multipath.conf.synthetic file contains a sample /etc/multipath.conf file that you can use as a guide for multipath settings. See /usr/share/doc/packages/multipath-tools/multipath.conf.annotated for a template with extensive comments for each of the attributes and their options.

- Section "Creating the multipath.conf File" (page 76)
- Section "Verifying the Setup in the etc/multipath.conf File" (page 77)
- Section "Configuring User-Friendly Names or Alias Names in /etc/multipath.conf" (page 79)
- Section "Blacklisting Non-Multipathed Devices in /etc/multipath.conf" (page 83)
- Section "Configuring Default Multipath Behavior in /etc/multipath.conf" (page 84)
- Section "Applying Changes Made to the /etc/multipath.conf File" (page 85)

Creating the multipath.conf File

If the /etc/multipath.conf file does not exist, copy the example to create the file:

1 In a terminal console, log in as the root user.

2 Enter the following command (all on one line, of course) to copy the template:

```
cp /usr/share/doc/packages/multipath-tools/multipath.conf.synthetic
/etc/multipath.conf
```

3 Use the /usr/share/doc/packages/multipath-tools/multipath.conf.annotated file as a reference to determine how to configure multipathing for your system.

4. Make sure there is an appropriate `device` entry for your SAN. Most vendors provide documentation on the proper setup of the `device` section.

   The `/etc/multipath.conf` file requires a different `device` section for different SANs. If you are using a storage subsystem that is automatically detected (see Section "Tested Storage Arrays for Multipathing Support" (page 63)), the default entry for that device can be used; no further configuration of the `/etc/multipath.conf` file is required.

5. Save the file.

Verifying the Setup in the etc/multipath.conf File

After setting up the configuration, you can perform a "dry run" by entering

```
multipath -v2 -d
```

This command scans the devices, then displays what the setup would look like. The output is similar to the following:

```
26353900f02796769
[size=127 GB]
[features="0"]
[hwhandler="1    emc"]

\_ round-robin 0 [first]
  \_ 1:0:1:2 sdav 66:240  [ready ]
  \_ 0:0:1:2 sdr  65:16   [ready ]

\_ round-robin 0
  \_ 1:0:0:2 sdag 66:0    [ready ]
  \_ 0:0:0:2 sdc   8:32   [ready ]
```

Paths are grouped into priority groups. Only one priority group is in active use at a time. To model an active/active configuration, all paths end in the same group. To model active/passive configuration, the paths that should not be active in parallel are placed in several distinct priority groups. This normally happens automatically on device discovery.

The output shows the order, the scheduling policy used to balance I/O within the group, and the paths for each priority group. For each path, its physical address (host:bus:target:lun), device node name, major:minor number, and state is shown.

By using a verbosity level of -v3 in the dry run, you can see all detected paths, multipaths, and device maps. Both wwid and device node blacklisted devices are displayed.

```
multipath -v3 d
```

The following is an example of -v3 output on a 64-bit SLES server with two Qlogic HBA connected to a Xiotech Magnitude 3000 SAN. Some multiple entries have been omitted to shorten the example.

```
dm-22: device node name blacklisted
< content omitted >
loop7: device node name blacklisted
< content omitted >
md0: device node name blacklisted
< content omitted >
dm-0: device node name blacklisted
sdf: not found in pathvec
sdf: mask = 0x1f
sdf: dev_t = 8:80
sdf: size = 105005056
sdf: subsystem = scsi
sdf: vendor = XIOtech
sdf: product = Magnitude 3D
sdf: rev = 3.00
sdf: h:b:t:l = 1:0:0:2
sdf: tgt_node_name = 0x202100d0b2028da
sdf: serial = 000028DA0014
sdf: getuid = /sbin/scsi_id -g -u -s /block/%n (config file default)
sdf: uid = 200d0b2da28001400 (callout)
sdf: prio = const (config file default)
sdf: const prio = 1
< content omitted >
ram15: device node name blacklisted
< content omitted >
===== paths list =====
uuid              hcil    dev dev_t pri dm_st  chk_st  vend/prod/rev
200d0b2da28001400 1:0:0:2 sdf 8:80  1   [undef][undef] XIOtech,Magnitude 3D
200d0b2da28005400 1:0:0:1 sde 8:64  1   [undef][undef] XIOtech,Magnitude 3D
200d0b2da28004d00 1:0:0:0 sdd 8:48  1   [undef][undef] XIOtech,Magnitude 3D
200d0b2da28001400 0:0:0:2 sdc 8:32  1   [undef][undef] XIOtech,Magnitude 3D
200d0b2da28005400 0:0:0:1 sdb 8:16  1   [undef][undef] XIOtech,Magnitude 3D
200d0b2da28004d00 0:0:0:0 sda 8:0   1   [undef][undef] XIOtech,Magnitude 3D
params = 0 0 2 1 round-robin 0 1 1 8:80 1000 round-robin 0 1 1 8:32 1000
status = 2 0 0 0 2 1 A 0 1 0 8:80 A 0 E 0 1 0 8:32 A 0
sdf: mask = 0x4
sdf: path checker = directio (config file default)
directio: starting new request
directio: async io getevents returns 1 (errno=Success)
directio: io finished 4096/0
sdf: state = 2
< content omitted >
```

Configuring User-Friendly Names or Alias Names in /etc/multipath.conf

A multipath device can be identified by its WWID, by a user-friendly name, or by an alias that you assign for it. Table 5.4, "Comparison of Multipath Device Name Types" (page 79) describes the types of device names that can be used for a device in the `/etc/multipath.conf` file.

Table 5.4 *Comparison of Multipath Device Name Types*

Name Types	Description
WWID (default)	The WWID (Worldwide Identifier) is an identifier for the multipath device that is guaranteed to be globally unique and unchanging. The default name used in multipathing is the ID of the logical unit as found in the `/dev/disk/by-id` directory. Because device node names in the form of `/dev/sdn` and `/dev/dm-n` can change on reboot, referring to multipath devices by their ID is preferred.
User-friendly	The Device Mapper Multipath device names in the `/dev/mapper` directory also reference the ID of the logical unit. These multipath device names are user-friendly names in the form of `/dev/mapper/mpath<`*n*`>`, such as `/dev/mapper/mpath0`. The names are unique and persistent because they use the `/var/lib/multipath/bindings` file to track the association between the UUID and user-friendly names.
Alias	An alias name is a globally unique name that the administrator provides for a multipath device. Alias names override the WWID and the user-friendly `/dev/mapper/mpathN` names.

The global multipath `user_friendly_names` option in the `/etc/multipath.conf` file is used to enable or disable the use of user-friendly names for multipath devices. If it is set to "no" (the default), multipath uses the WWID as the name of the device. If it is set to "yes", multipath uses the `/var/lib/multipath/bindings` file to assign a persistent and unique name to the device in the form of `mpath<`*n*`>`.

The bindings_file option in the /etc/multipath.conf file can be used to specify an alternate location for the bindings file.

The global multipath alias option in the /etc/multipath.conf file is used to explicitly assign a name to the device. If an alias name is set up for a multipath device, the alias is used instead of the WWID or the user-friendly name.

Using the user_friendly_names option can be problematic in the following situations:

- **Root Device Is Using Multipath:** If the system root device is using multipath and you use the user_friendly_names option, the user-friendly settings in the /var/lib/multipath/bindings file are included in the initrd. If you later change the storage setup, such as by adding or removing devices, there is a mismatch between the bindings setting inside the initrd and the bindings settings in /var/lib/multipath/bindings.

  WARNING

  A bindings mismatch between initrd and /var/lib/multipath/bindings can lead to a wrong assignment of mount points to devices, which can result in file system corruption and data loss.

  To avoid this problem, we recommend that you use the default WWID settings for the system root device. You can also use the alias option to override the user_friendly_names option for the system root device in the /etc/multipath.conf file.

  For example:

```
multipaths {
       multipath {
               wwid          36006048000028350131253594d303030
               alias            mpatha
       }
       multipath {
               wwid          36006048000028350131253594d303041
               alias            mpathb
       }
       multipath {
               wwid          36006048000028350131253594d303145
               alias            mpathc
       }
```

```
        multipath {
                wwid          36006048000028350131253594d303334
                alias            mpathd
        }
}
```

IMPORTANT

We recommend that you do not use aliases for the system root device, because the ability to seamlessly switch off multipathing via the kernel command line is lost because the device name differs.

- **Mounting /var from Another Partition:** The default location of the `user_friendly_names` configuration file is `/var/lib/multipath/bindings`. If the `/var` data is not located on the system root device but mounted from another partition, the `bindings` file is not available when setting up multipathing.

  Make sure that the `/var/lib/multipath/bindings` file is available on the system root device and multipath can find it. For example, this can be done as follows:

  1. Move the `/var/lib/multipath/bindings` file to `/etc/multipath/bindings`.

  2. Set the `bindings_file` option in the `defaults` section of `/etc/multipath.conf` to this new location. For example:

     ```
     defaults {
                    user_friendly_names yes
                    bindings_file "/etc/multipath/bindings"
     }
     ```

- **Multipath Is in the initrd:** Even if the system root device is not on multipath, it is possible for multipath to be included in the `initrd`. For example, this can happen of the system root device is on LVM. If you use the `user_friendly_names` option and multipath is in the `initrd`, you should boot with the parameter `multipath=off` to avoid problems.

  This disables multipath only in the `initrd` during system boots. After the system boots, the `boot.multipath` and `multipathd` boot scripts are able to activate multipathing.

For an example of multipath.conf settings, see the /usr/share/doc/packages/multipath-tools/multipath.conf.synthetic file.

To enable user-friendly names or to specify aliases:

1 In a terminal console, log in as the root user.

2 Open the /etc/multipath.conf file in a text editor.

3 (Optional) Modify the location of the /var/lib/multipath/bindings file.

The alternate path must be available on the system root device where multipath can find it.

3a Move the /var/lib/multipath/bindings file to /etc/multipath/bindings.

3b Set the bindings_file option in the defaults section of /etc/multipath.conf to this new location. For example:

```
defaults {
               user_friendly_names yes
               bindings_file "/etc/multipath/bindings"
}
```

4 (Optional) Enable user-friendly names:

4a Uncomment the defaults section and its ending bracket.

4b Uncomment the user_friendly_names option, then change its value from No to Yes.

For example:

```
## Use user friendly names, instead of using WWIDs as names.
defaults {
  user_friendly_names yes
}
```

5 (Optional) Specify your own names for devices by using the alias option in the multipath section.

For example:

```
## Use alias names, instead of using WWIDs as names.
multipaths {
       multipath {
               wwid           36006048000028350131253594d303030
               alias             mpatha
       }
       multipath {
               wwid           36006048000028350131253594d303041
               alias             mpathb
       }
       multipath {
               wwid           36006048000028350131253594d303145
               alias             mpathc
       }
       multipath {
               wwid           36006048000028350131253594d303334
               alias             mpathd
       }
}
```

6 Save your changes, then close the file.

Blacklisting Non-Multipathed Devices in /etc/multipath.conf

The `/etc/multipath.conf` file should contain a `blacklist` section where all non-multipathed devices should be listed. For example, local IDE hard drives and floppy drives are not normally multipathed. If you have single-path devices that multipath is trying to manage and you want `multipath` to ignore them, put them in the `blacklist` section to resolve the problem.

NOTE

Beginning in SLES 10 SP3, the keyword devnode_blacklist has been deprecated and replaced with the keyword blacklist.

For example, to blacklist local devices and all arrays from the `cciss` driver from being managed by multipath, the `blacklist` section looks like this:

```
blacklist {
      wwid 26353900f02796769
      devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st|sda)[0-9]*"
      devnode "^hd[a-z][0-9]*"
      devnode "^cciss!c[0-9]d[0-9].*"
}
```

You can also blacklist only the partitions from a driver instead of the entire array. For example, using the following regular expression would blacklist only partitions from the cciss driver and not the entire array:

```
^cciss!c[0-9]d[0-9]*[p[0-9]*]
```

After you modify the `/etc/multipath.conf` file, you must run `mkinitrd` to re-create the INITRD on your system, then reboot in order for the changes to take effect.

Afterwards, the local devices should no longer be listed in the multipath maps when you issue the `multipath -ll` command.

Configuring Default Multipath Behavior in /etc/multipath.conf

The `/etc/multipath.conf` file should contain a `defaults` section where you can specify default behaviors. If the field is not otherwise specified in a `device` section, the default setting is applied for that SAN configuration.

The following `defaults` section specifies a simple failover policy:

```
defaults {
       multipath_tool  "/sbin/multipath -v0"
       udev_dir        /dev
       polling_interval 10
       default_selector        "round-robin 0"
       default_path_grouping_policy    failover
       default_getuid "/sbin/scsi_id -g -u -s /block/%n"
       default_prio_callout    "/bin/true"
       default_features        "0"
       rr_min_io               100
       failback                immediate
```

NOTE

In the `default_getuid` command line, use the path `/sbin/scsi_id` as shown in the above example instead of the sample path of `/lib/udev/scsi_id` that is found in the sample file `/usr/share/doc/packages/`

multipath-tools/multipath.conf.synthetic file (and in the default and annotated sample files).

Applying Changes Made to the /etc/multipath.conf File

Changes to the /etc/multipath.conf file cannot take effect when multipathd is running. After you make changes, save and close the file, then do the following to apply the changes:

1 Stop the multipathd service.

2 Clear old multipath bindings by entering

```
/sbin/multipath -F
```

3 Create new multipath bindings by entering

```
/sbin/multipath -v2 -l
```

4 Start the multipathd service.

5 Run mkinitrd to re-create the INITRD on your system, then reboot in order for the changes to take effect.

5.5 Enabling and Starting Multipath I/O Services

To start multipath services and enable them to start at reboot:

1 Open a terminal console, then log in as the root user or equivalent.

2 At the terminal console prompt, enter

```
chkconfig multipathd on

chkconfig boot.multipath on
```

If the boot.multipath service does not start automatically on system boot, do the following:

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Enter

```
/etc/init.d/boot.multipath start
/etc/init.d/multipathd start
```

5.6 Configuring Path Failover Policies and Priorities

In a Linux host, when there are multiple paths to a storage controller, each path appears as a separate block device, and results in multiple block devices for single LUN. The Device Mapper Multipath service detects multiple paths with the same LUN ID, and creates a new multipath device with that ID. For example, a host with two HBAs attached to a storage controller with two ports via a single unzoned Fibre Channel switch sees four block devices: `/dev/sda`, `/dev/sdb`, `/dev/sdc`, and `/dev/sdd`. The Device Mapper Multipath service creates a single block device, `/dev/mpath/mpath1` that reroutes I/O through those four underlying block devices.

This section describes how to specify policies for failover and configure priorities for the paths.

5.6.1 Configuring the Path Failover Policies

Use the `multipath` command with the -p option to set the path failover policy:

```
multipath devicename -p policy
```

Replace *`policy`* with one of the following policy options:

***Table 5.5** Group Policy Options for the multipath -p Command*

Policy Option	Description
failover	One path per priority group.
multibus	All paths in one priority group.
group_by_serial	One priority group per detected serial number.
group_by_prio	One priority group per path priority value. Priorities are determined by callout programs specified as a global, per-controller, or per-multipath option in the `/etc/multipath.conf` configuration file.
group_by_node_name	One priority group per target node name. Target node names are fetched in the`/sys/class/fc_transport/target*/node_name` location.

5.6.2 Configuring Failover Priorities

You must manually enter the failover priorities for the device in the `/etc/multipath.conf` file. Examples for all settings and options can be found in the `/usr/share/doc/packages/multipath-tools/multipath.conf.annotated` file.

- Section “Understanding Priority Groups and Attributes” (page 88)
- Section “Configuring for Round-Robin Load Balancing” (page 94)
- Section “Configuring for Single Path Failover” (page 94)

- Section “Grouping I/O Paths for Round-Robin Load Balancing” (page 95)

Understanding Priority Groups and Attributes

A *priority group* is a collection of paths that go to the same physical LUN. By default, I/O is distributed in a round-robin fashion across all paths in the group. The `multipath` command automatically creates priority groups for each LUN in the SAN based on the path_grouping_policy setting for that SAN. The `multipath` command multiplies the number of paths in a group by the group’s priority to determine which group is the primary. The group with the highest calculated value is the primary. When all paths in the primary group are failed, the priority group with the next highest value becomes active.

A *path priority* is an integer value assigned to a path. The higher the value, the higher is the priority. An external program is used to assign priorities for each path. For a given device, its paths with the same priorities belong to the same priority group.

Table 5.6 *Multipath Attributes*

Multipath Attribute	**Description**	**Values**
user_friendly_names	Specifies whether to use IDs or to use the `/var/lib/multipath/bindings` file to assign a persistent and unique alias to the multipath devices in the form of `/dev/mapper/mpathN`.	**yes** Autogenerate user-friendly names as aliases for the multipath devices instead of the actual ID. **no** Default. Use the WWIDs shown in the `/dev/disk/by-id/` location.
blacklist	Specifies the list of device names to ignore as non-multipathed devices, such as cciss, fd,	For an example, see Section “Blacklisting Non-Multipathed Devices in /etc/multipath.conf” (page 83).

Multipath Attribute	Description	Values
	hd, md, dm, sr, scd, st, ram, raw, loop.	
blacklist_exceptions	Specifies the list of device names to treat as multipath devices even if they are included in the blacklist.	For an example, see the `/usr/share/doc/packages/multipath-tools/multipath.conf.annotated` file.

getuid	The default program and argumentss to call out to obtain a unique path identifier. Should be specified with an absolute path.	`/sbin/scsi_id -g -u -s /block/%n`
		This is the default location and arguments.
		Example:
		`getuid "/sbin/scsi_id -g -u -d /dev/%n"`

path_grouping_policy	Specifies the path grouping policy for a multipath device hosted by a given controller.	**failover** One path is assigned per priority group so that only one path at a time is used. **multibus** (Default) All valid paths are in one priority group. Traffic is load-balanced across all active paths in the group. **group_by_prio** One priority group exists for each path priority value. Paths with the same priority are in the same priority group. Priorities are assigned by an external program. **group_by_serial** Paths are grouped by the SCSI target serial number (controller node WWN).

