

SUSE® Linux Enterprise Server

11 SP1

www.novell.com

March 15, 2010

SLES 11 SP1: Storage Administration Guide



SLES 11 SP1: 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.

Copyright © 2009–2010 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

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	vii
1 Overview of File Systems in Linux	1
1.1 Terminology	1
1.2 Major File Systems in Linux	2
1.3 Other Supported File Systems	9
1.4 Large File Support in Linux	10
1.5 Additional Information	12
2 What's New	15
2.1 What's New in SLES 11 SP1	15
2.2 What's New in SLES 11	20
3 Planning a Storage Solution	25
3.1 Partitioning Devices	25
3.2 Multipath Support	25
3.3 Software RAID Support	26
3.4 File System Snapshots	26
3.5 Backup and Antivirus Support	26
4 LVM Configuration	29
4.1 Understanding the Logical Volume Manager	30
4.2 Creating LVM Partitions	32
4.3 Creating Volume Groups	32
4.4 Configuring Physical Volumes	33
4.5 Configuring Logical Volumes	34

4.6	Direct LVM Management	36
4.7	Resizing an LVM Partition	36
5	Resizing File Systems	39
5.1	Guidelines for Resizing	39
5.2	Increasing an Ext2 or Ext3 File System	41
5.3	Increasing the Size of a Reiser File System	42
5.4	Decreasing the Size of an Ext2 or Ext3 File System	43
5.5	Decreasing the Size of a Reiser File System	44
6	Using UUIDs to Mount Devices	47
6.1	Naming Devices with udev	47
6.2	Understanding UUIDs	48
6.3	Using UUIDs in the Boot Loader and /etc/fstab File (x86)	49
6.4	Using UUIDs in the Boot Loader and /etc/fstab File (IA64)	52
6.5	Additional Information	53
7	Managing Multipath I/O for Devices	55
7.1	Understanding Multipathing	56
7.2	Planning for Multipathing	57
7.3	Multipath Management Tools	67
7.4	Configuring the System for Multipathing	75
7.5	Enabling and Starting Multipath I/O Services	86
7.6	Configuring Path Failover Policies and Priorities	86
7.7	Tuning the Failover for Specific Host Bus Adapters	100
7.8	Configuring Multipath I/O for the Root Device	100
7.9	Configuring Multipath I/O for an Existing Software RAID	105
7.10	Scanning for New Devices without Rebooting	108
7.11	Scanning for New Partitioned Devices without Rebooting	111
7.12	Viewing Multipath I/O Status	113
7.13	Managing I/O in Error Situations	114
7.14	Resolving Stalled I/O	116
7.15	Additional Information	116
7.16	What's Next	117
8	Software RAID Configuration	119
8.1	Understanding RAID Levels	120
8.2	Soft RAID Configuration with YaST	122
8.3	Troubleshooting	124
8.4	For More Information	125

9	Configuring Software RAID for the Root Partition	127
9.1	Prerequisites for the Software RAID	127
9.2	Enabling iSCSI Initiator Support at Install Time	128
9.3	Enabling Multipath I/O Support at Install Time	129
9.4	Creating a Software RAID Device for the Root (/) Partition	129
10	Managing Software RAID 6 and 10 with mdadm	135
10.1	Creating a RAID 6	135
10.2	Creating Nested RAID 10 Devices with mdadm	137
10.3	Creating a Complex RAID 10 with mdadm	142
10.4	Creating a Degraded RAID Array	147
11	Resizing Software RAID Arrays with mdadm	151
11.1	Understanding the Resizing Process	151
11.2	Increasing the Size of a Software RAID	153
11.3	Decreasing the Size of a Software RAID	159
12	iSNS for Linux	167
12.1	How iSNS Works	168
12.2	Installing iSNS Server for Linux	169
12.3	Configuring iSNS Discovery Domains	171
12.4	Starting iSNS	177
12.5	Stopping iSNS	178
12.6	For More Information	178
13	Mass Storage over IP Networks: iSCSI	179
13.1	Installing iSCSI	180
13.2	Setting Up an iSCSI Target	182
13.3	Configuring iSCSI Initiator	191
14	Volume Snapshots	199
14.1	Understanding Volume Snapshots	199
14.2	Creating Linux Snapshots with LVM	201
14.3	Monitoring a Snapshot	201
14.4	Deleting Linux Snapshots	202
15	Troubleshooting Storage Issues	203
15.1	Is DM-MPIO Available for the Boot Partition?	203

A Documentation Updates

205

A.1	May 2010 (SLES 11 SP1)	206
A.2	February 23, 2010	207
A.3	January 20, 2010	208
A.4	December 1, 2009	209
A.5	October 20, 2009	211
A.6	August 3, 2009	212
A.7	June 22, 2009	213
A.8	May 21, 2009	215

About This Guide

This guide provides information about how to manage storage devices on a SUSE® Linux Enterprise Server 11 Support Pack 1 (SP1) server.