Multipath Attribute	Description	Values
		group_by_node_name One priority group is assigned per target node name. Target node names are fetched in `/sys/class/fc_transport/target*/node_name`.
path_checker	Determines the state of the path.	**directio** (Default in `multipath-tools` version 0.4.8 and later) Reads the first sector that has direct I/O. This is useful for DASD devices. Logs failure messages in `/var/log/messages`. **readsector0** (Default in `multipath-tools` version 0.4.7 and earlier) Reads the first sector of the device. Logs failure messages in `/var/log/messages`. **tur** Issues a SCSI test unit ready command to the device. This is the preferred setting if the LUN supports it. The command does not fill up `/var/log/messages` on failure with messages. Some SAN vendors provide custom `path_checker` options: • **`emc_clariion`** Queries the EMC Clariion EVPD page 0xC0 to determine the path state.

Multipath Attribute	Description	Values
		- **`hp_sw`** Checks the path state (Up, Down, or Ghost) for HP storage arrays with Active/Standby firmware. - **`rdac`** Checks the path state for the LSI/Engenio RDAC storage controller.
path_selector	Specifies the path-selector algorithm to use for load-balancing.	**round-robin 0** (Default) The load-balancing algorithm used to balance traffic across all active paths in a priority group. This is currently the only algorithm available.
pg_timeout	Specifies path group timeout handling.	NONE (internal default)
prio_callout	Specifies the program and arguments to use to determine the layout of the multipath map. When queried by the `multipath` command, the specified mpath_prio_* callout program returns the priority for a given path in relation to the entire multipath layout. When it is used with the path_grouping_policy	If no `prio_callout` attribute is used, all paths are equal. This is the default. **/bin/true** Use this value when the group_by_priority is not being used. The *prioritizer* programs generate path priorities when queried by the `multipath` command. The program names must begin with `mpath_prio_` and are named by the device type or balancing method used. Current prioritizer programs include the following:

Multipath Attribute	Description	Values
	of group_by_prio, all paths with the same priority are grouped into one multipath group. The group with the highest aggregate priority becomes the active group. When all paths in a group fail, the group with the next highest aggregate priority becomes active. Additionally, a failover command (as determined by the hardware handler) might be send to the target. The mpath_prio_* program can also be a custom script created by a vendor or administrator for a specified setup. A %n in the command line expands to the device name in the `/dev` directory. A %b expands to the device number in *`major:minor`* format in the `/dev` directory.	**/sbin/mpath_prio_alua %n** Generates path priorities based on the SCSI-3 ALUA settings. **/sbin/mpath_prio_balance_units** Generates the same priority for all paths. **/sbin/mpath_prio_emc %n** Generates the path priority for EMC arrays. **/sbin/mpath_prio_hds_modular %b** Generates the path priority for Hitachi HDS Modular storage arrays. **/sbin/mpath_prio_hp_sw %n** Generates the path priority for Compaq/HP controller in active/standby mode. **/sbin/mpath_prio_netapp %n** Generates the path priority for NetApp arrays. **/sbin/mpath_prio_random %n** Generates a random priority for each path. **/sbin/mpath_prio_rdac %n** Generates the path priority for LSI/Engenio RDAC controller. **/sbin/mpath_prio_tpc %n** You can optionally use a script created by a vendor or administrator that gets the priorities from a file where you specify priorities to use for each path. **/usr/local/sbin/mpath_prio_spec.sh %n** Provides the path of a user-created script that generates the priorities for

Multipath Attribute	Description	Values
	A %d expands to the device ID in the /dev/disk/by-id directory. If devices are hot-pluggable, use the %d flag instead of %n. This addresses the short time that elapses between the time when devices are available and when udev creates the device nodes.	multipathing based on information contained in a second data file. (This path and filename are provided as an example. Specify the location of your script instead.) The script can be created by a vendor or administrator. The script's target file identifies each path for all multipathed devices and specifies a priority for each path. For an example, see Section 5.6.3, "Using a Script to Set Path Priorities" (page 95).
rr_min_io	Specifies the number of I/O transactions to route to a path before switching to the next path in the same path group, as determined by the specified algorithm in the path_selector setting.	**n (>0)** Specify an integer value greater than 0. **1000** Default.
rr_weight	Specifies the weighting method to use for paths.	**uniform** Default. All paths have the same round-robin weightings. **priorities** Each path's weighting is determined by the path's priority times the rr_min_io setting.
no_path_retry	Specifies the behaviors to use on path failure.	**n (> 0)** Specifies the number of retries until multipath stops the queuing and fails the path. Specify an integer value greater than 0.

Multipath Attribute	Description	Values
		fail Specified immediate failure (no queuing). **queue** Never stop queuing (queue forever until the path comes alive).
failback	Specifies whether to monitor the failed path recovery, and indicates the timing for group failback after failed paths return to service. When the failed path recovers, the path is added back into the multipath enabled path list based on this setting. Multipath evaluates the priority groups, and changes the active priority group when the priority of the primary path exceeds the secondary group.	**immediate** When a path recovers, enable the path immediately. **n (> 0)** When the path recovers, wait n seconds before enabling the path. Specify an integer value greater than 0. **manual** (Default) The failed path is not monitored for recovery. The administrator runs the `multipath` command to update enabled paths and priority groups.

Configuring for Round-Robin Load Balancing

All paths are active. I/O is configured for some number of seconds or some number of I/O transactions before moving to the next open path in the sequence.

Configuring for Single Path Failover

A single path with the highest priority (lowest value setting) is active for traffic. Other paths are available for failover, but are not used unless failover occurs.

Grouping I/O Paths for Round-Robin Load Balancing

Multiple paths with the same priority fall into the active group. When all paths in that group fail, the device fails over to the next highest priority group. All paths in the group share the traffic load in a round-robin load balancing fashion.

5.6.3 Using a Script to Set Path Priorities

You can create a script that interacts with DM-MP to provide priorities for paths to the LUN when set as a resource for the `prio_callout` setting.

First, set up a text file that lists information about each device and the priority values you want to assign to each path. For example, name the file `/usr/local/etc/primary-paths`. Enter one line for each path in the following format:

```
host_wwpn target_wwpn scsi_id priority_value
```

Return a priority value for each path on the device. Make sure that the variable FILE_PRIMARY_PATHS resolves to a real file with appropriate data (host wwpn, target wwpn, scsi_id and priority value) for each device.

The contents of the `primary-paths` file for a single LUN with eight paths each might look like this:

```
0x10000000c95ebeb4 0x200200a0b8122c6e 2:0:0:0 sdb
3600a0b8000122c6d00000000453174fc 50

0x10000000c95ebeb4 0x200200a0b8122c6e 2:0:0:1 sdc
3600a0b80000fd6320000000045317563 2

0x10000000c95ebeb4 0x200200a0b8122c6e 2:0:0:2 sdd
3600a0b8000122c6d0000000345317524 50

0x10000000c95ebeb4 0x200200a0b8122c6e 2:0:0:3 sde
3600a0b80000fd6320000000245317593 2

0x10000000c95ebeb4 0x200300a0b8122c6e 2:0:1:0 sdi
3600a0b8000122c6d00000000453174fc 5

0x10000000c95ebeb4 0x200300a0b8122c6e 2:0:1:1 sdj
3600a0b80000fd6320000000045317563 51

0x10000000c95ebeb4 0x200300a0b8122c6e 2:0:1:2 sdk
3600a0b8000122c6d0000000345317524 5

0x10000000c95ebeb4 0x200300a0b8122c6e 2:0:1:3 sdl
3600a0b80000fd6320000000245317593 51
```

To continue the example mentioned in Table 5.6, “Multipath Attributes” (page 88), create a script named /usr/local/sbin/path_prio.sh. You can use any path and filename. The script does the following:

- On query from multipath, grep the device and its path from the /usr/local/etc/primary-paths file.
- Return to multipath the priority value in the last column for that entry in the file.

5.6.4 Configuring ALUA

The mpath_prio_alua(8) command is used as a priority callout for the Linux multipath(8) command. It returns a number that is used by DM-MP to group SCSI devices with the same priority together. This path priority tool is based on ALUA (Asynchronous Logical Unit Access).

- Section “Syntax” (page 96)
- Section “Prerequisite” (page 96)
- Section “Options” (page 96)
- Section “Return Values” (page 97)

Syntax

```
mpath_prio_alua [-d directory] [-h] [-v] [-V] device [device...]
```

Prerequisite

SCSI devices

Options

-d *directory*

Specifying the Linux directory path where the listed device node names can be found. The default directory is /dev. When used, specify the device node name only (such as sda) for the device or devices you want to manage.

-h

Displays help for this command, then exits.

-v

Turns on verbose output to display status in human-readable format. Output includes information about which port group the specified device is in and its current state.

-V

Displays the version number of this tool, then exits.

device

Specifies the SCSI device you want to manage. The device must be a SCSI device that supports the Report Target Port Groups (`sg_rtpg(8)`) command. Use one of the following formats for the device node name:

- The full Linux directory path, such as `/dev/sda`. Do not use with the -d option.
- The device node name only, such as `sda`. Specify the directory path using the -d option.
- The major and minor number of the device separated by a colon (:) with no spaces, such as `8:0`. This creates a temporary device node in the `/dev` directory with a name in the format of `tmpdev-<major>:<minor>-<pid>`. For example, `/dev/tmpdev-8:0-<pid>`.

Return Values

On success, returns a value of 0 and the priority value for the group. Table 5.7, “ALUA Priorities for Device Mapper Multipath” (page 97) shows the priority values returned by the `mpath_prio_alua` command.

Table 5.7 *ALUA Priorities for Device Mapper Multipath*

Priority Value	Description
50	The device is in the active, optimized group.
10	The device is in an active but non-optimized group.

Priority Value	Description
1	The device is in the standby group.
0	All other groups.

Values are widely spaced because of the way the multipath command handles them. It multiplies the number of paths in a group with the priority value for the group, then selects the group with the highest result. For example, if a non-optimized path group has six paths (6 x 10 = 60) and the optimized path group has a single path (1 x 50 = 50), the non-optimized group has the highest score, so multipath chooses the non-optimized group. Traffic to the device uses all six paths in the group in a round-robin fashion.

On failure, returns a value of 1 to 5 indicating the cause for the command's failure. For information, see the man page for mpath_prio_alua.

5.6.5 Reporting Target Path Groups

Use the SCSI Report Target Port Groups (sg_rtpg(8)) command. For information, see the man page for sg_rtpg(8).

5.7 Configuring Multipath I/O for the Root Device

In the SUSE Linux Enterprise Server 10, the root partition (/) on multipath is supported only if the /boot partition is on a separate, non-multipathed partition. Otherwise, no boot loader is written.

IMPORTANT

If you apply all online updates, the DM-MP is available but is not supported for /boot and /root in SUSE Linux Enterprise Server 10 SP1 and later. More specifically, you need mkinitrd 1.2-106.61 and multipath-tools 0.4.7-

34.23 or later. However, if you install the packages and set up the configuration, you might run into update issues later.

Full multipath support is available in SUSE Linux Enterprise Server 11.

To enable multipathing on the existing root device:

1. Install Linux with only a single path active, preferably one where the `by-id` symlinks are listed in the partitioner.

2. Mount the devices by using the `/dev/disk/by-id` path used during the install.

3. After installation, add `dm-multipath` to `/etc/sysconfig/kernel:INITRD_MODULES`.

4. For System Z, before running mkinitrd, edit the `/etc/zipl.conf` file to change the by-path information in `zipl.conf` with the same by-id information that was used in the `/etc/fstab`.

5. Re-run `/sbin/mkinitrd` to update the `initrd` image.

6. For System Z, after running mkinitrd, run `zipl`.

7. Reboot the server.

To disable multipathing on the root device:

- Add `multipath=off` to the kernel command line.

  This affects only the root device. All other devices are not affected.

5.8 Configuring Multipath I/O for an Existing Software RAID

Ideally, you should configure multipathing for devices before you use them as components of a software RAID device. If you add multipathing after creating any software RAID devices, the DM-MP service might be starting after the `multipath` service on reboot, which makes multipathing appear not to be available for RAIDs. You can use the procedure in this section to get multipathing running for a previously existing software RAID.

For example, you might need to configure multipathing for devices in a software RAID under the following circumstances:

- If you create a new software RAID as part of the Partitioning settings during a new install or upgrade.
- If you did not configure the devices for multipathing before using them in the software RAID as a member device or spare.
- If you grow your system by adding new HBA adapters to the server or expanding the storage subsystem in your SAN.

NOTE

The following instructions assume the software RAID device is `/dev/mapper/mpath0`, which is its device name as recognized by the kernel. Make sure to modify the instructions for the device name of your software RAID.

---

1 Open a terminal console, then log in as the `root` user or equivalent.

Except where otherwise directed, use this console to enter the commands in the following steps.

2 If any software RAID devices are currently mounted or running, enter the following commands for each device to dismount the device and stop it.

```
umount /dev/mapper/mpath0
```

```
mdadm --misc --stop /dev/mapper/mpath0
```

3 Stop the `boot.md` service by entering

```
/etc/init.d/boot.md stop
```

4 Start the `boot.multipath` and `multipathd` services by entering the following commands:

```
/etc/init.d/boot.multipath start
```

```
/etc/init.s/multipathd start
```

5. After the multipathing services are started, verify that the software RAID's component devices are listed in the `/dev/disk/by-id` directory. Do one of the following:

   - **Devices Are Listed:** The device names should now have symbolic links to their Device Mapper Multipath device names, such as `/dev/dm-1`.

   - **Devices Are Not Listed:** Force the multipath service to recognize them by flushing and rediscovering the devices.

     To do this, enter the following commands:

     ```
     multipath -F
     ```

     ```
     multipath -v0
     ```

     The devices should now be listed in `/dev/disk/by-id`, and have symbolic links to their Device Mapper Multipath device names. For example:

     ```
     lrwxrwxrwx 1 root root 10 Jun 15 09:36 scsi-mpath1 -> ../../dm-1
     ```

6. Restart the `boot.md` service and the RAID device by entering

   ```
   /etc/init.d/boot.md start
   ```

7. Check the status of the software RAID by entering

   ```
   mdadm --detail /dev/mapper/mpath0
   ```

   The RAID's component devices should match their Device Mapper Multipath device names that are listed as the symbolic links of devices in the `/dev/disk/by-id` directory.

8. Make a new `initrd` to ensure that the Device Mapper Multipath services are loaded before the RAID services on reboot. Enter

   ```
   mkinitrd -f mpath
   ```

9. Reboot the server to apply these post-install configuration settings.

10 Verify that the software RAID array comes up properly on top of the multipathed devices by checking the RAID status. Enter

```
mdadm --detail /dev/mapper/mpath0
```

For example:

```
Number Major Minor RaidDevice State
0 253 0 0 active sync /dev/dm-0
1 253 1 1 active sync /dev/dm-1
2 253 2 2 active sync /dev/dm-2
```

5.9 Scanning for New Devices without Rebooting

If your system has already been configured for multipathing and you later need to add more storage to the SAN, you can use the `rescan-scsi-bus.sh` script to scan for the new devices. By default, this script scans all HBAs with typical LUN ranges.

Syntax

```
rescan-scsi-bus.sh [options] [host [host ...]]
```

You can specify hosts on the command line (deprecated), or use the `--hosts=LIST` option (recommended).

Options

For most storage subsystems, the script can be run successfully without options. However, some special cases might need to use one or more of the following parameters for the `rescan-scsi-bus.sh` script:

Option	Description
`-l`	Activates scanning for LUNs 0-7. [Default: 0]

Option	Description
`-L NUM`	Activates scanning for LUNs 0 to NUM. [Default: 0]
`-w`	Scans for target device IDs 0 to 15. [Default: 0 to 7]
`-c`	Enables scanning of channels 0 or 1. [Default: 0]
`-r` `--remove`	Enables removing of devices. [Default: Disabled]
`-i` `--issueLip`	Issues a Fibre Channel LIP reset. [Default: Disabled]
`--forcerescan`	Rescans existing devices.
`--forceremove`	Removes and re-adds every device. (DANGEROUS)
`--nooptscan`	Don’t stop looking for LUNs if 0 is not found.
`--color`	Use colored prefixes OLD/NEW/DEL.
`--hosts=LIST`	Scans only hosts in LIST, where LIST is a comma-separated list of single values and ranges. No spaces are allowed. `--hosts=A[-B][,C[-D]]`
`--channels=LIST`	Scans only channels in LIST, where LIST is a comma-separated list of single values and ranges. No spaces are allowed. `--channels=A[-B][,C[-D]]`
`--ids=LIST`	Scans only target IDs in LIST, where LIST is a comma-separated list of single values and ranges. No spaces are allowed. `--ids=A[-B][,C[-D]]`

Option	Description
`--luns=LIST`	Scans only LUNs in LIST, where LIST is a comma-separated list of single values and ranges. No spaces are allowed. `--luns=A[-B][,C[-D]]`

Procedure

Use the following procedure to scan the devices and make them available to multipathing without rebooting the system.

1 On the storage subsystem, use the vendor’s tools to allocate the device and update its access control settings to allow the Linux system access to the new storage. Refer to the vendor’s documentation for details.

2 Scan all targets for a host to make its new device known to the middle layer of the Linux kernel’s SCSI subsystem. At a terminal console prompt, enter

```
rescan-scsi-bus.sh [options]
```

3 Check for scanning progress in the system log (the `/var/log/messages` file). At a terminal console prompt, enter

```
tail -30 /var/log/messages
```

This command displays the last 30 lines of the log. For example:

```
# tail -30 /var/log/messages
. . .
Feb 14 01:03 kernel: SCSI device sde: 81920000
Feb 14 01:03 kernel: SCSI device sdf: 81920000
Feb 14 01:03 multipathd: sde: path checker registered
Feb 14 01:03 multipathd: sdf: path checker registered
Feb 14 01:03 multipathd: mpath4: event checker started
Feb 14 01:03 multipathd: mpath5: event checker started
Feb 14 01:03:multipathd: mpath4: remaining active paths: 1
Feb 14 01:03 multipathd: mpath5: remaining active paths: 1
```

4 Repeat Step 2 (page 104) through Step 3 (page 104) to add paths through other HBA adapters on the Linux system that are connected to the new device.

5 Run the `multipath` command to recognize the devices for DM-MP configuration. At a terminal console prompt, enter

```
multipath
```

You can now configure the new device for multipathing.

5.10 Scanning for New Partitioned Devices without Rebooting

Use the example in this section to detect a newly added multipathed LUN without rebooting.

1 Open a terminal console, then log in as the `root` user.

2 Scan all targets for a host to make its new device known to the middle layer of the Linux kernel's SCSI subsystem. At a terminal console prompt, enter

```
rescan-scsi-bus.sh [options]
```

For syntax and options information for the `rescan-scsi-bus-sh` script, see Section 5.9, "Scanning for New Devices without Rebooting" (page 102).

3 Verify that the device is seen (the link has a new time stamp) by entering

```
ls -lrt /dev/dm-*
```

4 Verify the new WWN of the device appears in the log by entering

```
tail -33 /var/log/messages
```

5 Use a text editor to add a new alias definition for the device in the `/etc/multipath.conf` file, such as `oradata3`.

6 Create a partition table for the device by entering

```
fdisk /dev/dm-8
```

7 Trigger udev by entering

```
echo 'add' > /sys/block/dm-8/uevent
```

This generates the device-mapper devices for the partitions on dm-8.

8 Create a file system and label for the new partition by entering

```
mke2fs -j /dev/dm-9
```

```
tune2fs -L oradata3 /dev/dm-9
```

9 Restart DM-MP to let it read the aliases by entering

```
/etc/init.d/multipathd restart
```

10 Verify that the device is recognized by multipathd by entering

```
multipath -ll
```

11 Use a text editor to add a mount entry in the /etc/fstab file.

At this point, the alias you created in Step 5 (page 105) is not yet in the /dev/disk/by-label directory. Add the mount entry the /dev/dm-9 path, then change the entry before the next time you reboot to

```
LABEL=oradata3
```

12 Create a directory to use as the mount point, then mount the device by entering

```
md /oradata3
```

```
mount /oradata3
```

5.11 Viewing Multipath I/O Status

Querying the multipath I/O status outputs the current status of the multipath maps.

The multipath -l option displays the current path status as of the last time that the path checker was run. It does not run the path checker.

The multipath -ll option runs the path checker, updates the path information, then displays the current status information. This option always the displays the latest information about the path status.

- At a terminal console prompt, enter

```
multipath -ll
```

This displays information for each multipathed device. For example:

```
3600601607cf30e00184589a37a31d911
[size=127 GB][features="0"][hwhandler="1 emc"]

\_ round-robin 0 [active][first]
  \_ 1:0:1:2 sdav 66:240  [ready ][active]
  \_ 0:0:1:2 sdr  65:16   [ready ][active]

\_ round-robin 0 [enabled]
  \_ 1:0:0:2 sdag 66:0    [ready ][active]
  \_ 0:0:0:2 sdc  8:32    [ready ][active]
```

For each device, it shows the device's ID, size, features, and hardware handlers.

Paths to the device are automatically grouped into priority groups on device discovery. Only one priority group is active at a time. For an active/active configuration, all paths are in the same group. For an active/passive configuration, the passive paths are placed in separate priority groups.

The following information is displayed for each group:

- Scheduling policy used to balance I/O within the group, such as round-robin
- Whether the group is active, disabled, or enabled
- Whether the group is the first (highest priority) group
- Paths contained within the group

The following information is displayed for each path:

- The physical address as *`host:bus:target:lun`*, such as 1:0:1:2
- Device node name, such as `sda`
- Major:minor numbers
- Status of the path and device

Each path line contains the following:

```
\_ host:channel:id:lun devnode major:minor [path_status] [dm_status]
```

When the path is up and ready for I/O, `path_status` shows a state of `ready` or `active`. When the path is down, the `path_status` shows a state of `faulty` or `failed`. The `path_status` is updated periodically based on the value of the `polling_interval` setting in `/etc/multipath.conf`. For information about the `polling_interval`, see Section "Configuring Default Multipath Behavior in /etc/multipath.conf" (page 84).

The `dm_status` field reports two states: `failed` and `active`.

It is normal for `path_status` and `dm_status` to temporarily disagree.

5.12 Managing I/O in Error Situations

You might need to configure multipathing to queue I/O if all paths fail concurrently. In certain scenarios, where the driver, the HBA, or the fabric experiences spurious errors, it is advisable that DM-MP be configured to queue all I/O where those errors lead to a loss of all paths, and never propagate errors upwards. Because this leads to I/O being queued indefinitely unless a path is reinstated, make sure that `multipathd` is running and works for your scenario. Otherwise, I/O might be stalled indefinitely on the affected multipathed device, until reboot or until you manually return to failover instead of queuing.

To test the scenario:

1. In a terminal console, log in as the `root` user.

2. Activate queuing instead of failover for the device I/O by entering:

   ```
   dmsetup message device_ID 0 queue_if_no_path
   ```

   Replace the *`device_ID`* with the ID for your device. For example, enter:

   ```
   dmsetup message 3600601607cf30e00184589a37a31d911 0 queue_if_no_path
   ```

3. Return to failover for the device I/O by entering:

   ```
   dmsetup message device_ID 0 fail_if_no_path
   ```

   This command immediately causes all queued I/O to fail.

Replace the *device_ID* with the ID for your device. For example, enter:

```
dmsetup message 3600601607cf30e00184589a37a31d911 0 fail_if_no_path
```

To set up queuing I/O for scenarios where all paths fail:

1. In a terminal console, log in as the `root` user.

2. Open the `/etc/multipath.conf` file in a text editor.

3. Uncomment the defaults section and its ending bracket, then add the `default_features` setting, as follows:

   ```
   defaults {
     default_features "1 queue_if_no_path"
   }
   ```

4. After you modify the `/etc/multipath.conf` file, you must run `mkinitrd` to re-create the INITRD on your system, then reboot in order for the changes to take effect.

5. When you are ready to return over to failover for the device I/O, enter:

   ```
   dmsetup message mapname 0 fail_if_no_path
   ```

   Replace the *mapname* with the mapped alias name or the device ID for the device.

   This command immediately causes all queued I/O to fail and propagates the error to the calling application.

5.13 Resolving Stalled I/O

If all paths fail concurrently and I/O is queued and stalled, do the following:

1. Enter the following command at a terminal console prompt:

   ```
   dmsetup message mapname 0 fail_if_no_path
   ```

   Replace *mapname* with the correct device ID or mapped alias name for the device. This causes all queued I/O to fail and propagates the error to the calling application.

2. Reactivate queueing by entering the following command at a terminal console prompt:

```
dmsetup message mapname 0 queue_if_no_path
```

5.14 Additional Information

For more information about configuring and using multipath I/O on SUSE Linux Enterprise Server, see the following additional resources in the Novell Support Knowledgebase:

- *Troubleshooting SLES Multipathing (MPIO) Problems (Technical Information Document 3231766)* [http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=3231766&sliceId=SAL_Public]

- *DM MPIO Device Blacklisting Not Honored in multipath.conf (Technical Information Document 3029706)* [http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=3029706&sliceId=SAL_Public&dialogID=57872426&stateId=0%200%2057878058]

- *Static Load Balancing in Device-Mapper Multipathing (DM-MP) (Technical Information Document 3858277)* [http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=3858277&sliceId=SAL_Public&dialogID=57872426&stateId=0%200%2057878058]

- *Troubleshooting SCSI (LUN) Scanning Issues (Technical Information Document 3955167)* [http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=3955167&sliceId=SAL_Public&dialogID=57868704&stateId=0%200%2057878206]

5.15 What’s Next

If you want to use software RAIDs, create and configure them before you create file systems on the devices. For information, see the following:

- Chapter 6, *Managing Software RAIDs with EVMS* (page 111)

- Chapter 7, *Managing Software RAIDs 6 and 10 with mdadm* (page 147)

Managing Software RAIDs with EVMS

6

This section describes how to create and manage software RAIDs with the Enterprise Volume Management System (EVMS). EVMS supports only RAIDs 0, 1, 4, and 5 at this time. For RAID 6 and 10 solutions, see Chapter 7, *Managing Software RAIDs 6 and 10 with mdadm* (page 147).

6.1 Understanding Software RAIDs on Linux

6.1.1 What Is a Software RAID?

A RAID combines multiple devices into a multi-disk array to provide resiliency in the storage device and to improve storage capacity and I/O performance. If a disk fails, some RAID levels keep data available in a degraded mode until the failed disk can be replaced and its content reconstructed.

A software RAID provides the same high availability that you find in a hardware RAID. The key operational differences are described in the following table:

Table 6.1 *Comparison of Software RAIDs and Hardware RAIDs*

Feature	Linux Software RAID	Hardware RAID
RAID function	Multi-disk (`md`) driver or `mdadm`	RAID controller on the disk array
RAID processing	In the host server’s processor	RAID controller on the disk array
RAID levels	0, 1, 4, 5, and 10 plus the `mdadm` raid10	Varies by vendor

Feature	Linux Software RAID	Hardware RAID
Component devices	Disks from same or different disk array	Same disk array

6.1.2 Overview of RAID Levels

The following table describes the advantages and disadvantages of the RAID levels supported by EVMS. The description assumes that the component devices reside on different disks and that each disk has its own dedicated I/O capability.

IMPORTANT

For information about creating complex or nested RAID devices with mdadm, see Chapter 7, *Managing Software RAIDs 6 and 10 with mdadm* (page 147).

Table 6.2 *RAID Levels Supported by EVMS*

RAID Level	Description	Performance and Fault Tolerance
0	Stripes data using a round-robin method to distribute data over the RAID's component devices.	Improves disk I/O performance for both reads and writes. Actual performance depends on the stripe size, the actual data, and the application. Does not provide disk fault tolerance and data redundancy. Any disk failure causes all data in the RAID to be lost.
1	Mirrors data by copying blocks of one disk to another and keeping them in continuous synchronization. If	Improves disk reads by making multiple copies of data available via different I/O paths. The write performance is about the same as for a single disk because a copy of the data must be written to each of the disks in the mirror.

RAID Level	Description	Performance and Fault Tolerance
	disks are different sizes, the smallest disk determines the size of the RAID.	Provides 100% data redundancy. If one disk fails then the data remains available on its mirror, and processing continues.
4	Stripes data and records parity to a dedicated disk. If disks are different sizes, the smallest disk determines the size of the RAID.	Improves disk I/O performance for both reads and writes. Write performance is considerably slower than for RAID 0, because parity must be calculated and written. Write performance is slightly slower than RAID 5. Read performance is slower than for a RAID 1 array with the same number of component devices. The dedicated parity disk can become a bottleneck for writing parity. Provides disk fault tolerance. If a disk fails, performance is degraded while the RAID uses the parity to reconstruct data for the replacement disk.
5	Stripes data and distributes parity in a round-robin fashion across all disks. If disks are different sizes, the smallest disk determines the size of the RAID.	Improves disk I/O performance for reads and writes. Write performance is considerably less than for RAID 0, because parity must be calculated and written. Write performance is faster than RAID 4. Read performance is slower than for a RAID 1 array with the same number of component disks. Actual performance depends on the number of component disks, the stripe size, the actual data, and the application. Provides disk fault tolerance. If a disk fails, performance is degraded while the RAID uses the parity to reconstruct data for the replacement disk. Provides slightly less data redundancy than mirroring because it uses parity to reconstruct the data.

6.1.3 Comparison of RAID Performance

The following table compares the read and write performance for RAID devices.

Table 6.3 Read and Write Performance for RAIDs

Raid Level	Read Performance	Write Performance
0	Faster than for a single disk	Faster than for a single disk and other RAIDs.
1	Faster than for a single disk, increasing as more mirrors are added	Slower than for a single disk, declining as more mirrors are added.
4	Faster than for a single disk. Slower than a RAID 0 because one disk is used for parity.	Faster than for a single disk. Slower than a RAID 0 because of writes for parity. Slower than a RAID 5 because of possible bottlenecks for writes of parity to the dedicated parity disk.
5	Faster than for a single disk; comparable to a RAID 0.	Faster than a single disk. Slower than a RAID 0 because of writes for parity.

6.1.4 Comparison of Disk Fault Tolerance

The following table compares the disk fault tolerance for RAID devices.

Table 6.4 Fault Tolerance for RAIDs

Raid Level	Number of Disk Failures Tolerated	Data Redundancy
0	None	No
1	Number of disks minus 1	100% redundancy for each mirror

Raid Level	Number of Disk Failures Tolerated	Data Redundancy
4	1	Dedicated parity disk to reconstruct data. If the parity disk fails, all parity must be recalculated.
5	1	Distributed parity to reconstruct data and parity on the failed disk.

6.1.5 Configuration Options for RAIDs

In EVMS management tools, the following RAID configuration options are provided:

Table 6.5 *Configuration Options in EVMS*

Option	Description
Spare Disk	For RAIDs 1, 4, and 5, you can optionally specify a device, segment, or region to use as the replacement for a failed disk (the member device, segment, or region). On failure, the spare disk automatically replaces the failed disk, then reconstructs the data. However, if the parity disk fails on a RAID 5, parity cannot be reconstructed.
Chunk Size (KB)	For RAIDs 0, 4, or 5, specify the stripe size in KB. Consider the intended use of the RAID, such as the file system block size, the applications used, and the actual data (file sizes and typical reads and writes). A typical write size for large files is 128 KB. Default: 32 KB Range: 4 KB to 4096 KB, in powers of 2.

Option	Description
RAID Level	If you selected *MD RAID 4/5 Region Manager*, specify *RAID 4* or *RAID 5* (default).
RAID Algorithm	For RAID 5, specify one of the following algorithms to use for striping and distributing parity on the disk. • Left Asymmetric • Left Symmetric (Default, fastest performance for large reads) • Right Asymmetric • Right Symmetric

6.1.6 Interoperability Issues

Linux software RAID cannot be used underneath clustered file systems because it does not support concurrent activation. If you want RAID and OCFS2, you need the RAID to be handled by the storage subsystem.

WARNING

Activating Linux software RAID devices concurrently on multiple servers can result in data corruption or inconsistencies.

6.1.7 Guidelines for Component Devices

For efficient use of space and performance, the disks you use to create the RAID should have the same storage capacity. Typically, if component devices are not of identical storage capacity, then each member of the RAID uses only an amount of space equal to the capacity of the smallest member disk.

Version 2.3 and later of `mdadm` supports component devices up to 4 TB in size each. Earlier versions support component devices up to 2 TB in size.

IMPORTANT

If you have a local disk, external disk arrays, or SAN devices that are larger than the supported device size, use a third-party disk partitioner to carve the devices into smaller logical devices.

You can combine up to 28 component devices to create the RAID array. The `md` RAID device you create can be up to the maximum device size supported by the file system you plan to use. For information about file system limits for SUSE Linux Enterprise Server 10, see “Large File System Support” in the *SUSE Linux Enterprise Server 10 Installation and Administration Guide*. [http://www.novell.com/documentation/sles10].

In general, each storage object included in the RAID should be from a different physical disk to maximize I/O performance and to achieve disk fault tolerance where supported by the RAID level you use. In addition, they should be of the same type (disks, segments, or regions).

Using component devices of differing speeds might introduce a bottleneck during periods of demanding I/O. The best performance can be achieved by using the same brand and models of disks and controllers in your hardware solution. If they are different, you should try to match disks and controllers with similar technologies, performance, and capacity. Use a low number of drives on each controller to maximize throughput.

IMPORTANT

As with any hardware solution, using the same brand and model introduces the risk of concurrent failures over the life of the product, so plan maintenance accordingly.

The following table provides recommendations for the minimum and maximum number of storage objects to use when creating a software RAID:

Table 6.6 *Recommended Number of Storage Objects to Use in the Software RAID*

RAID Type	Minimum Number of Storage Objects	Recommended Maximum Number of Storage Objects
RAID 0 (striping)	2	8
RAID 1 (mirroring)	2	4
RAID 4 (striping with dedicated parity)	3	8
RAID 5 (striping with distributed parity)	3	8

Connection fault tolerance can be achieved by having multiple connection paths to each storage object in the RAID. For more information about configuring multipath I/O support before configuring a software RAID, see Chapter 5, *Managing Multipath I/O for Devices* (page 53).

6.1.8 RAID 5 Algorithms for Distributing Stripes and Parity

RAID 5 uses an algorithm to determine the layout of stripes and parity. The following table describes the algorithms.

Table 6.7 *RAID 5 Algorithms*

Algorithm	EVMS Type	Description
Left Asymmetric	1	Stripes are written in a round-robin fashion from the first to last member segment. The parity's position in the striping sequence moves in a round-robin fashion from last to first. For example: `sda1 sdb1 sdc1 sde1`

Algorithm	EVMS Type	Description

		`0 1 2 p`
		`3 4 p 5`
		`6 p 7 8`
		`p 9 10 11`
		`12 13 14 p`

Left Symmetric	2	This is the default setting and is considered the fastest method for large reads.
		Stripes wrap to follow the parity. The parity’s position in the striping sequence moves in a round-robin fashion from last to first. For example:
		`sda1 sdb1 sdc1 sde1`
		`0 1 2 p`
		`4 5 p 3`
		`8 p 6 7`
		`p 9 10 11`
		`12 13 14 p`

Right Asymmetric	3	Stripes are written in a round-robin fashion from the first to last member segment. The parity’s position in the striping sequence moves in a round-robin fashion from first to last. For example:
		`sda1 sdb1 sdc1 sde1`
		`p 0 1 2`
		`3 p 4 5`
		`6 7 p 8`
		`9 10 11 p`
		`p 12 13 14`

Right Symmetric	4	Stripes wrap to follow the parity. The parity’s position in the striping sequence moves in a round-robin fashion from first to last. For example:

Algorithm	EVMS Type	Description

		`sda1 sdb1 sdc1 sde1`
		`p 0 1 2`
		`5 p 3 4`
		`7 8 p 6`
		`9 10 11 p`
		`p 12 13 14`

For information about the layout of stripes and parity with each of these algorithms, see *Linux RAID-5 Algorithms* [`http://www.accs.com/p_and_p/RAID/LinuxRAID.html`].

6.1.9 Multi-Disk Plug-In for EVMS

The Multi-Disk (MD) plug-in supports creating software RAIDs 0 (striping), 1 (mirror), 4 (striping with dedicated parity), and 5 (striping with distributed parity). The MD plug-in to EVMS allows you to manage all of these MD features as “regions” with the Regions Manager.