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 11 SP1 Storage Administration Guide*, visit the Novell® Documentation Web site for SUSE Linux Enterprise Server 11 SP1 [<http://www.novell.com/documentation/sles11>].

Additional Documentation

For information about partitioning and managing devices, see the *SUSE Linux Enterprise Server 11 SP1 Installation and Administration Guide* [<http://www.novell.com/documentation/sles11>].

Documentation Conventions

In Novell documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (**) denotes a third-party trademark.

Overview of File Systems in Linux

1

SUSE® Linux Enterprise Server ships with a number of different file systems from which to choose, including Ext3, Ext2, ReiserFS, and XFS. Each file system has its own advantages and disadvantages. To meet the requirements of high-performance clustering scenarios, SUSE Linux Enterprise Server includes OCFS2 (Oracle Cluster File System 2) in the High-Availability Storage Infrastructure (HASI) release.

- Section 1.1, “Terminology” (page 1)
- Section 1.2, “Major File Systems in Linux” (page 2)
- Section 1.3, “Other Supported File Systems” (page 9)
- Section 1.4, “Large File Support in Linux” (page 10)
- Section 1.5, “Additional Information” (page 12)

1.1 Terminology

metadata

A data structure that is internal to the file system. It assures that all of the on-disk data is properly organized and accessible. Essentially, it is “data about the data.”

Almost every file system has its own structure of metadata, which is on reason that the file systems show different performance characteristics. It is extremely important to maintain metadata intact, because otherwise all data on the file system could become inaccessible.

inode

A data structure on a file system that contains various information about a file, including size, number of links, pointers to the disk blocks where the file contents are actually stored, and date and time of creation, modification, and access.

journal

In the context of a file system, a journal is an on-disk structure containing a type of log in which the file system stores what it is about to change in the file system's metadata. Journaling greatly reduces the recovery time of a file system because it has no need for the lengthy search process that checks the entire file system at system startup. Instead, only the journal is replayed.

1.2 Major File Systems in Linux

SUSE Linux Enterprise Server offers a variety of file systems from which to choose. This section contains an overview of how these file systems work and which advantages they offer.

It is very important to remember that no file system best suits all kinds of applications. Each file system has its particular strengths and weaknesses, which must be taken into account. In addition, even the most sophisticated file system cannot replace a reasonable backup strategy.

The terms *data integrity* and *data consistency*, when used in this section, do not refer to the consistency of the user space data (the data your application writes to its files). Whether this data is consistent must be controlled by the application itself.

IMPORTANT

Unless stated otherwise in this section, all the steps required to set up or change partitions and file systems can be performed by using YaST.

- Section 1.2.1, “Ext2” (page 3)
- Section 1.2.2, “Ext3” (page 4)
- Section 1.2.3, “Oracle Cluster File System 2” (page 6)
- Section 1.2.4, “ReiserFS” (page 6)

- Section 1.2.5, “XFS” (page 8)

1.2.1 Ext2

The origins of Ext2 go back to the early days of Linux history. Its predecessor, the Extended File System, was implemented in April 1992 and integrated in Linux 0.96c. The Extended File System underwent a number of modifications and, as Ext2, became the most popular Linux file system for years. With the creation of journaling file systems and their short recovery times, Ext2 became less important.

A brief summary of Ext2’s strengths might help understand why it was—and in some areas still is—the favorite Linux file system of many Linux users.