6.1.10 Device Mapper Plug-In for EVMS

The Device Mapper plug-in supports the following features in the EVMS MD Region Manager:

- **Multipath I/O:** Connection fault tolerance and load balancing for connections between the server and disks where multiple paths are available. If you plan to use multipathing, you should configure MPIO for the devices that you plan to use in the RAID before configuring the RAID itself. For information, see Chapter 5, *Managing Multipath I/O for Devices* (page 53).

  IMPORTANT

  The EVMS interface manages multipathing under the MD Region Manager, which originally supported the `md multipath` functions. It uses the legacy `md` terminology in the interface and in naming of device nodes, but implements the storage objects with Device Mapper.

- **Linear RAID:** A linear concatenation of discontinuous areas of free space from the same or multiple storage devices. Areas can be of different sizes.
- **Snapshots:** Snapshots of a file system at a particular point in time, even while the system is active, thereby allowing a consistent backup.

The Device Mapper driver is not started by default in the rescue system.

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Start the Device Mapper by entering the following at the terminal console prompt:

```
/etc/init.d/boot.device-mapper start
```

6.2 Creating and Configuring a Software RAID

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Start the EVMS GUI by entering the following at the terminal console prompt:

```
evmsgui
```

3 If the disks have not been initialized, initialize them by adding the DOS Segment Manager now.

The following instructions assume you are initializing new disks. For information about initializing an existing disk or a disk moved from another system, see Section 4.2, “Initializing Disks” (page 43).

Repeat the following steps for each disk that you want to initialize:

3a Select *Actions > Add > Segment Manager to Storage Object.*

3b From the list, select the *DOS Segment Manager*, then click *Next.*

3c Select the device, then click *Add* to initialize it.

4 If segments have not been created on the disks, create a segment on each disk that you plan to use in the RAID.

For x86 platforms, this step is optional if you treat the entire disk as one segment.

For IA-64 platforms, this step is necessary to make the *RAID 4/5* option available in the Regions Manager.

For information about creating segments, see Section 4.4, “Creating Disk Segments (or Partitions)” (page 47).

4a Select *Action* > *Create* > *Segment* to open the *DOS Segment Manager*.

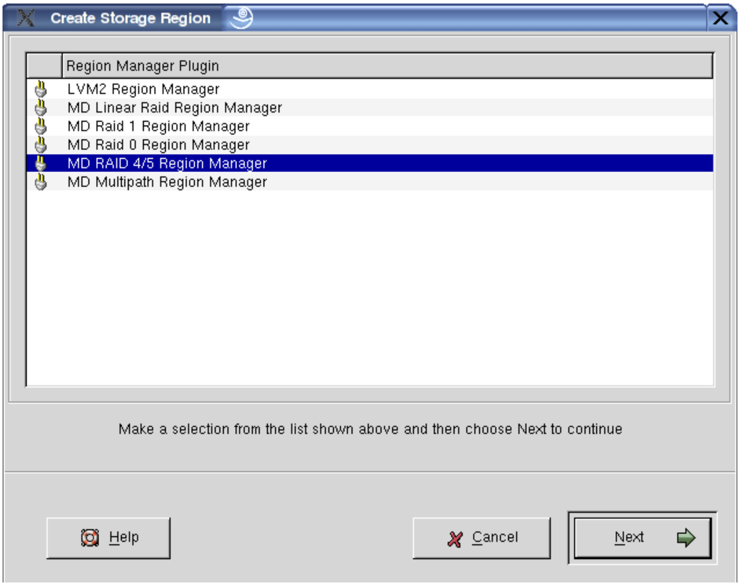
4b Select the free space segment you want to use.

4c Specify the amount of space to use for the segment.

4d Specify the segment options, then click *Create*.

5 Create and configure a software RAID Device.

5a Select *Action* > *Create* > *Region* to open the Create Storage Region dialog box.


Create Storage Region
Region Manager Plugin
LVM2 Region Manager
MD Linear Raid Region Manager
MD Raid 1 Region Manager
MD Raid 0 Region Manager
MD RAID 4/5 Region Manager
MD Multipath Region Manager
Make a selection from the list shown above and then choose Next to continue
Help
Cancel
Next

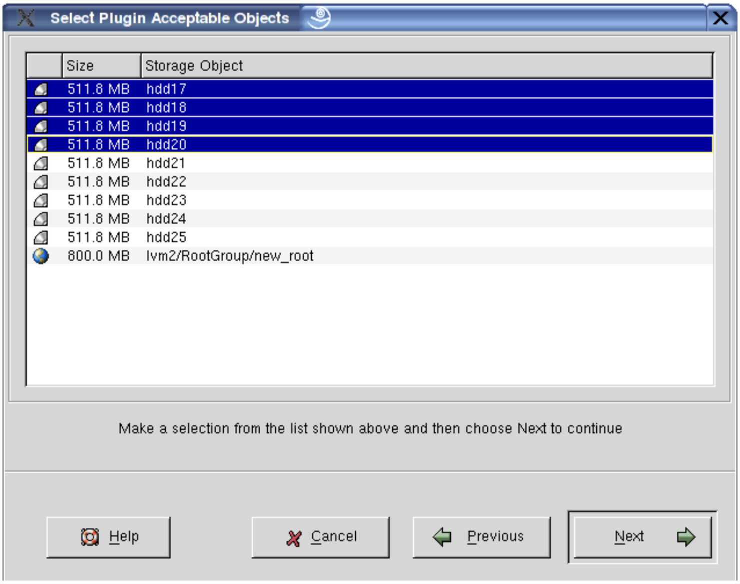
5b Specify the type of software RAID you want to create by selecting one of the following Region Managers, then click *Next*.

- *MD RAID 0 Region Manager*
- *MD RAID 1 Region Manager*
- *MD RAID 4/5 Region Manager*

5c From the Storage Objects listed, select the ones to use for the RAID device.

IMPORTANT

The order of the objects in the RAID is implied by their order in the list.



Size	Storage Object
511.8 MB	hdd17
511.8 MB	hdd18
511.8 MB	hdd19
511.8 MB	hdd20
511.8 MB	hdd21
511.8 MB	hdd22
511.8 MB	hdd23
511.8 MB	hdd24
511.8 MB	hdd25
800.0 MB	lvm2/RootGroup/new_root

Make a selection from the list shown above and then choose Next to continue

5d Specify values for *Configuration Options* by changing the following default settings as desired.

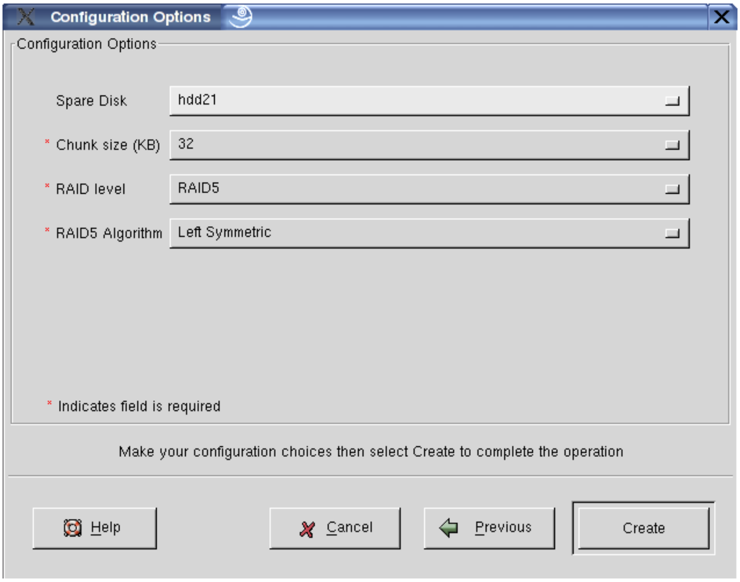
- For RAIDs 1, 4, or 5, optionally specify a device to use as the spare disk for the RAID. The default is none.

- For RAIDs 0, 4, or 5, specify the chunk (stripe) size in KB. The default is 32 KB.

- For RAIDs 4/5, specify *RAID 4* or *RAID 5* (default).

- For RAID 5, specify the algorithm to use for striping and parity. The default is *Left Symmetric*.

For information about these settings, see Section 6.1.5, “Configuration Options for RAIDs” (page 116).



Configuration Options

Spare Disk: hdd21

* Chunk size (KB): 32

* RAID level: RAID5

* RAID5 Algorithm: Left Symmetric

* Indicates field is required

Make your configuration choices then select Create to complete the operation

5e Click *Create* to create the RAID device under the `/dev/evms/md` directory.

The device is given a name such as `md0`, so its EVMS mount location is `/dev/evms/md/md0`.

6 Specify a human-readable label for the device.

6a Select *Action* > *Create* > *EVMS Volume or Compatible Volume*.

6b Select the device that you created in Step 5 (page 123).

6c Specify a name for the device.

Use standard ASCII characters and naming conventions. Spaces are allowed.

6d Click *Done*.

7 Create a file system on the RAID device you created.

7a Select *Action* > *File System* > *Make* to view a list of file system modules.

7b Select the type of file system you want to create, such as the following:

- *ReiserFS File System Module*
- *Ext2/3FS File System Module*

7c Select the RAID device you created in Step 5 (page 123), such as `/dev/evms/md/md0`.

7d Specify a name to use as the *Volume Label*, then click *Make*.

The name must not contain space or it will fail to mount later.

7e Click *Save* to create the file system.

8 Mount the RAID device.

8a Select *Action* > *File System* > *Mount*.

8b Select the RAID device you created in Step 5 (page 123), such as `/dev/evms/md/md0`.

8c Specify the location where you want to mount the device, such as `/home`.

8d Click *Mount*.

9 Enable `boot.evms` to activate EVMS automatically at reboot.

9a In YaST, select *System* > *System Services (Run Level)*.

9b Select *Expert Mode*.

9c Select *Boot.evms*.

9d Select *Set/Reset*.

9e Select *Enable the Service*.

10 Edit the `/etc/fstab` file to automount the RAID mount point created in Step 8c (page 127), or you can mount the device manually from `evmsgui`.

6.3 Expanding a RAID

This section explains how to expand a RAID by adding segments to it.

IMPORTANT

Before you can expand the size of a RAID device, you must deactivate it.

- Section 6.3.1, "Adding Mirrors to a RAID 1 Device" (page 128)
- Section 6.3.2, "Adding Segments to a RAID 4 or 5" (page 129)

6.3.1 Adding Mirrors to a RAID 1 Device

In a RAID 1 device, each member segment contains its own copy of all of the data stored in the RAID. You can add a mirror to the RAID to increase redundancy. The segment must be at least the same size as the smallest member segment in the existing RAID 1 device. Any excess space in the segment is not used. Ideally, all member segments of a RAID 1 device are the same size.

Adding an Available Segment as the New Mirror

1. Deactivate the RAID 1 device.
2. Use the Add Active (`addactive` plug-in) function.
3. From the list of available segments, select one that is the same size or larger than the smallest existing member of the RAID device.

4. Reactivate the RAID device.

Activating A Spare Disk as the New Mirror

1. If you have not set up a spare disk, do it now.

   For information, see Section 6.4, “Adding or Removing a Spare Disk” (page 129).

2. Use the Activate Spare (`activatespare` plug-in) function to add it to the RAID 1 device as a new mirror.

6.3.2 Adding Segments to a RAID 4 or 5

If the RAID region is clean and operating normally, the kernel driver adds the new object as a regular spare, and it acts as a hot standby for future failures. If the RAID region is currently degraded, the kernel driver immediately activates the new spare object and begins synchronizing the data and parity information.

6.4 Adding or Removing a Spare Disk

The MD driver allows you to optionally designate a spare disk (device, segment, or region) for RAID 1, 4, and 5 devices. You can assign a spare disk when you create the RAID or at any time thereafter. The RAID can be active and in use when you add or remove the spare. The spare is activated for the RAID only on disk failure.

- Section 6.4.1, “Do You Need a Spare Disk?” (page 130)
- Section 6.4.2, “Adding a Spare Disk When You Create the RAID ” (page 130)
- Section 6.4.3, “Adding a Spare Disk to an Existing RAID” (page 131)
- Section 6.4.4, “Removing a Spare Disk from a RAID” (page 131)

6.4.1 Do You Need a Spare Disk?

The advantage of specifying a spare disk for a RAID is that the system monitors the failure and begins recovery without human interaction. The disadvantage is that the space on the spare disk is not available until it is activated by a failed RAID.

As noted in Section 6.1.2, “Overview of RAID Levels” (page 113), RAIDs 1, 4, and 5 can tolerate at least one disk failure. Any given RAID can have one spare disk designated for it, but the spare itself can serve as the designated spare for one RAID, for multiple RAIDs, or for all arrays. The spare disk is a hot standby until it is needed. It is not an active member of any RAIDs where it is assigned as the spare disk until it is activated for that purpose.

If a spare disk is defined for the RAID, the RAID automatically deactivates the failed disk and activates the spare disk on disk failure. The MD driver then begins synchronizing mirrored data for a RAID 1 or reconstructing the missing data and parity information for RAIDs 4 and 5. The I/O performance remains in a degraded state until the failed disk’s data is fully remirrored or reconstructed.

Creating a spare-group name allows a single hot spare to service multiple RAID arrays. The spare-group name can be any character string, but must be uniquely named for the server. For `mdadm` to move spares from one array to another, the different arrays must be labelled with the same spare-group name in the configuration file.

For example, when `mdadm` detects that an array is missing a component device, it first checks to see if the array has a spare device. If no spare is available, mdadm looks in the array’s assigned spare-group for another array that has a full complement of working drives and a spare. It attempts to remove the spare from the working array and add it to the degraded array. If the removal succeeds but the adding fails, then the spare is added back to its source array.

6.4.2 Adding a Spare Disk When You Create the RAID

When you create a RAID 1, 4, or 5 in EVMS, specify the *Spare Disk* in the *Configuration Options* dialog box. You can browse to select the available device, segment, or region that you want to be the RAID’s spare disk. For information, see Step 5d (page 125) in Section 6.2, “Creating and Configuring a Software RAID” (page 122).

6.4.3 Adding a Spare Disk to an Existing RAID

The RAID 1, 4, or 5 device can be active and in use when you add a spare disk to it. If the RAID is operating normally, the specified disk is added as a spare and it acts as a hot standby for future failures. If the RAID is currently degraded because of a failed disk, the specified disk is added as a spare disk, then it is automatically activated as a replacement disk for the failed disk, and it begins synchronizing the data and parity information.

1 Prepare a disk, segment, or region to use as the replacement disk, just as you did for the component devices of the RAID device.

2 In EVMS, select the *ActionsAddSpare Disk to a Region* (the `addspare` plug-in for the EVMS GUI).

3 Select the RAID device you want to manage from the list of Regions, then click *Next*.

4 Select the device to use as the spare disk.

5 Click *Add*.

6.4.4 Removing a Spare Disk from a RAID

The RAID 1, 4, or 5 device can be active and in use when you remove its spare disk.

1 In EVMS, select the *ActionsRemoveSpare Disk from a Region* (the `remspare` plug-in for the EVMS GUI).

2 Select the RAID device you want to manage from the list of Regions, then click *Next*.

3 Select the spare disk.

4 Click *Remove*.

6.5 Managing Disk Failure and RAID Recovery

6.5.1 Understanding the Disk Failure and RAID Recovery

RAIDs 1, 4, and 5 can survive a disk failure. A RAID 1 device survives if all but one mirrored array fails. Its read performance is degraded without the multiple data sources available, but its write performance might actually improve when it does not write to the failed mirrors. During the synchronization of the replacement disk, write and read performance are both degraded. A RAID 5 can survive a single disk failure at a time. A RAID 4 can survive a single disk failure at a time if the disk is not the parity disk.

Disks can fail for many reasons such as the following:

- Disk crash
- Disk pulled from the system
- Drive cable removed or loose
- I/O errors

When a disk fails, the RAID removes the failed disk from membership in the RAID, and operates in a degraded mode until the failed disk is replaced by a spare. Degraded mode is resolved for a single disk failure in one of the following ways:

- **Spare Exists:** If the RAID has been assigned a spare disk, the MD driver automatically activates the spare disk as a member of the RAID, then the RAID begins synchronizing (RAID 1) or reconstructing (RAID 4 or 5) the missing data.

- **No Spare Exists:** If the RAID does not have a spare disk, the RAID operates in degraded mode until you configure and add a spare. When you add the spare, the MD driver detects the RAID's degraded mode, automatically activates the spare as a member of the RAID, then begins synchronizing (RAID 1) or reconstructing (RAID 4 or 5) the missing data.

6.5.2 Identifying the Failed Drive

On failure, `md` automatically removes the failed drive as a component device in the RAID array. To determine which device is a problem, use `mdadm` and look for the device that has been reported as "removed".

- Enter the following a a terminal console prompt

  ```
  mdadm -D /dev/md1
  ```

  Replace `/dev/md1` with the actual path for your RAID.

For example, an `mdadm` report for a RAID 1 device consisting of `/dev/sda2` and `/dev/sdb2` might look like this:

```
blue6:~ # mdadm -D /dev/md1
/dev/md1:
        Version : 00.90.03
  Creation Time : Sun Jul  2 01:14:07 2006
     Raid Level : raid1
     Array Size : 180201024 (171.85 GiB 184.53 GB)
    Device Size : 180201024 (171.85 GiB 184.53 GB)
   Raid Devices : 2
  Total Devices : 1
Preferred Minor : 1
    Persistence : Superblock is persistent
    Update Time : Tue Aug 15 18:31:09 2006
          State : clean, degraded
 Active Devices : 1
Working Devices : 1
 Failed Devices : 0
```

```
Spare Devices : 0

         UUID : 8a9f3d46:3ec09d23:86e1ffbc:ee2d0dd8

       Events : 0.174164

  Number   Major   Minor   RaidDevice State

     0       0       0        0      removed

     1       8      18        1      active sync   /dev/sdb2
```

The “Total Devices : 1”, “Active Devices : 1”, and “Working Devices : 1” indicate that only one of the two devices is currently active. The RAID is operating in a “degraded” state.

The “Failed Devices : 0” might be confusing. This setting has a non-zero number only for that brief period where the `md` driver finds a problem on the drive and prepares to remove it from the RAID. When the failed drive is removed, it reads “0” again.

In the devices list at the end of the report, the device with the “removed” state for Device 0 indicates that the device has been removed from the software RAID definition, not that the device has been physically removed from the system. It does not specifically identify the failed device. However, the working device (or devices) are listed. Hopefully, you have a record of which devices were members of the RAID. By the process of elimination, the failed device is `/dev/sda2`.

The “Spare Devices : 0” indicates that you do not have a spare assigned to the RAID. You must assign a spare device to the RAID so that it can be automatically added to the array and replace the failed device.

6.5.3 Replacing a Failed Device with a Spare

When a component device fails, the `md` driver replaces the failed device with a spare device assigned to the RAID. You can either keep a spare device assigned to the RAID as a hot standby to use as an automatic replacement, or assign a spare device to the RAID as needed.

IMPORTANT

Even if you correct the problem that caused the problem disk to fail, the RAID does not automatically accept it back into the array because it is a “faulty object” in the RAID and is no longer synchronized with the RAID.

If a spare is available, md automatically removes the failed disk, replaces it with the spare disk, then begins to synchronize the data (for RAID 1) or reconstruct the data from parity (for RAIDs 4 or 5).

If a spare is not available, the RAID operates in degraded mode until you assign spare device to the RAID.

To assign a spare device to the RAID:

1 Prepare the disk as needed to match the other members of the RAID.

2 In EVMS, select the *ActionsAddSpare Disk to a Region* (the `addspare` plug-in for the EVMS GUI).

3 Select the RAID device you want to manage from the list of Regions, then click *Next*.

4 Select the device to use as the spare disk.

5 Click *Add*.

   The `md` driver automatically begins the replacement and reconstruction or synchronization process.

6 Monitor the status of the RAID to verify the process has begun.

   For information about how monitor RAID status, see Section 6.6, “Monitoring Status for a RAID” (page 136).

7 Continue with Section 6.5.4, “Removing the Failed Disk” (page 135).

6.5.4 Removing the Failed Disk

You can remove the failed disk at any time after it has been replaced with the spare disk. EVMS does not make the device available for other use until you remove it from the RAID. After you remove it, the disk appears in the *Available-Objects* list in the EVMS GUI, where it can be used for any purpose.

NOTE

If you pull a disk or if it is totally unusable, EVMS no longer recognizes the failed disk as part of the RAID.

The RAID device can be active and in use when you remove its faulty object.

1 In EVMS, select the *ActionsRemoveFaulty Object from a Region* (the `remfaulty` plug-in the EVMS GUI).

2 Select the RAID device you want to manage from the list of Regions, then click *Next*.

3 Select the failed disk.

4 Click *Remove*.

6.6 Monitoring Status for a RAID

- Section 6.6.1, “Monitoring Status with EVMSGUI” (page 136)
- Section 6.6.2, “Monitoring Status with /proc/mdstat” (page 137)
- Section 6.6.3, “Monitoring Status with mdadm” (page 138)
- Section 6.6.4, “Monitoring a Remirror or Reconstruction” (page 141)
- Section 6.6.5, “Configuring mdadm to Send an E-Mail Alert for RAID Events” (page 141)

6.6.1 Monitoring Status with EVMSGUI

The *Regions* tab in EVMS GUI (`evmsgui`) reports any software RAID devices that are defined and whether they are currently active.

6.6.2 Monitoring Status with /proc/mdstat

A summary of RAID and status information (active/not active) is available in the /proc/mdstat file.

1 Open a terminal console, then log in as the root user or equivalent.

2 View the /proc/mdstat file by entering the following at the console prompt:

```
cat /proc/mdstat
```

3 Evaluate the information.

The following table shows an example output and how to interpret the information.

Status Information	Description	Interpretation
Personalities : [raid5] [raid4]	List of the RAIDs on the server by RAID label.	You have two RAIDs defined with labels of raid5 and raid4.
md0 : active raid5 sdg1[0] sdk1[4] sdj1[3] sdi1[2]	<device> : <active | not active> <RAID label you specified> < storage object> [RAID order]	The RAID is active and mounted at /dev/evms/md/md0. The RAID label is raid5. The active segments are sdg1, sdi1, sdj1, and sdk1, as ordered in the RAID. The RAID numbering of 0 to 4 indicates that the RAID has 5 segments, and the second segment [1] is missing from the list. Based on the segment names, the missing segment is sdh1.

Status Information	Description	Interpretation
`35535360 blocks level 5, 128k chunk, algorithm 2 [5/4] [U_UUU]`	<number of blocks> blocks level < 0 | 1 | 4 | 5 > <stripe size in KB> chunk algorithm <1 | 2 | 3 | 4 > [number of devices/number of working devices] [U-UUU]	If the block size on the server is 4 KB, the total size of the RAID (including parity) is 142 GB, with a data capacity of 113.7 GB. The stripe size is 128 KB. The RAID is using left symmetric. algorithm <1 | 2 | 3 | 4 > [number of devices/number of working devices] [U-UUU]
`unused devices: <none>`	All segments in the RAID are in use.	There are no spare devices available on the server.

6.6.3 Monitoring Status with mdadm

To view the RAID status with the `mdadm` command, enter the following at a terminal prompt:

```
mdadm -D /dev/mdx
```

Replace *`mdx`* with the RAID device number.

Example 1: A Disk Fails

In the following example, only four of the five devices in the RAID are active (`Raid Devices : 5`,`Total Devices : 4`). When it was created, the component devices in the device were numbered 0 to 5 and are ordered according to their alphabetic appearance in the list where they were chosen, such as `/dev/sdg1`, `/dev/sdh1`, `/dev/sdi1`, `/dev/sdj1`, and `/dev/sdk1`. From the pattern of filenames of the other devices, you determine that the device that was removed was named `/dev/sdh1`.

```
/dev/md0:
Version : 00.90.03
Creation Time : Sun Apr 16 11:37:05 2006
Raid Level : raid5
Array Size : 35535360 (33.89 GiB 36.39 GB)
Device Size : 8883840 (8.47 GiB 9.10 GB)
Raid Devices : 5
Total Devices : 4
Preferred Minor : 0
Persistence : Superblock is persistent
Update Time : Mon Apr 17 05:50:44 2006
State : clean, degraded
Active Devices : 4
Working Devices : 4
Failed Devices : 0
Spare Devices : 0
Layout : left-symmetric
Chunk Size : 128K
UUID : 2e686e87:1eb36d02:d3914df8:db197afe
Events : 0.189
Number Major Minor RaidDevice State
0 8 97 0 active sync /dev/sdg1
1 8 0 1 removed
2 8 129 2 active sync /dev/sdi1
3 8 45 3 active sync /dev/sdj1
4 8 161 4 active sync /dev/sdk1
```

Example 2: Spare Disk Replaces the Failed Disk

In the following `mdadm` report, only 4 of the 5 disks are active and in good condition (`Active Devices : 4`, `Working Devices : 5`). The failed disk was automatically detected and removed from the RAID (`Failed Devices: 0`). The spare was activated as the replacement disk, and has assumed the diskname of the failed disk (`/dev/sdh1`). The faulty object (the failed disk that was removed from the RAID) is not identified in the report. The RAID is running in degraded mode (`State : clean, degraded, recovering`). The data is being rebuilt (`spare rebuilding`

`/dev/sdh1`), and the process is 3% complete (`Rebuild Status : 3% complete`).

```
mdadm -D /dev/md0
/dev/md0:
Version : 00.90.03
Creation Time : Sun Apr 16 11:37:05 2006
Raid Level : raid5
Array Size : 35535360 (33.89 GiB 36.39 GB)
Device Size : 8883840 (8.47 GiB 9.10 GB)
Raid Devices : 5
Total Devices : 5
Preferred Minor : 0
Persistence : Superblock is persistent
Update Time : Mon Apr 17 05:50:44 2006
State : clean, degraded, recovering
Active Devices : 4
Working Devices : 5
Failed Devices : 0
Spare Devices : 1
Layout : left-symmetric
Chunk Size : 128K
Rebuild Status : 3% complete
UUID : 2e686e87:1eb36d02:d3914df8:db197afe
Events : 0.189
Number Major Minor RaidDevice State
0 8 97 0 active sync /dev/sdg1
1 8 113 1 spare rebuilding /dev/sdh1
2 8 129 2 active sync /dev/sdi1
3 8 145 3 active sync /dev/sdj1
4 8 161 4 active sync /dev/sdk1
```

6.6.4 Monitoring a Remirror or Reconstruction

You can follow the progress of the synchronization or reconstruction process by examining the `/proc/mdstat` file.

You can control the speed of synchronization by setting parameters in the `/proc/sys/dev/raid/speed_limit_min` and `/proc/sys/dev/raid/speed_limit_max` files. To speed up the process, echo a larger number into the `speed_limit_min` file.

6.6.5 Configuring mdadm to Send an E-Mail Alert for RAID Events

You might want to configure the `mdadm` service to send an e-mail alert for software RAID events. Monitoring is only meaningful for RAIDs 1, 4, 5, 6, 10 or multipath arrays because only these have missing, spare, or failed drives to monitor. RAID 0 and Linear RAIDs do not provide fault tolerance so they have no interesting states to monitor.

The following table identifies RAID events and indicates which events trigger e-mail alerts. All events cause the program to run. The program is run with two or three arguments: the event name, the array device (such as `/dev/md1`), and possibly a second device. For Fail, Fail Spare, and Spare Active, the second device is the relevant component device. For MoveSpare, the second device is the array that the spare was moved from.

Table 6.8 *RAID Events in mdadm*

RAID Event	Trigger E-Mail Alert	Description
Device Disappeared	No	An `md` array that was previously configured appears to no longer be configured. (syslog priority: Critical)

RAID Event	Trigger E-Mail Alert	Description
		If mdadm was told to monitor an array which is RAID0 or Linear, then it reports DeviceDisappeared with the extra information Wrong-Level. This is because RAID0 and Linear do not support the device-failed, hot-spare, and resynchronize operations that are monitored.
Rebuild Started	No	An `md` array started reconstruction. (syslog priority: Warning)
Rebuild NN	No	Where NN is 20, 40, 60, or 80. This indicates the percent completed for the rebuild. (syslog priority: Warning)
Rebuild Finished	No	An `md` array that was rebuilding is no longer rebuilding, either because it finished normally or was aborted. (syslog priority: Warning)
Fail	Yes	An active component device of an array has been marked as faulty. (syslog priority: Critical)
Fail Spare	Yes	A spare component device that was being rebuilt to replace a faulty device has failed. (syslog priority: Critical)
Spare Active	No	A spare component device that was being rebuilt to replace a faulty device has been successfully rebuilt and has been made active. (syslog priority: Info)
New Array	No	A new `md` array has been detected in the `/proc/mdstat` file. (syslog priority: Info)
Degraded Array	Yes	A newly noticed array appears to be degraded. This message is not generated when `mdadm` notices a drive failure that causes degradation. It is generated only when

RAID Event	Trigger E-Mail Alert	Description
		mdadm notices that an array is degraded when it first sees the array. (syslog priority: Critical)
Move Spare	No	A spare drive has been moved from one array in a spare group to another to allow a failed drive to be replaced. (syslog priority: Info)
Spares Missing	Yes	The mdadm.conffile indicates that an array should have a certain number of spare devices, but mdadm detects that the array has fewer than this number when it first sees the array. (syslog priority: Warning)
Test Message	Yes	An array was found at startup, and the --test flag was given. (syslog priority: Info)

To configure an e-mail alert:

1 At a terminal console, log in as the root user.

2 Edit the /etc/mdadm/mdadm.conf file to add your e-mail address for receiving alerts. For example, specify the MAILADDR value (using your own e-mail address, of course):

```
DEVICE partitions

ARRAY /dev/md0 level=raid1 num-devices=2

     UUID=1c661ae4:818165c3:3f7a4661:af475fda

     devices=/dev/sdb3,/dev/sdc3

MAILADDR yourname@example.com
```

The MAILADDR line gives an e-mail address that alerts should be sent to when mdadm is running in --monitor mode with the --scan option. There should be only one MAILADDR line in mdadm.conf, and it should have only one address.

3 Start mdadm monitoring by entering the following at the terminal console prompt:

```
mdadm --monitor --mail=yourname@example.com --delay=1800 /dev/md0
```

The --monitor option causes mdadm to periodically poll a number of md arrays and to report on any events noticed. mdadm never exits once it decides that there are arrays to be checked, so it should normally be run in the background.

In addition to reporting events in this mode, mdadm might move a spare drive from one array to another if they are in the same spare-group and if the destination array has a failed drive but no spares.

Listing the devices to monitor is optional. If any devices are listed on the command line, mdadm monitors only those devices. Otherwise, all arrays listed in the configuration file are monitored. Further, if --scan option is added in the command, then any other md devices that appear in/proc/mdstat are also monitored.

For more information about using mdadm, see the mdadm(8) and mdadm.conf(5) man pages.

4 To configure the /etc/init.d/mdadmd service as a script:

```
suse:~ # egrep 'MAIL|RAIDDEVICE' /etc/sysconfig/mdadm

MDADM_MAIL="yourname@example.com"

MDADM_RAIDDEVICES="/dev/md0"

MDADM_SEND_MAIL_ON_START=no

suse:~ # chkconfig mdadmd --list

mdadmd      0:off  1:off  2:off  3:on   4:off  5:on 6:off
```

6.7 Deleting a Software RAID and Its Data

If you want to remove the prior multipath settings, deactivate the RAID, delete the data on the RAID, and release all resources used by the RAID, do the following:

1. If you want to keep the data stored on the software RAID device, make sure to back up the data to alternate media, using your normal backup procedures. Make sure the backup is good before proceeding.

2. Open a terminal console prompt as the `root` user or equivalent. Use this console to enter the commands described in the remaining steps.

3. Dismount the software RAID device by entering

   ```
   umount <raid-device>
   ```

4. Stop the RAID device and its component devices by entering

   ```
   mdadm --stop <raid-device>
   ```

   ```
   mdadm --stop <member-devices>
   ```

   For more information about using `mdadm`, please see the `mdadm(8)` man page.

5. Delete all data on the disk by literally overwriting the entire device with zeroes. Enter

   ```
   mdadm --misc --zero-superblock <member-devices>
   ```

6. You must now reinitialize the disks for other uses, just as you would when adding a new disk to your system.

Managing Software RAIDs 6 and 10 with mdadm

7

This section describes how to create software RAID 6 and 10 devices, using the Multiple Devices Administration (`mdadm(8)`) tool. You can also use `mdadm` to create RAIDs 0, 1, 4, and 5. The `mdadm` tool provides the functionality of legacy programs `mdtools` and `raidtools`.

7.1 Creating a RAID 6

7.1.1 Understanding RAID 6

RAID 6 is essentially an extension of RAID 5 that allows for additional fault tolerance by using a second independent distributed parity scheme (dual parity). Even if one of the hard disk drives fails during the data recovery process, the system continues to be operational, with no data loss.

RAID6 provides for extremely high data fault tolerance by sustaining multiple simultaneous drive failures. It handles the loss of any two devices without data loss. Accordingly, it requires N+2 drives to store N drives worth of data. It requires a minimum of 4 devices.

The performance for RAID 6 is slightly lower but comparable to RAID 5 in normal mode and single disk failure mode. It is very slow in dual disk failure mode.

Table 7.1 *Comparison of RAID 5 and RAID 6*

Feature	RAID 5	RAID 6
Number of devices	N+1, minimum of 3	N+2, minimum of 4
Parity	Distributed, single	Distributed, dual
Performance	Medium impact on write and rebuild	More impact on sequential write than RAID 5
Fault-tolerance	Failure of one component device	Failure of two component devices

7.1.2 Creating a RAID 6

The procedure in this section creates a RAID 6 device `/dev/md0` with four devices: `/dev/sda1`, `/dev/sdb1`, `/dev/sdc1`, and `/dev/sdd1`. Make sure to modify the procedure to use your actual device nodes.

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Create a RAID 6 device. At the command prompt, enter

```
mdadm --create /dev/md0 --run --level=raid6 --chunk=128 --raid-devices=4
/dev/sdb1 /dev/sdc1 /dev/sdc1 /dev/sdd1
```

The default chunk size is 64 (KB).

3 Create a file system on the RAID 6 device `/dev/md0`, such as a Reiser file system (reiserfs). For example, at the command prompt, enter

```
mkfs.reiserfs /dev/md0
```

Modify the command if you want to use a different file system.

4 Edit the `/etc/mdadm.conf` file to add entries for the component devices and the RAID device `/dev/md0`.

5 Edit the `/etc/fstab` file to add an entry for the RAID 6 device `/dev/md0`.

6 Reboot the server.

The RAID 6 device is mounted to `/local`.

7 (Optional) Add a hot spare to service the RAID array. For example, at the command prompt enter:

```
mdadm /dev/md0 -a /dev/sde1
```

7.2 Creating Nested RAID 10 Devices with mdadm

- Section 7.2.1, "Understanding Nested RAID Devices" (page 149)
- Section 7.2.2, "Creating Nested RAID 10 (1+0) with mdadm" (page 151)
- Section 7.2.3, "Creating Nested RAID 10 (0+1) with mdadm" (page 153)

7.2.1 Understanding Nested RAID Devices

A nested RAID device consists of a RAID array that uses another RAID array as its basic element, instead of using physical disks. The goal of this configuration is to improve the performance and fault tolerance of the RAID.

Linux supports nesting of RAID 1 (mirroring) and RAID 0 (striping) arrays. Generally, this combination is referred to as RAID 10. To distinguish the order of the nesting, this document uses the following terminology:

- **RAID 1+0:** RAID 1 (mirror) arrays are built first, then combined to form a RAID 0 (stripe) array.
- **RAID 0+1:** RAID 0 (stripe) arrays are built first, then combined to form a RAID 1 (mirror) array.

The following table describes the advantages and disadvantages of RAID 10 nesting as 1+0 versus 0+1. It assumes that the storage objects you use reside on different disks, each with a dedicated I/O capability.