- Section “Solidity and Speed” (page 3)
- Section “Easy Upgradability” (page 3)

Solidity and Speed

Being quite an “old-timer,” Ext2 underwent many improvements and was heavily tested. This might be the reason why people often refer to it as rock-solid. After a system outage when the file system could not be cleanly unmounted, `e2fsck` starts to analyze the file system data. Metadata is brought into a consistent state and pending files or data blocks are written to a designated directory (called `lost+found`). In contrast to journaling file systems, `e2fsck` analyzes the entire file system and not just the recently modified bits of metadata. This takes significantly longer than checking the log data of a journaling file system. Depending on file system size, this procedure can take half an hour or more. Therefore, it is not desirable to choose Ext2 for any server that needs high availability. However, because Ext2 does not maintain a journal and uses significantly less memory, it is sometimes faster than other file systems.

Easy Upgradability

Because Ext3 is based on the Ext2 code and shares its on-disk format as well as its metadata format, upgrades from Ext2 to Ext3 are very easy.

1.2.2 Ext3

Ext3 was designed by Stephen Tweedie. Unlike all other next-generation file systems, Ext3 does not follow a completely new design principle. It is based on Ext2. These two file systems are very closely related to each other. An Ext3 file system can be easily built on top of an Ext2 file system. The most important difference between Ext2 and Ext3 is that Ext3 supports journaling. In summary, Ext3 has three major advantages to offer:

- Section “Easy and Highly Reliable Upgrades from Ext2” (page 4)
- Section “Reliability and Performance” (page 4)
- Section “Converting an Ext2 File System into Ext3” (page 5)

Easy and Highly Reliable Upgrades from Ext2

The code for Ext2 is the strong foundation on which Ext3 could become a highly-acclaimed next-generation file system. Its reliability and solidity are elegantly combined in Ext3 with the advantages of a journaling file system. Unlike transitions to other journaling file systems, such as ReiserFS or XFS, which can be quite tedious (making backups of the entire file system and recreating it from scratch), a transition to Ext3 is a matter of minutes. It is also very safe, because re-creating an entire file system from scratch might not work flawlessly. Considering the number of existing Ext2 systems that await an upgrade to a journaling file system, you can easily see why Ext3 might be of some importance to many system administrators. Downgrading from Ext3 to Ext2 is as easy as the upgrade. Just perform a clean unmount of the Ext3 file system and remount it as an Ext2 file system.

Reliability and Performance

Some other journaling file systems follow the “metadata-only” journaling approach. This means your metadata is always kept in a consistent state, but this cannot be automatically guaranteed for the file system data itself. Ext3 is designed to take care of both metadata and data. The degree of “care” can be customized. Enabling Ext3 in the `data=journal` mode offers maximum security (data integrity), but can slow down the system because both metadata and data are journaled. A relatively new approach is to use the `data=ordered` mode, which ensures both data and metadata integrity,

but uses journaling only for metadata. The file system driver collects all data blocks that correspond to one metadata update. These data blocks are written to disk before the metadata is updated. As a result, consistency is achieved for metadata and data without sacrificing performance. A third option to use is `data=writeback`, which allows data to be written into the main file system after its metadata has been committed to the journal. This option is often considered the best in performance. It can, however, allow old data to reappear in files after crash and recovery while internal file system integrity is maintained. Ext3 uses the `data=ordered` option as the default.

Converting an Ext2 File System into Ext3

To convert an Ext2 file system to Ext3:

- 1** Create an Ext3 journal by running `tune2fs -j` as the `root` user.

This creates an Ext3 journal with the default parameters.

To specify how large the journal should be and on which device it should reside, run `tune2fs -J` instead together with the desired journal options `size=` and `device=`. More information about the `tune2fs` program is available in the `tune2fs` man page.

- 2** Edit the file `/etc/fstab` as the `root` user to change the file system type specified for the corresponding partition from `ext2` to `ext3`, then save the changes.

This ensures that the Ext3 file system is recognized as such. The change takes effect after the next reboot.

- 3** To boot a root file system that is set up as an Ext3 partition, include the modules `ext3` and `jbd` in the `initrd`.

3a Edit `/etc/sysconfig/kernel` as `root`, adding `ext3` and `jbd` to the `INITRD_MODULES` variable, then save the changes.

3b Run the `mkinitrd` command.

This builds a new `initrd` and prepares it for use.

4 Reboot the system.

1.2.3 Oracle Cluster File System 2

OCFS2 is a journaling file system that has been tailor-made for clustering setups. In contrast to a standard single-node file system like Ext3, OCFS2 is capable of managing several nodes. OCFS2 allows you to spread a file system across shared storage, such as a SAN or multipath setup.

Every node in an OCFS2 setup has concurrent read and write access to all data. This requires OCFS2 to be cluster-aware, meaning that OCFS2 must include a means to determine which nodes are in the cluster and whether these nodes are actually alive and available. To compute a cluster's membership, OCFS2 includes a node manager. To monitor the availability of the nodes in a cluster, OCFS2 includes a simple heartbeat implementation. To avoid problems arising from various nodes directly accessing the file system, OCFS2 also contains a distributed lock manager. Communication between the nodes is handled via a TCP-based messaging system.

Major features and benefits of OCFS2 include:

- Metadata caching and journaling
- Asynchronous and direct I/O support for database files for improved database performance
- Support for multiple block sizes (where each volume can have a different block size) up to 4 KB, for a maximum volume size of 16 TB
- Cross-node file data consistency
- Support for up to 255 cluster nodes

For more in-depth information about OCFS2, refer to the *High Availability Storage Infrastructure Administration Guide*.

1.2.4 ReiserFS

Officially one of the key features of the 2.4 kernel release, ReiserFS has been available as a kernel patch for 2.2.x SUSE kernels since version 6.4. ReiserFS was designed by

Hans Reiser and the Namesys development team. It has proven itself to be a powerful alternative to Ext2. Its key assets are better disk space utilization, better disk access performance, faster crash recovery, and reliability through data journaling.

- Section “Better Disk Space Utilization” (page 7)
- Section “Better Disk Access Performance” (page 7)
- Section “Fast Crash Recovery” (page 7)
- Section “Reliability through Data Journaling” (page 7)

Better Disk Space Utilization

In ReiserFS, all data is organized in a structure called a B*-balanced tree. The tree structure contributes to better disk space utilization because small files can be stored directly in the B* tree leaf nodes instead of being stored elsewhere and just maintaining a pointer to the actual disk location. In addition to that, storage is not allocated in chunks of 1 or 4 KB, but in portions of the exact size needed. Another benefit lies in the dynamic allocation of inodes. This keeps the file system more flexible than traditional file systems, like Ext2, where the inode density must be specified at file system creation time.

Better Disk Access Performance

For small files, file data and “stat_data” (inode) information are often stored next to each other. They can be read with a single disk I/O operation, meaning that only one access to disk is required to retrieve all the information needed.

Fast Crash Recovery

Using a journal to keep track of recent metadata changes makes a file system check a matter of seconds, even for huge file systems.

Reliability through Data Journaling

ReiserFS also supports data journaling and ordered data modes similar to the concepts outlined in Section 1.2.2, “Ext3” (page 4). The default mode is `data=ordered`, which ensures both data and metadata integrity, but uses journaling only for metadata.

1.2.5 XFS

Originally intended as the file system for their IRIX OS, SGI started XFS development in the early 1990s. The idea behind XFS was to create a high-performance 64-bit journaling file system to meet extreme computing challenges. XFS is very good at manipulating large files and performs well on high-end hardware. However, even XFS has a drawback. Like ReiserFS, XFS takes great care of metadata integrity, but less care of data integrity.

A quick review of XFS's key features explains why it might prove to be a strong competitor for other journaling file systems in high-end computing.

- Section “High Scalability through the Use of Allocation Groups” (page 8)
- Section “High Performance through Efficient Management of Disk Space” (page 8)
- Section “Preallocation to Avoid File System Fragmentation” (page 9)

High Scalability through the Use of Allocation Groups

At the creation time of an XFS file system, the block device underlying the file system is divided into eight or more linear regions of equal size. Those are referred to as *allocation groups*. Each allocation group manages its own inodes and free disk space. Practically, allocation groups can be seen as file systems in a file system. Because allocation groups are rather independent of each other, more than one of them can be addressed by the kernel simultaneously. This feature is the key to XFS's great scalability. Naturally, the concept of independent allocation groups suits the needs of multiprocessor systems.

High Performance through Efficient Management of Disk Space

Free space and inodes are handled by B^+ trees inside the allocation groups. The use of B^+ trees greatly contributes to XFS's performance and scalability. XFS uses *delayed allocation*, which handles allocation by breaking the process into two pieces. A pending transaction is stored in RAM and the appropriate amount of space is reserved. XFS still does not decide where exactly (in file system blocks) the data should be stored. This decision is delayed until the last possible moment. Some short-lived temporary data

might never make its way to disk, because it is obsolete by the time XFS decides where actually to save it. In this way, XFS increases write performance and reduces file system fragmentation. Because delayed allocation results in less frequent write events than in other file systems, it is likely that data loss after a crash during a write is more severe.