Table 7.2 *RAID Levels Supported in EVMS*

RAID Level	Description	Performance and Fault Tolerance
10 (1+0)	RAID 0 (stripe) built with RAID 1 (mirror) arrays	RAID 1+0 provides high levels of I/O performance, data redundancy, and disk fault tolerance. Because each member device in the RAID 0 is mirrored individually, multiple disk failures can be tolerated and data remains available as long as the disks that fail are in different mirrors. You can optionally configure a spare for each underlying mirrored array, or configure a spare to serve a spare group that serves all mirrors.
10 (0+1)	RAID 1 (mirror) built with RAID 0 (stripe) arrays	RAID 0+1 provides high levels of I/O performance and data redundancy, but slightly less fault tolerance than a 1+0. If multiple disks fail on one side of the mirror, then the other mirror is available. However, if disks are lost concurrently on both sides of the mirror, all data is lost. This solution offers less disk fault tolerance than a 1+0 solution, but if you need to perform maintenance or maintain the mirror on a different site, you can take an entire side of the mirror offline and still have a fully functional storage device. Also, if you lose the connection

RAID Level	Description	Performance and Fault Tolerance
		between the two sites, either site operates independently of the other. That is not true if you stripe the mirrored segments, because the mirrors are managed at a lower level. If a device fails, the mirror on that side fails because RAID 1 is not fault-tolerant. Create a new RAID 0 to replace the failed side, then resynchronize the mirrors.

7.2.2 Creating Nested RAID 10 (1+0) with mdadm

A nested RAID 1+0 is built by creating two or more RAID 1 (mirror) devices, then using them as component devices in a RAID 0.

IMPORTANT

If you need to manage multiple connections to the devices, you must configure multipath I/O before configuring the RAID devices. For information, see Chapter 5, *Managing Multipath I/O for Devices* (page 53).

The procedure in this section uses the device names shown in the following table. Make sure to modify the device names with the names of your own devices.

Table 7.3 *Scenario for Creating a RAID 10 (1+0) by Nesting*

Raw Devices	RAID 1 (mirror)	RAID 1+0 (striped mirrors)
`/dev/sdb1`	`/dev/md0`	`/dev/md2`
`/dev/sdc1`		

Raw Devices	RAID 1 (mirror)	RAID 1+0 (striped mirrors)
/dev/sdd1	/dev/md1	
/dev/sde1		

1 Open a terminal console, then log in as the root user or equivalent.

2 Create 2 software RAID 1 devices, using two different devices for each RAID 1 device. At the command prompt, enter these two commands:

```
mdadm --create /dev/md0 --run --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
```

```
mdadm --create /dev/md1 --run --level=1 --raid-devices=2 /dev/sdd1 /dev/sde1
```

3 Create the nested RAID 1+0 device. At the command prompt, enter the following command using the software RAID 1 devices you created in Step 2 (page 152):

```
mdadm --create /dev/md2 --run --level=0 --chunk=64 --raid-devices=2 /dev/md0
 /dev/md1
```

The default chunk size is 64 KB.

4 Create a file system on the RAID 1+0 device /dev/md2, such as a Reiser file system (reiserfs). For example, at the command prompt, enter

```
mkfs.reiserfs /dev/md2
```

Modify the command if you want to use a different file system.

5 Edit the /etc/mdadm.conf file to add entries for the component devices and the RAID device /dev/md2.

6 Edit the /etc/fstab file to add an entry for the RAID 1+0 device /dev/md2.

7 Reboot the server.

The RAID 1+0 device is mounted to /local.

8 (Optional) Add hot spares to service the underlying RAID 1 mirrors.

For information, see Section 6.4, “Adding or Removing a Spare Disk” (page 129).

7.2.3 Creating Nested RAID 10 (0+1) with mdadm

A nested RAID 0+1 is built by creating two to four RAID 0 (striping) devices, then mirroring them as component devices in a RAID 1.

IMPORTANT

If you need to manage multiple connections to the devices, you must configure multipath I/O before configuring the RAID devices. For information, see Chapter 5, *Managing Multipath I/O for Devices* (page 53).

In this configuration, spare devices cannot be specified for the underlying RAID 0 devices because RAID 0 cannot tolerate a device loss. If a device fails on one side of the mirror, you must create a replacement RAID 0 device, than add it into the mirror.

The procedure in this section uses the device names shown in the following table. Make sure to modify the device names with the names of your own devices.

Table 7.4 *Scenario for Creating a RAID 10 (0+1) by Nesting*

Raw Devices	RAID 0 (stripe)	RAID 0+1 (mirrored stripes)
`/dev/sdb1`	`/dev/md0`	`/dev/md2`
`/dev/sdc1`		
`/dev/sdd1`	`/dev/md1`	
`/dev/sde1`		

1 Open a terminal console, then log in as the root user or equivalent.

2 Create 2 software RAID 0 devices, using two different devices for each RAID 0 device. At the command prompt, enter these two commands:

```
mdadm --create /dev/md0 --run --level=0 --chunk=64 --raid-devices=2 /dev/sdb1
 /dev/sdc1
```

```
mdadm --create /dev/md1 --run --level=0 --chunk=64 --raid-devices=2 /dev/sdd1
 /dev/sde1
```

The default chunk size is 64 KB.

3 Create the nested RAID 0+1 device. At the command prompt, enter the following command using the software RAID 0 devices you created in Step 2 (page 153):

```
mdadm --create /dev/md2 --run --level=1 --raid-devices=2 /dev/md0 /dev/md1
```

4 Create a file system on the RAID 0+1 device `/dev/md2`, such as a Reiser file system (reiserfs). For example, at the command prompt, enter

```
mkfs.reiserfs /dev/md2
```

Modify the command if you want to use a different file system.

5 Edit the `/etc/mdadm.conf` file to add entries for the component devices and the RAID device `/dev/md2`.

6 Edit the `/etc/fstab` file to add an entry for the RAID 0+1 device `/dev/md2`.

7 Reboot the server.

The RAID 0+1 device is mounted to `/local`.

7.3 Creating a Complex RAID 10 with mdadm

- Section 7.3.1, “Understanding the mdadm RAID10” (page 155)
- Section 7.3.2, “Creating a RAID10 with mdadm” (page 158)

7.3.1 Understanding the mdadm RAID10

In `mdadm`, the RAID10 level creates a single complex software RAID that combines features of both RAID 0 (striping) and RAID 1 (mirroring). Multiple copies of all data blocks are arranged on multiple drives following a striping discipline. Component devices should be the same size.

- Section "Comparison of RAID10 Option and Nested RAID 10 (1+0)" (page 155)
- Section "Number of Replicas in the mdadm RAID10" (page 156)
- Section "Number of Devices in the mdadm RAID10" (page 156)
- Section "Near Layout" (page 157)
- Section "Far Layout" (page 157)

Comparison of RAID10 Option and Nested RAID 10 (1+0)

The complex RAID 10 is similar in purpose to a nested RAID 10 (1+0), but differs in the following ways:

***Table 7.5** Complex vs. Nested RAID 10*

Feature	mdadm RAID10 Option	Nested RAID 10 (1+0)
Number of devices	Allows an even or odd number of component devices	Requires an even number of component devices
Component devices	Managed as a single RAID device	Manage as a nested RAID device
Striping	Striping occurs in the near or far layout on component devices.	Striping occurs consecutively across component devices

Feature	mdadm RAID10 Option	Nested RAID 10 (1+0)
	The far layout provides sequential read throughput that scales by number of drives, rather than number of RAID 1 pairs.	
Multiple copies of data	Two or more copies, up to the number of devices in the array	Copies on each mirrored segment
Hot spare devices	A single spare can service all component devices	Configure a spare for each underlying mirrored array, or configure a spare to serve a spare group that serves all mirrors.

Number of Replicas in the mdadm RAID10

When configuring a RAID10-level array, you must specify the number of replicas of each data block that are required. The default number of replicas is 2, but the value can be 2 to the number of devices in the array.

Number of Devices in the mdadm RAID10

You must use at least as many component devices as the number of replicas you specify. However, number of component devices in a RAID10-level array does not need to be a multiple of the number of replicas of each data block. The effective storage size is the number of devices divided by the number of replicas.

For example, if you specify 2 replicas for an array created with 5 component devices, a copy of each block is stored on two different devices. The effective storage size for one copy of all data is 5/2 or 2.5 times the size of a component device.

Near Layout

With the near layout, copies of a block of data are striped near each other on different component devices. That is, multiple copies of one data block are at similar offsets in different devices. Near is the default layout for RAID10. For example, if you use an odd number of component devices and two copies of data, some copies are perhaps one chunk further into the device.

The near layout for the `mdadm` RAID10 yields read and write performance similar to RAID 0 over half the number of drives.

Near layout with an even number of disks and two replicas:

```
sda1 sdb1 sdc1 sde1
0 0 1 1
2 2 3 3
4 4 5 5
6 6 7 7
8 8 9 9
```

Near layout with an odd number of disks and two replicas:

```
sda1 sdb1 sdc1 sde1 sdf1
0 0 1 1 2
2 3 3 4 4
5 5 6 6 7
7 8 8 9 9
10 10 11 11 12
```

Far Layout

The far layout stripes data over the early part of all drives, then stripes a second copy of the data over the later part of all drives, making sure that all copies of a block are on different drives. The second set of values start halfway through the component drives.

With a far layout, the read performance of the `mdadm` RAID10 is similar to a RAID 0 over the full number of drives, but write performance is substantially slower than a RAID 0 because there is more seeking of the drive heads. It is best used for read-intensive operations such as for read-only file servers.

Far layout with an even number of disks and two replicas:

```
sda1 sdb1 sdc1 sde1
0 1 2 3
4 5 6 7
. . .
3 0 1 3
7 4 5 6
```

Far layout with an odd number of disks and two replicas:

```
sda1 sdb1 sdc1 sde1 sdf1
0 1 2 3 4
5 6 7 8 9
. . .
4 0 1 2 3
9 5 6 7 8
```

7.3.2 Creating a RAID10 with mdadm

The RAID10-level option for `mdadm` creates a RAID 10 device without nesting. For information about the RAID10-level, see Section 7.3, “Creating a Complex RAID 10 with mdadm” (page 154).

The procedure in this section uses the device names shown in the following table. Make sure to modify the device names with the names of your own devices.

Table 7.6 *Scenario for Creating a RAID 10 Using the mdadm RAID10 Option*

Raw Devices	RAID10 (near or far striping scheme)
`/dev/sdf1`	`/dev/md3`
`/dev/sdg1`	
`/dev/sdh1`	
`/dev/sdi1`	

1. In YaST, create a 0xFD Linux RAID partition on the devices you want to use in the RAID, such as /dev/sdf1, /dev/sdg1, /dev/sdh1, and /dev/sdi1.

2. Open a terminal console, then log in as the root user or equivalent.

3. Create a RAID 10 command. At the command prompt, enter (all on the same line):

```
mdadm --create /dev/md3 --run --level=10 --chunk=4 --raid-devices=4 /dev/sdf1
 /dev/sdg1 /dev/sdh1 /dev/sdi1
```

4. Create a Reiser file system on the RAID 10 device /dev/md3. At the command prompt, enter

```
mkfs.reiserfs /dev/md3
```

5. Edit the /etc/mdadm.conf file to add entries for the component devices and the RAID device /dev/md3. For example:

```
DEVICE /dev/md3
```

6. Edit the /etc/fstab file to add an entry for the RAID 10 device /dev/md3.

7. Reboot the server.

   The RAID10 device is mounted to /raid10.

7.4 Creating a Degraded RAID Array

A degraded array is one in which some devices are missing. Degraded arrays are supported only for RAID 1, RAID 4, RAID 5, and RAID 6. These RAID types are designed to withstand some missing devices as part of their fault-tolerance features. Typically, degraded arrays occur when a device fails. It is possible to create a degraded array on purpose.

RAID Type	Allowable Number of Slots Missing
RAID 1	All but one device

RAID Type	Allowable Number of Slots Missing
RAID 4	One slot
RAID 5	One slot
RAID 6	One or two slots

To create a degraded array in which some devices are missing, simply give the word `missing` in place of a device name. This causes `mdadm` to leave the corresponding slot in the array empty.

When creating a RAID 5 array, `mdadm` automatically creates a degraded array with an extra spare drive. This is because building the spare into a degraded array is generally faster than resynchronizing the parity on a non-degraded, but not clean, array. You can override this feature with the `--force` option.

Creating a degraded array might be useful if you want create a RAID, but one of the devices you want to use already has data on it. In that case, you create a degraded array with other devices, copy data from the in-use device to the RAID that is running in degraded mode, add the device into the RAID, then wait while the RAID is rebuilt so that the data is now across all devices. An example of this process is given in the following procedure:

1 Create a degraded RAID 1 device `/dev/md0`, using one single drive `/dev/sd1`, enter the following at the command prompt:

```
mdadm --create /dev/md0 -l 1 -n 2 /dev/sda1 missing
```

The device should be the same size or larger than the device you plan to add to it.

2 If the device you want to add to the mirror contains data that you want to move to the RAID array, copy it now to the RAID array while it is running in degraded mode.

3 Add a device to the mirror. For example, to add `/dev/sdb1` to the RAID, enter the following at the command prompt:

```
mdadm /dev/md0 -a /dev/sdb1
```

You can add only one device at a time. You must wait for the kernel to build the mirror and bring it fully online before you add another mirror.

4 Monitor the build progress by entering the following at the command prompt:

```
cat /proc/mdstat
```

To see the rebuild progress while being refreshed every second, enter

```
watch -n 1 cat /proc/mdstat
```

Resizing Software RAID Arrays with mdadm

8

This section describes how to increase or reduce the size of a software RAID 1, 4, 5, or 6 device with the Multiple Device Administration (`mdadm(8)`) tool.

WARNING

Before starting any of the tasks described in this chapter, make sure that you have a valid backup of all of the data.

8.1 Understanding the Resizing Process

Resizing an existing software RAID device involves growing or shrinking the space contributed by each component partition.

8.1.1 Guidelines for Resizing a Software RAID

The mdadm(8) tool supports resizing only for software RAID levels 1, 4, 5, and 6. These RAID levels provide disk fault tolerance so that one component partition can be removed at a time for resizing. In principle, it is possible to perform a hot resize for RAID partitions, but you must take extra care for your data when doing so.

The file system that resides on the RAID must also be able to be resized in order to take advantage of the changes in available space on the device. In SUSE Linux Enterprise Server 10 SP1, file system resizing utilities are available for file systems Ext2, Ext3, JFS, and ReiserFS. The utilities support growing and shrinking the size as follows:

Table 8.1 *File System Support for Resizing*

File System	Utility	Increase Size	Decrease Size
Ext2 or Ext3	resize2fs	Yes, offline only	Yes, offline only
JFS	mount -o remount,resize	Yes, online only	No
ReiserFS	resize_reiserfs	Yes, online or offline	Yes, offline only

Resizing any partition or file system involves some risks that can potentially result in losing data.

WARNING

To avoid data loss, make sure to back up your data before you begin any resizing task.

8.1.2 Overview of Tasks

Resizing the RAID involves the following tasks. The order in which these tasks is performed depends on whether you are increasing or decreasing its size.

Table 8.2 Tasks Involved in Resizing a RAID

Tasks	Description	Order If Increasing Size	Order If Decreasing Size
Resize each of the component partitions.	Increase or decrease the active size of each component partition. You remove only one component partition at a time, modify its size, then return it to the RAID.	1	2
Resize the software RAID itself.	The RAID does not automatically know about the increases or decreases you make to the underlying component partitions. You must inform it about the new size.	2	3
Resize the file system.	You must resize the file system that resides on the RAID. This is possible only for file systems that provide tools for resizing, such as Ext2, Ext3, JFS, and ReiserFS.	3	1

8.2 Increasing the Size of a Software RAID

Before you begin, review the guidelines in Section 8.1, “Understanding the Resizing Process” (page 163).

- Section 8.2.1, “Increasing the Size of Component Partitions” (page 166)
- Section 8.2.2, “Increasing the Size of the RAID Array” (page 168)
- Section 8.2.3, “Increasing the Size of the File System” (page 169)

8.2.1 Increasing the Size of Component Partitions

Apply the procedure in this section to increase the size of a RAID 1, 4, 5, or 6. For each component partition in the RAID, remove the partition from the RAID, modify its size, return it to the RAID, then wait until the RAID stabilizes to continue. While a partition is removed, the RAID operates in degraded mode and has no or reduced disk fault tolerance. Even for RAIDs that can tolerate multiple concurrent disk failures, do not remove more than one component partition at a time.

WARNING

If a RAID does not have disk fault tolerance, or it is simply not consistent, data loss results if you remove any of its partitions. Be very careful when removing partitions, and make sure that you have a backup of your data available.

The procedure in this section uses the device names shown in the following table. Make sure to modify the names to use the names of your own devices.

Table 8.3 *Scenario for Increasing the Size of Component Partitions*

RAID Device	Component Partitions
`/dev/md0`	`/dev/sda1`
	`/dev/sdb1`
	`/dev/sdc1`

To increase the size of the component partitions for the RAID:

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Make sure that the RAID array is consistent and synchronized by entering

```
cat /proc/mdstat
```

If your RAID array is still synchronizing according to the output of this command, you must wait until synchronization is complete before continuing.

3. Remove one of the component partitions from the RAID array. For example, to remove /dev/sda1, enter

   ```
   mdadm /dev/md0 --fail /dev/sda1 --remove /dev/sda1
   ```

   In order to succeed, both the fail and remove actions must be done.

4. Increase the size of the partition that you removed in Step 3 (page 167) by doing one of the following:

   - Increase the size of the partition, using a disk partitioner such as fdisk(8), cfdisk(8), or parted(8). This is the usual choice.

   - Replace the disk on which the partition resides with a higher-capacity device.

     This option is possible only if no other file systems on the original disk are accessed by the system. When the replacement device is added back into the RAID, it takes much longer to synchronize the data because all of the data that was on the original device must be rebuilt.

5. Re-add the partition to the RAID array. For example, to add /dev/sda1, enter

   ```
   mdadm -a /dev/md0 /dev/sda1
   ```

   Wait until the RAID is synchronized and consistent before continuing with the next partition.

6. Repeat Step 2 (page 166) through Step 5 (page 167) for each of the remaining component devices in the array. Make sure to modify the commands for the correct component partition.

7. If you get a message that tells you that the kernel could not re-read the partition table for the RAID, you must reboot the computer after all partitions have been resized to force an update of the partition table.

8. Continue with Section 8.2.2, “Increasing the Size of the RAID Array” (page 168).

8.2.2 Increasing the Size of the RAID Array

After you have resized each of the component partitions in the RAID (see Section 8.2.1, “Increasing the Size of Component Partitions” (page 166)), the RAID array configuration continues to use the original array size until you force it to be aware of the newly available space. You can specify a size for the RAID or use the maximum available space.

The procedure in this section uses the device name `/dev/md0` for the RAID device. Make sure to modify the name to use the name of your own device.

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Check the size of the array and the device size known to the array by entering

```
mdadm -D /dev/md0 | grep -e "Array Size" -e "Device
    Size"
```

3 Do one of the following:

- Increase the size of the array to the maximum available size by entering

  ```
  mdadm --grow /dev/md0 -z max
  ```

- Increase the size of the array to a specified value by entering

  ```
  mdadm --grow /dev/md0 -z size
  ```

  Replace *`size`* with an integer value in kilobytes (a kilobyte is 1024 bytes) for the desired size.

4 Recheck the size of your array and the device size known to the array by entering

```
mdadm -D /dev/md0 | grep -e "Array Size" -e "Device
    Size"
```

5 Do one of the following:

- If your array was successfully resized, continue with Section 8.2.3, “Increasing the Size of the File System” (page 169).

- If your array was not resized as you expected, you must reboot, then try this procedure again.

8.2.3 Increasing the Size of the File System

After you increase the size of the array (see Section 8.2.2, "Increasing the Size of the RAID Array" (page 168)), you are ready to resize the file system.

You can increase the size of the file system to the maximum space available or specify an exact size. When specifying an exact size for the file system, make sure the new size satisfies the following conditions:

- The new size must be greater than the size of the existing data; otherwise, data loss occurs.
- The new size must be equal to or less than the current RAID size because the file system size cannot extend beyond the space available.

Ext2 or Ext3

Ext2 and Ext3 file systems can be resized when mounted or unmounted with the command `resize2fs`.

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Increase the size of the file system using one of the following methods:

- To extend the file system size to the maximum available size of the software RAID device called `/dev/md0`, enter

  ```
  resize2fs /dev/md0
  ```

  If a size parameter is not specified, the size defaults to the size of the partition.

- To extend the file system to a specific size, enter

  ```
  resize2fs /dev/md0 size
  ```

  The *`size`* parameter specifies the requested new size of the file system. If no units are specified, the unit of the size parameter is the block size of the file system.

Optionally, the size parameter may be suffixed by one of the following the unit designators: s for 512 byte sectors; K for kilobytes (1 kilobyte is 1024 bytes); M for megabytes; or G for gigabytes.

Wait until the resizing is completed before continuing.

3 If the file system is not mounted, mount it now.

For example, to mount an Ext2 file system for a RAID named `/dev/md0` at mount point `/raid`, enter

```
mount -t ext2 /dev/md0 /raid
```

4 Check the effect of the resize on the mounted file system by entering

```
df -h
```

The Disk Free (`df`) command shows the total size of the disk, the number of blocks used, and the number of blocks available on the file system. The -h option print sizes in human-readable format, such as 1K, 234M, or 2G.

JFS

Resizing a JFS partition is done with a special option to the `mount` command that is specific to the JFS file system:

```
mount -o remount,resize /mnt
```

Using the resize option is valid only during a remount when the volume is already mounted read-write. The mount point is specified rather than the device name.

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Increase the size of the file system on the software RAID device mounted at `/mnt/point`, using one of the following methods:

- To extend the file system size to the maximum available size of the device, enter

  ```
  mount -o remount,resize /mnt/point
  ```

  The resize option with no value specified grows the volume to the full size of the partition.

- To extend the file system to a specific size, enter

  ```
  mount -o remount,resize=size /mnt/point
  ```

  Replace *size* with the desired size in blocks based on the block size used for the file system.

  For example, if you have a 4 GB device with a block size of 4KB, enter

  ```
  mount -o remount,resize=1048576 /mnt/point
  ```

Wait until the resizing is completed before continuing.

3 Check the effect of the resize on the mounted file system by entering

```
df -h
```

The Disk Free (`df`) command shows the total size of the disk, the number of blocks used, and the number of blocks available on the file system. The -h option print sizes in human-readable format, such as 1K, 234M, or 2G.

ReiserFS

As with Ext2 and Ext3, a ReiserFS file system can be increased in size while mounted or unmounted. The resize is done on the block device of your RAID array.

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Increase the size of the file system on the software RAID device called `/dev/md0`, using one of the following methods:

- To extend the file system size to the maximum available size of the device, enter

  ```
  resize_reiserfs /dev/md0
  ```

  When no size is specified, this grows the volume to the full size of the partition.

- To extend the file system to a specific size, enter

  ```
  resize_reiserfs -s size /dev/md0
  ```

  Replace *size* with the desired size in bytes. You can also specify units on the value, such as 50000K (kilobytes), 250M (megabytes), or 2G (gigabytes). Alternatively, you can specify an increase to the current size by prefixing the value with a plus (+) sign. For example, the following command increases the size of the file system on /dev/md0 by 500 MB:

```
resize_reiserfs -s +500M /dev/md0
```

Wait until the resizing is completed before continuing.

3 If the file system is not mounted, mount it now.

For example, to mount an ReiserFS file system for a RAID named /dev/md0 at mount point /raid, enter

```
mount -t reiserfs /dev/md0 /raid
```

4 Check the effect of the resize on the mounted file system by entering

```
df -h
```

The Disk Free (df) command shows the total size of the disk, the number of blocks used, and the number of blocks available on the file system. The -h option print sizes in human-readable format, such as 1K, 234M, or 2G.

8.3 Decreasing the Size of a Software RAID

Before you begin, review the guidelines in Section 8.1, "Understanding the Resizing Process" (page 163).

- Section 8.3.1, "Decreasing the Size of the File System" (page 172)
- Section 8.3.2, "Decreasing the Size of Component Partitions" (page 175)
- Section 8.3.3, "Decreasing the Size of the RAID Array" (page 177)

8.3.1 Decreasing the Size of the File System

When decreasing the size of the file system on a RAID device, make sure the new size satisfies the following conditions:

- The new size must be greater than the size of the existing data; otherwise, data loss occurs.

- The new size must be equal to or less than the current RAID size because the file system size cannot extend beyond the space available.

In SUSE Linux Enterprise Server SP1, only Ext2, Ext3, and ReiserFS provide utilities for shrinking the size of the file system. Use the appropriate procedure below for decreasing the size of your file system.

The procedures in this section use the device name `/dev/md0` for the RAID device. Make sure to modify commands to use the name of your own device.

Ext2 or Ext3

The Ext2 and Ext3 file systems can be resized when mounted or unmounted.

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Decrease the size of the file system on the RAID by entering

```
resize2fs /dev/md0 <size>
```

Replace *`size`* with an integer value in kilobytes for the desired size. (A kilobyte is 1024 bytes.)

Wait until the resizing is completed before continuing.

3 If the file system is not mounted, mount it now. For example, to mount an Ext2 file system for a RAID named `/dev/md0` at mount point `/raid`, enter

```
mount -t ext2 /dev/md0 /raid
```

4 Check the effect of the resize on the mounted file system by entering

```
df -h
```

The Disk Free (`df`) command shows the total size of the disk, the number of blocks used, and the number of blocks available on the file system. The -h option print sizes in human-readable format, such as 1K, 234M, or 2G.

JFS

JFS does not support shrinking a volume.

ReiserFS

ReiserFS file systems can be shrunk only if the volume is unmounted.

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Unmount the device by entering

```
umount /mnt/point
```

If the partition you are attempting to shrink contains system files (such as the root (`/`) volume), unmounting is possible only when booting from a bootable CD or floppy.

3 Decrease the size of the file system on the software RAID device called `/dev/md0` by entering

```
resize_reiserfs -s size /dev/md0
```

Replace *`size`* with the desired size in bytes. You can also specify units on the value, such as 50000K (kilobytes), 250M (megabytes), or 2G (gigabytes). Alternatively, you can specify a decrease to the current size by prefixing the value with a minus (-) sign. For example, the following command reduces the size of the file system on `/dev/md0` by 500 MB:

```
resize_reiserfs -s -500M /dev/md0
```

Wait until the resizing is completed before continuing.

4 Mount the file system by entering

```
mount -t reiserfs /dev/md0 /mnt/point
```

5 Check the effect of the resize on the mounted file system by entering

```
df -h
```

The Disk Free (`df`) command shows the total size of the disk, the number of blocks used, and the number of blocks available on the file system. The -h option print sizes in human-readable format, such as 1K, 234M, or 2G.

8.3.2 Decreasing the Size of Component Partitions

Resize the RAID's component partitions one at a time. For each component partition, you remove it from the RAID, modify its partition size, return the partition to the RAID, then wait until the RAID stabilizes. While a partition is removed, the RAID operates in degraded mode and has no or reduced disk fault tolerance. Even for RAIDs that can tolerate multiple concurrent disk failures, you should never remove more than one component partition at a time.

WARNING

If a RAID does not have disk fault tolerance, or it is simply not consistent, data loss results if you remove any of its partitions. Be very careful when removing partitions, and make sure that you have a backup of your data available.

The procedure in this section uses the device names shown in the following table. Make sure to modify the commands to use the names of your own devices.

Table 8.4 *Scenario for Increasing the Size of Component Partitions*

RAID Device	Component Partitions
`/dev/md0`	`/dev/sda1`
	`/dev/sdb1`
	`/dev/sdc1`

To resize the component partitions for the RAID:

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Make sure that the RAID array is consistent and synchronized by entering

```
cat /proc/mdstat
```

If your RAID array is still synchronizing according to the output of this command, you must wait until synchronization is complete before continuing.

3 Remove one of the component partitions from the RAID array. For example, to remove `/dev/sda1`, enter

```
mdadm /dev/md0 --fail /dev/sda1 --remove /dev/sda1
```

In order to succeed, both the fail and remove actions must be done.

4 Increase the size of the partition that you removed in Step 3 (page 167) by doing one of the following:

- Use a disk partitioner such as `fdisk`, `cfdisk`, or `parted` to increase the size of the partition.
- Replace the disk on which the partition resides with a different device.

  This option is possible only if no other file systems on the original disk are accessed by the system. When the replacement device is added back into the RAID, it takes much longer to synchronize the data.

5 Re-add the partition to the RAID array. For example, to add `/dev/sda1`, enter

```
mdadm -a /dev/md0 /dev/sda1
```

Wait until the RAID is synchronized and consistent before continuing with the next partition.

6 Repeat Step 2 (page 166) through Step 5 (page 167) for each of the remaining component devices in the array. Make sure to modify the commands for the correct component partition.

7 If you get a message that tells you that the kernel could not re-read the partition table for the RAID, you must reboot the computer after resizing all of its component partitions.

8 Continue with Section 8.3.3, “Decreasing the Size of the RAID Array” (page 177).

8.3.3 Decreasing the Size of the RAID Array

After you have resized each of the component partitions in the RAID, the RAID array configuration continues to use the original array size until you force it to be aware of the newly available space. You can specify a size for the RAID or use the maximum available space.

The procedure in this section uses the device name `/dev/md0` for the RAID device. Make sure to modify commands to use the name of your own device.

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Check the size of the array and the device size known to the array by entering

```
mdadm -D /dev/md0 | grep -e "Array Size" -e "Device
    Size"
```

3 Do one of the following:

- Increase the size of the array to the maximum available size by entering

  ```
  mdadm --grow /dev/md0 -z max
  ```

- Increase the size of the array to a specified value by entering

  ```
  mdadm --grow /dev/md0 -z size
  ```

  Replace *`size`* with an integer value in kilobytes for the desired size. (A kilobyte is 1024 bytes.)

4 Recheck the size of your array and the device size known to the array by entering

```
mdadm -D /dev/md0 | grep -e "Array Size" -e "Device
    Size"
```

5 Do one of the following:

- If your array was successfully resized, you are done.

- If your array was not resized as you expected, you must reboot, then try this procedure again.

Installing and Managing DRBD Services 9

This section describes how to install, configure, and manage a device-level software RAID 1 across a network using DRBD (Distributed Replicated Block Device) for Linux.

9.1 Understanding DRBD

DRBD allows you to create a mirror of two block devices that are located at two different sites across an IP network. When used with HeartBeat 2 (HB2), DRBD supports distributed high-availability Linux clusters.

IMPORTANT

The data traffic between mirrors is not encrypted. For secure data exchange, you should deploy a virtual private network (VPN) solution for the connection.

Data on the primary device is replicated to the secondary device in a way that ensures that both copies of the data are always identical.

By default, DRBD uses the TCP port 7788 for communications between DRBD nodes. By default, DRBD typically uses the TCP port 7780 and higher for communication between DRBD nodes. Each DRBD resource users a different port. Make sure that your firewall does not prevent communication on the configured ports.

The open source version of DRBD supports up to 4 TB as maximal overall size of all devices. If you need a device larger than 4 TB, you can use DRBD+, which is commercially available from Linbit.

You must set up the DRBD devices before creating file systems on them. Everything to do with user data should be done solely via the `/dev/drbd<n>` device, not on the raw device, because DRBD uses the last 128 MB of the raw device for metadata if you specify `internal` for the `meta-disk` setting.

IMPORTANT

Make sure to create file systems only on the `/dev/drbd<n>` device, not on the raw device.

For example, if the raw device is 1024 MB in size, the DRBD device has only 896 MB available for data, with 128 MB hidden and reserved for the metadata. Any attempt to access the space between 896 MB and 1024 MB fails because it is not available for user data.

9.2 Installing DRBD Services

1 Install the High Availability (HA) pattern on both SUSE Linux Enterprise Servers in your networked cluster. Installing HA also installs the `drbd` program files.

1a Log in as the `root` user or equivalent, then open YaST.

1b Choose *Software* > *Software Management.*

1c Change the filter to *Patterns*.

1d Under *Base Technologies*, select *High Availability*.

1e Click *Accept*.

2 Install the `drbd` kernel modules on both servers.

2a Log in as the `root` user or equivalent, then open YaST.

2b Choose *Software* > *Software Management.*

2c Change the filter to *Search*.

2d Type `drbd`, then click *Search*.

2e Select all of the `drbd-kmp-*` packages.

2f Click *Accept*.

9.3 Configuring the DRBD Service

DRDB is controlled by settings in the `/etc/drbd.conf` configuration file. The file contents should be identical on both nodes. A sample configuration file is located in the `/usr/share/doc/packages/drbd` folder. Options are also described in the `drbd.conf(5)` man page.

IMPORTANT

Beginning with DRBD 8.3 the configuration file is split into separate files, located under the directory `/etc/drbd.d`.

Only one server can mount a DRBD partition at a time. On node 1, the assigned disk (such as `/dev/sdc`) is mounted as primary and mirrored to the same device (`/dev/sdc`) on node 2.

NOTE

The following procedure uses the server names node 1 and node 2, and the cluster resource name r0. It sets up node 1 as the primary node. Make sure to modify the instructions to use your own node and file names.

1. Log in as the `root` user or equivalent on each node.

2. Open the `/etc/drbd.conf` file on the primary node (`node1`) in a text editor, modify the following parameters in the `on hostname {}` sections, then save the file.

   - **on:** Specify the hostname for each node, such as `node1` and `node2`.

   - **device:** Specify the assigned `/dev/drbd<n>` device, such as `/dev/drdb0`.

   - **disk:** Specify the assigned disk (such as `//devsdc`) or partition (such as `/dev/sdc7`) to use for the `/dev/drbd<n>` device.

   - **address:** Specify the IP address of each node and the port number (default is 7788) to use for communications between the nodes. Each resource needs an individual port, usually beginning with port 7780. The port must be opened in the firewall on each node.

   - **metadisk:** Specify internal to use the last 128 megabytes of the device, or specify an external device to use.

   All of these options are explained in the examples in the `/usr/share/doc/packages/drbd/drbd.conf` file and in the man page of `drbd.conf(5).`

   For example, the following is a sample DRBD resource setup that defines `/dev/drbd1`:

   ```
   global {
     usage-count ask;
   }
   common {
     protocol C;
   }
   resource r0 {
     on node1 {
       device    /dev/drbd0;
       disk      /dev/sdb;
   ```

```
    address   192.168.1.101:7788;
    meta-disk internal;
  }
  on node2 {
    device    /dev/drbd0;
    disk      /dev/sdb;
    address   192.168.1.102:7788;
    meta-disk internal;
  }
}
```

3 Verify the syntax of your configuration file by entering

```
drbdadm dump all
```

4 Copy the `/etc/drbd.conf` file to the `/etc/drbd.conf` location on the secondary server (node 2).

```
scp /etc/drbd.conf <node 2> /etc
```

5 Initialize and start the drbd service on both systems by entering the following commands on each node:

```
drbdadm --ignore-sanity-checks create-md r0
```

```
rcdrbd start
```

6 Configure `node1` as the primary node by entering the following on `node1`:

```
drbdsetup /dev/drbd0 primary --do-what-I-say
```

NOTE

The --do-what-i-say option has been renamed to --overwrite-data-of-peer in the recent versions of DRBD.

7 Check the DRBD service status by entering the following on each node:

```
rcdrbd status
```

Before proceeding, wait until the block devices on both nodes are fully synchronized. Repeat the `rcdrbd status` command to follow the synchronization progress.

8 After the block devices on both nodes are fully synchronized, format the DRBD device on the primary with a file system such as reiserfs. Any Linux file system can be used. For example, enter

```
mkfs.reiserfs -f /dev/drbd0
```

IMPORTANT

Always use the `/dev/drbd<n>` name in the command, not the actual `/dev/disk` device name.

9.4 Testing the DRBD Service

If the install and configuration procedures worked as expected, you are ready to run a basic test of the drbd functionality. This test also helps with understanding how the software works.

1 Test the DRBD service on node 1.

- **1a** Open a terminal console, then log in as the `root` user or equivalent.

- **1b** Create a mount point on node 1, such as `/r0mount`, by entering

  ```
  mkdir /r0mount
  ```

- **1c** Mount the `drbd` device by entering

  ```
  mount -o rw /dev/drbd0 /r0mount
  ```

- **1d** Create a file from the primary node by entering

  ```
  touch /r0mount/from_node1
  ```

2 Test the DRBD service on node 2.

- **2a** Open a terminal console, then log in as the `root` user or equivalent.

- **2b** Dismount the disk on node 1 by typing the following command on node 1:

  ```
  umount /r0mount
  ```

2c Downgrade the DRBD service on node 1 by typing the following command on node 1:

```
drbdadm secondary r0
```

2d On node 2, promote the DRBD service to primary by entering

```
drbdadm primary r0
```

2e On node 2, check to see if node 2 is primary by entering

```
rcdrbd status
```

2f On node 2, create a mount point such as `/r0mount`, by entering

```
mkdir /r0mount
```

2g On node 2, mount the DRBD device by entering

```
mount -o rw /dev/drbd0 /r0mount
```

2h Verify that the file you created on node 1 in Step 1d (page 184) is viewable by entering

```
ls /r0mount
```

The `/r0mount/from_node1` file should be listed.