Preallocation to Avoid File System Fragmentation

Before writing the data to the file system, XFS *reserves* (preallocates) the free space needed for a file. Thus, file system fragmentation is greatly reduced. Performance is increased because the contents of a file are not distributed all over the file system.

1.3 Other Supported File Systems

Table 1.1, “File System Types in Linux” (page 9) summarizes some other file systems supported by Linux. They are supported mainly to ensure compatibility and interchange of data with different kinds of media or foreign operating systems.

Table 1.1 *File System Types in Linux*

File System Type	Description
<code>cramfs</code>	Compressed ROM file system: A compressed read-only file system for ROMs.
<code>hpfs</code>	High Performance File System: The IBM* OS/2* standard file system. Only supported in read-only mode.
<code>iso9660</code>	Standard file system on CD-ROMs.
<code>minix</code>	This file system originated from academic projects on operating systems and was the first file system used in Linux. Today, it is used as a file system for floppy disks.
<code>msdos</code>	<code>fat</code> , the file system originally used by DOS, is today used by various operating systems.
<code>ncpfs</code>	File system for mounting Novell® volumes over networks.

File System Type	Description
<code>nfs</code>	Network File System: Here, data can be stored on any machine in a network and access might be granted via a network.
<code>smbfs</code>	Server Message Block is used by products such as Windows* to enable file access over a network.
<code>sysv</code>	Used on SCO UNIX*, Xenix, and Coherent (commercial UNIX systems for PCs).
<code>ufs</code>	Used by BSD*, SunOS*, and NextStep*. Only supported in read-only mode.
<code>umsdos</code>	UNIX on MS-DOS*: Applied on top of a standard <code>fat</code> file system, achieves UNIX functionality (permissions, links, long filenames) by creating special files.
<code>vfat</code>	Virtual FAT: Extension of the <code>fat</code> file system (supports long filenames).
<code>ntfs</code>	Windows NT file system; read-only.

1.4 Large File Support in Linux

Originally, Linux supported a maximum file size of 2 GB. This was enough before the explosion of multimedia and as long as no one tried to manipulate huge databases on Linux. Becoming more and more important for server computing, the kernel and C library were modified to support file sizes larger than 2 GB when using a new set of interfaces that applications must use. Today, almost all major file systems offer LFS support, allowing you to perform high-end computing. Table 1.2, “Maximum Sizes of File Systems (On-Disk Format)” (page 11) offers an overview of the current limitations of Linux files and file systems.

Table 1.2 *Maximum Sizes of File Systems (On-Disk Format)*

File System	File Size (Bytes)	File System Size (Bytes)
Ext2 or Ext3 (1 KB block size)	2^{34} (16 GB)	2^{41} (2 TB)
Ext2 or Ext3 (2 KB block size)	2^{38} (256 GB)	2^{43} (8 TB)
Ext2 or Ext3 (4 KB block size)	2^{41} (2 TB)	2^{44} -4096 (16 TB-4096 Bytes)
Ext2 or Ext3 (8 KB block size) (systems with 8 KB pages, like Alpha)	2^{46} (64 TB)	2^{45} (32 TB)
ReiserFS v3	2^{46} (64 TB)	2^{45} (32 TB)
XFS	2^{63} (8 EB)	2^{63} (8 EB)
NFSv2 (client side)	2^{31} (2 GB)	2^{63} (8 EB)
NFSv3 (client side)	2^{63} (8 EB)	2^{63} (8 EB)

IMPORTANT

Table 1.2, “Maximum Sizes of File Systems (On-Disk Format)” (page 11) describes the limitations regarding the on-disk format. The 2.6 Linux kernel imposes its own limits on the size of files and file systems handled by it. These are as follows:

File Size

On 32-bit systems, files cannot exceed 2 TB (2^{41} bytes).

File System Size

File systems can be up to 2^{73} bytes in size. However, this limit is still out of reach for the currently available hardware.

1.5 Additional Information

The *File System Primer* [http://wiki.novell.com/index.php/File_System_Primer] on the Novell Web site describes a variety of file systems for Linux. It discusses the file systems, why there are so many, and which ones are the best to use for which workloads and data.