3 If the service is working on both nodes, the DRBD setup is complete.

4 Set up node 1 as the primary again.

4a Dismount the disk on node 2 by typing the following command on node 2:

```
umount /r0mount
```

4b Downgrade the DRBD service on node 2 by typing the following command on node 2:

```
drbdadm secondary r0
```

4c On node 1, promote the DRBD service to primary by entering

```
drbdadm primary r0
```

4d On node 1, check to see if node 1 is primary by entering

```
service drbd status
```

5 To get the service to automatically start and fail over if the server has a problem, you can set up DRBD as a high availability service with HeartBeat 2.

For information about installing and configuring HeartBeat 2 for SUSE Linux Enterprise Server 10, see the *HeartBeat 2 Installation and Setup Guide* [`http://www.novell.com/documentation/sles10/hb2/data/hb2_config.html`] on the Novell Documentation Web site for SUSE Linux Enterprise Server 10 [`http://www.novell.com/documentation/sles10`].

9.5 Troubleshooting DRBD

Use the following to troubleshoot problems with your DRBD setup:

- Section 9.5.1, "Configuration" (page 186)
- Section 9.5.2, "Host Names" (page 187)
- Section 9.5.3, "TCP Port 7788" (page 187)
- Section 9.5.4, "The --do-what-i-say Option " (page 187)
- Section 9.5.5, "DRBD Devices Are Broken after Reboot" (page 187)

9.5.1 Configuration

If the initial `drbd` setup does not work as expected, there is probably something wrong with your configuration.

To get information about the configuration:

1 Open a terminal console, then log in as the `root` user or equivalent.

2 Test the configuration file by running `drbdadm` with the `-d` option. Enter

```
drbdadm -d adjust r0
```

In a dry run of the `adjust` option, `drbdadm` compares the actual configuration of the DRBD resource with your DRBD configuration file, but it does not execute the calls. Review the output to make sure you know the source and cause of any errors.

3 If there are errors in the `drbd.conf` file, correct them before continuing.

4 If the partitions and settings are correct, run `drbdadm` again without the `-d` option. Enter

```
drbdadm adjust r0
```

This applies the configuration file to the DRBD resource.

9.5.2 Host Names

Please note that for DRBD, hostnames are case sensitive and therefore `Node0` would be a different host than `node0`.

9.5.3 TCP Port 7788

If your system is unable to connect to the peer, this also may be a problem of a local firewall. By default, DRBD uses the TCP port 7788 to access the other node. Make sure that this port is accessible on both nodes.

9.5.4 The --do-what-i-say Option

The --do-what-i-say option has been renamed to --overwrite-data-of-peer in the recent versions of DRBD.

9.5.5 DRBD Devices Are Broken after Reboot

In cases when DRBD does not know which of the real devices holds the latest data, it changes to a split brain condition. In this case, the respective DRBD subsystems come up as secondary and do not connect to each other. The following message is written to `/var/log/messages`:

```
Split-Brain detected, dropping connection!
```

To resolve this situation, the node with the data to be discarded should be assigned as secondary.

1 Log in as the `root` user on the node that has data to be discarded, and open a terminal console.

2 Enter the following commands:

   ```
   drbdadm secondary r0
   ```

   ```
   drbdadm -- --discard-my-data connect r0
   ```

3 Log in as the `root` user on the node that has the good data, and open a terminal console.

4 Enter:

   ```
   drbdadm connect r0
   ```

9.6 Additional Information

- Section 9.6.1, “Open Source Resources for DRBD” (page 188)
- Section 9.6.2, “HeartBeat2” (page 189)

9.6.1 Open Source Resources for DRBD

The following open-source resources are available for DRBD:

- Find a commented example configuration for DRBD at `/usr/share/doc/packages/drbd/drbd.conf`.
- The following man pages for DRBD are available in the distribution:

  ```
  drbd(8)
  drbddisk(8)
  drbdsetup(8)
  ```

```
drbdadm(8)
drbd.conf(5)
```

- DRBD.org [http://www.drbd.org]
- DRBD references at the Linux High-Availability Project [http://linux-ha.org/DRBD] by the Linux High-Availability Project
- DRBD How-To by the Linux Pacemaker Cluster Stack Project [http://clusterlabs.org/wiki/DRBD_HowTo_1.0]

9.6.2 HeartBeat2

For information about installing and configuring HeartBeat 2 for SUSE Linux Enterprise Server 10, see the *HeartBeat 2 Installation and Setup Guide* [http://www.novell.com/documentation/sles10/hb2/data/hb2_config.html] on the Novell Documentation Web site for SUSE Linux Enterprise Server 10 [http://www.novell.com/documentation/sles10].

Troubleshooting Storage Issues 10

This section describes how to work around known issues for EVMS devices, software RAIDs, multipath I/O, and volumes.

10.1 Is DM-MP Available for the Boot Partition?

In the initial release of SUSE Linux Enterprise Server 10, DM-MP is not available for the boot partition, because the boot loader cannot handle multipath I/O. Therefore, we recommend you set up a separate boot (`/boot`) partition when using multipathing. This issue has been resolved in Support Pack 1 and later.

10.2 Rescue System Cannot Find Devices That Are Managed by EVMS

The Linux rescue system does not automatically activate volume manager support for LVM or EVMS. For example, if you are using EVMS as the volume manager for your system device, you might not be able to see the device in order to mount it under the rescue system.

You must manually activate EVMS in order for the Linux rescue system to detect system devices that are managed by EVMS.

- At the terminal console prompt, enter the following as the `root` user:

  ```
  evms_activate
  ```

10.3 Volumes on EVMS Devices Do Not Appear After Reboot

If you create volumes on EVMS devices and they cannot be found after you reboot the server, run `chkconfig` to make sure that evms_activate runs before FSTAB.

- At a terminal console prompt, enter either

  ```
  chkconfig evms on
  ```

  or

  ```
  chkconfig boot.evms on
  ```

  This ensures that evms_activate runs before FSTAB each time your servers reboot.

10.4 Volumes on EVMS Devices Do Not Appear When Using iSCSI

If you have installed and configured an iSCSI SAN, and have created and configured EVMS disks or volumes on that iSCSI SAN, your EVMS volumes might not be visible or accessible after reboot. This problem is caused by EVMS starting before the iSCSI service. iSCSI must be started and running before any disks or volumes on the iSCSI SAN can be accessed.

To resolve this problem, use the `chkconfig` command at the Linux server console of every server that is part of your iSCSI SAN to correct the order that iSCSI and EVMS are started.

- At a terminal console prompt, enter

  ```
  chkconfig boot.evms on
  ```

  This ensures that EVMS and iSCSI start in the proper order each time your servers reboot.

10.5 Device Nodes Are Not Automatically Re-Created on Restart

Effective in SUSE Linux Enterprise 10, the `/dev` directory is on `tmpfs` and the device nodes are automatically re-created on boot. It is no longer necessary to modify the `/etc/init.d/boot.evms` script to delete the device nodes on system reboot as was required for previous versions of SUSE Linux.

The following procedure is provided for users who might encounter this issue for SUSE Linux Enterprise Server 9 and earlier:

1. Open the `/etc/init.d/boot.evms` script in a text editor.

2. Add the following lines to the Stop section:

```
mount -n -o remount,rw /

echo -en "\nDeleting devices nodes"

rm -rf /dev/evms

mount -n -o remount,ro /
```

For example, the Stop section looks like this after the edit:

```
stop)

    echo -n "Stopping EVMS"

    mount -n -o remount,rw /

    echo -en "\nDeleting devices nodes"

    rm -rf /dev/evms

    mount -n -o remount,ro /

    rc_status -v

  ;;
```

3 Save the file.

4 Continue with Section “Restart the Server” (page 19).

A Documentation Updates

This section contains information about documentation content changes made to the *SUSE Linux Enterprise Server Storage Administration Guide* since the initial release of SUSE Linux Enterprise Server 10. If you are an existing user, review the change entries to readily identify modified content. If you are a new user, simply read the guide in its current state.

This document was updated on the following dates:

- Section A.1, “January 2011” (page 196)
- Section A.2, “June 21, 2010” (page 197)
- Section A.3, “June 11, 2010” (page 197)
- Section A.4, “February 23, 2010” (page 199)
- Section A.5, “January 20, 2010” (page 199)
- Section A.6, “October 20, 2009” (page 200)
- Section A.7, “September 2, 2009 (SLES 10 SP3)” (page 201)
- Section A.8, “May 15, 2009” (page 203)
- Section A.9, “November 24, 2008” (page 203)
- Section A.10, “June 10, 2008” (page 204)

- Section A.11, “March 20, 2008 (SLES 10 SP2)” (page 205)

A.1 January 2011

Updates were made to the following sections. The changes are explained below.

- Section A.1.1, “Managing Multipath I/O for Devices” (page 196)
- Section A.1.2, “Installing and Managing DRBD Services” (page 196)

A.1.1 Managing Multipath I/O for Devices

Location	Change
Tuning the Failover for Specific Host Bus Adapters	This section was removed. For HBA failover guidance, refer to your vendor documentation.
Section 5.14, “Additional Information” (page 110)	Removed obolete references.

A.1.2 Installing and Managing DRBD Services

Location	Change
Section 9.3, “Configuring the DRBD Service” (page 181)	The procedure was updated with an example. The following command is used to initialize a newly defined DRBD resource: `drbdadm --ignore-sanity-checks create-md r0`

Location	Change
Script for Configuring the /etc/drbd.conf File	This section was removed. The open source `add_resource_script.sh` file is no longer available.

A.2 June 21, 2010

Updates were made to the following sections. The changes are explained below.

- Section A.2.1, "Managing Multipath I/O" (page 197)

A.2.1 Managing Multipath I/O

Location	Change
Section "Configuring User-Friendly Names or Alias Names in /etc/multipath.conf" (page 79)	Using user-friendly names for the root device can result in data loss. Added alternatives from *TID 7001133: Recommendations for the usage of user_friendly_names in multipath configurations* [http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=7001133].

A.3 June 11, 2010

Updates were made to the following sections. The changes are explained below.

- Section A.3.1, "Managing Multipath I/O" (page 198)
- Section A.3.2, "Managing Software RAIDs 6 and 10 with mdadm" (page 199)

A.3.1 Managing Multipath I/O

Location	Change
Section 5.2.3, “Using LVM2 on Multipath Devices” (page 58)	The example in Step 3 (page 58) was corrected.
Section 5.2.6, “SAN Timeout Settings When the Root Device Is Multipathed” (page 59)	This section is new.
Section 5.3.4, “The Linux multipath(8) Command” (page 69)	Expanded the defintions for options.
Section 5.3.2, “Multipath I/O Management Tools” (page 67)	The file list for a package can vary for different server architectures. For a list of files included in the multipath-tools package, go to the *SUSE Linux Enterprise Server Technical SpecificationsPackage Descriptions* Web page [`http://www.novell.com/products/server/techspecs.html`], find your architecture and select *Packages Sorted by Name*, then search on “multipath-tools” to find the package list for that architecture.
Section 5.4.1, “Preparing SAN Devices for Multipathing” (page 72)	If the SAN device will be used as the root device on the server, modify the timeout settings for the device as described in Section 5.2.6, “SAN Timeout Settings When the Root Device Is Multipathed” (page 59).
Section “Verifying the Setup in the etc/multipath.conf File” (page 77)	Added example output for -v3 verbosity.

A.3.2 Managing Software RAIDs 6 and 10 with mdadm

Location	Change
Section "Far Layout" (page 157)	Errata in the example were corrected.

A.4 February 23, 2010

Updates were made to the following section. The changes are explained below.

- Section A.4.1, "Managing Multipath I/O" (page 199)

A.4.1 Managing Multipath I/O

Location	Change
Section 5.9, "Scanning for New Devices without Rebooting" (page 102)	Added information about using the `rescan-scsi-bus.sh` script to scan for devices without rebooting.
Section 5.10, "Scanning for New Partitioned Devices without Rebooting" (page 105)	Added information about using the `rescan-scsi-bus.sh` script to scan for devices without rebooting.

A.5 January 20, 2010

Updates were made to the following section. The changes are explained below.

- Section A.5.1, "Managing Multipath I/O" (page 200)

A.5.1 Managing Multipath I/O

Location	Change
Section “Configuring Default Multipath Behavior in /etc/multipath.conf” (page 84)	In the `default_getuid` command line, use the path `/sbin/scsi_id` as shown in the above example instead of the sample path of `/lib/udev/scsi_id` that is found in the sample file `/usr/share/doc/packages/multipath-tools/multipath.conf.synthetic` file (and in the default and annotated sample files).
getuid in Table 5.6, “Multipath Attributes” (page 88)	Use the path `/sbin/scsi_id` instead of the path `/lib/udev/sbin_id` that is in the sample file `/usr/share/doc/packages/multipath-tools/multipath.conf.synthetic` file (and in the default and annotated sample files).

A.6 October 20, 2009

Updates were made to the following section. The changes are explained below.

- Section A.6.1, “Managing Multipath I/O” (page 200)

A.6.1 Managing Multipath I/O

Location	Change
Section “Configuring Default Multipath Behavior in /etc/multipath.conf” (page 84)	Changed getuid_callout to getuid.

Location	Change
Section “Understanding Priority Groups and Attributes” (page 88)	Changed getuid_callout to getuid.

A.7 September 2, 2009 (SLES 10 SP3)

Updates were made to the following sections. The changes are explained below.

- Section A.7.1, “Managing Multipath I/O” (page 201)
- Section A.7.2, “Managing Software RAIDs 6 and 10 with mdadm” (page 202)
- Section A.7.3, “Overview of EVMS” (page 202)

A.7.1 Managing Multipath I/O

Location	Change
Section 5.2.5, “Using --noflush with Multipath Devices” (page 59)	This section is new.
Section “Blacklisting Non-Multipathed Devices in /etc/multipath.conf” (page 83)	The keyword devnode_blacklist has been deprecated and replaced with the keyword blacklist.
Section 5.7, “Configuring Multipath I/O for the Root Device” (page 98)	Added Step 4 (page 99) and Step 6 (page 99) for System Z.

Location	Change
Section 5.10, "Scanning for New Partitioned Devices without Rebooting" (page 105)	Corrected the syntax for the command lines in Step 2.
Section 5.10, "Scanning for New Partitioned Devices without Rebooting" (page 105)	Step 7 (page 105) replaces old Step 7 and Step 8.

A.7.2 Managing Software RAIDs 6 and 10 with mdadm

Location	Change
Section 7.4, "Creating a Degraded RAID Array" (page 159)	To see the rebuild progress while being refreshed every second, enter `watch -n 1 cat /proc/mdstat`

A.7.3 Overview of EVMS

Location	Change
Section 1.3, "Supported File Systems" (page 3)	The Novell Storage Services (NSS) file system is also supported when used with the Novell Open Enterprise Server 2 for SUSE Linux Enterprise Server 10 SP1 (or later versions of OES 2 and SLES 10).

A.8 May 15, 2009

Updates were made to the following section. The changes are explained below.

- Section A.8.1, "Managing Multipath I/O" (page 203)

A.8.1 Managing Multipath I/O

Location	Change
Section 5.7, "Configuring Multipath I/O for the Root Device" (page 98)	DM-MP is available but is not supported for `/boot` and `/root` in SUSE Linux Enterprise Server 10 SP1 and later if all online updates have been applied. More specifically, you need `mkinitrd` 1.2-106.61 and `multipath-tools` 0.4.7-34.23 or later. However, if you install the packages and set up the configuration, you might run into update issues later. Full multipath support is available in SUSE Linux Enterprise Server 11.

A.9 November 24, 2008

Updates were made to the following sections. The changes are explained below.

- Section A.9.1, "Managing Multipath I/O" (page 204)
- Section A.9.2, "Using UUIDs to Mount Devices" (page 204)

A.9.1 Managing Multipath I/O

Location	Change
Section 5.2.9, "Supported Storage Arrays for Multipathing" (page 61) Section "Applying Changes Made to the /etc/multipath.conf File" (page 85)	After you modify the `/etc/multipath.conf` file, you must run `mkinitrd` to re-create the INITRD on your system, then reboot in order for the changes to take effect.

A.9.2 Using UUIDs to Mount Devices

Location	Change
Section 3.2.1, "Using UUIDs to Assemble or Activate File System Devices" (page 34)	The device ID assigned by the manufactuer for a drive never changes, no matter where the device is mounted, so it can always be found at boot. The UUID is a property of the filesystem and can change if you reformat the drive.

A.10 June 10, 2008

Updates were made to the following section. The changes are explained below.

- Section A.10.1, "Managing Multipath I/O" (page 205)

A.10.1 Managing Multipath I/O

Location	Change
Section 5.3.3, “Using mdadm for Multipathed Devices” (page 68)	For information about modifying the `/etc/lvm/lvm.conf`file, see Section 5.2.3, “Using LVM2 on Multipath Devices” (page 58).
Section “Verifying the Setup in the etc/multipath.conf File” (page 77)	Use the -v3 option to show the full path list.

A.11 March 20, 2008 (SLES 10 SP2)

Updates were made to the following section. The changes are explained below.

- Section A.11.1, “Managing Multipath I/O” (page 205)

A.11.1 Managing Multipath I/O

Location	Change
Section 5.2.8, “Supported Architectures for Multipath I/O” (page 61)	This section is new.
Section 5.2.9, “Supported Storage Arrays for Multipathing” (page 61)	This section was reorganized for clarity.
Section 5.4.5, “Creating and Configuring the /etc/multipath.conf File” (page 76)	The `/etc/multipath.conf` file does not exist unless you create it.

Location	Change
Section 5.6, “Configuring Path Failover Policies and Priorities” (page 86)	This section is new.
Section 5.7, “Configuring Multipath I/O for the Root Device” (page 98)	DM-MP is now available and supported for `/boot` and `/root` in SUSE Linux Enterprise Server 10 SP1 and later if all online updates have been applied. More specifically, you need `mkinitrd` 1.2-106.61 and `multipath-tools` 0.4.7-34.23 or later.