Each of the file system projects described above maintains its own home page on which to find mailing list information, further documentation, and FAQs:

- *E2fsprogs: Ext2/3/4 Filesystem Utilities* [<http://e2fsprogs.sourceforge.net/>]
- *Introducing Ext3* [<http://www.ibm.com/developerworks/linux/library/l-fs7.html>]
- *ReiserFSprogs* [http://chichkin_i.zelnet.ru/namesys/]
- *XFS: A High-Performance Journaling Filesystem* [<http://oss.sgi.com/projects/xfst/>]
- *OCFS2 Project* [<http://oss.oracle.com/projects/ocfs2/>]

A comprehensive multipart tutorial about Linux file systems can be found at IBM developerWorks in the *Advanced Filesystem Implementor's Guide* [<http://www-106.ibm.com/developerworks/library/l-fs.html>]. An in-depth comparison of file systems (not only Linux file systems) is available from the Wikipedia project in *Comparison of File Systems* [http://en.wikipedia.org/wiki/Comparison_of_file_systems#Comparison].

What's New

The features and behavior changes noted in this section were made for SUSE® Linux Enterprise Server 11.

- Section 2.1, “What’s New in SLES 11 SP1” (page 15)
- Section 2.2, “What’s New in SLES 11” (page 20)

2.1 What's New in SLES 11 SP1

In addition to bug fixes, the features and behavior changes noted in this section were made for the SUSE® Linux Enterprise Server 11 SP1 release.

- Section 2.1.1, “Saving iSCSI Target Information” (page 16)
- Section 2.1.2, “Modifying Authentication Parameters in the iSCSI Initiator” (page 16)
- Section 2.1.3, “Allowing Persistent Reservations for MPIO Devices” (page 16)
- Section 2.1.4, “MDADM 3.0.2” (page 17)
- Section 2.1.5, “Boot Loader Support for MDRAID External Metadata” (page 17)
- Section 2.1.6, “YaST Install and Boot Support for MDRAID External Metadata” (page 17)

- Section 2.1.7, “Improved Shutdown for MDRAID Arrays that Contain the Root File System” (page 18)
- Section 2.1.8, “MD over iSCSI Devices” (page 18)
- Section 2.1.9, “MD-SGPIO” (page 19)
- Section 2.1.10, “Resizing LVM 2 Mirrors” (page 19)
- Section 2.1.11, “Updating Storage Drivers for Adapters on IBM Servers” (page 19)

2.1.1 Saving iSCSI Target Information

In the *YaSTNetwork ServicesiSCSI Target* function, a *Save* option was added that allows you to export the iSCSI target information. This makes it easier to provide information to consumers of the resources.

2.1.2 Modifying Authentication Parameters in the iSCSI Initiator

In the *YaSTNetwork ServicesiSCSI Initiator* function, you can modify the authentication parameters for connecting to a target devices. Previously, you needed to delete the entry and re-create it in order to change the authentication information.

2.1.3 Allowing Persistent Reservations for MPIO Devices

A SCSI initiator can issue SCSI reservations for a shared storage device, which locks out SCSI initiators on other servers from accessing the device. These reservations persist across SCSI resets that might happen as part of the SCSI exception handling process.

The following are possible scenarios where SCSI reservations would be useful:

- In a simple SAN environment, persistent SCSI reservations help protect against administrator errors where a LUN is attempted to be added to one server but it is

already in use by another server, which might result in data corruption. SAN zoning is typically used to prevent this type of error.

- In a high-availability environment with failover set up, persistent SCSI reservations help protect against errant servers connecting to SCSI devices that are reserved by other servers.

2.1.4 MDADM 3.0.2

Use the latest version of the Multiple Devices Administration (MDADM, `mdadm`) utility to take advantage of bug fixes and improvements.

2.1.5 Boot Loader Support for MDRAID External Metadata

Support was added to use the external metadata capabilities of the MDADM utility version 3.0 to install and run the operating system from RAID volumes defined by the Intel* Matrix Storage Technology metadata format. This moves the functionality from the Device Mapper RAID (DMRAID) infrastructure to the Multiple Devices RAID (MDRAID) infrastructure, which offers the more mature RAID 5 implementation and offers a wider feature set of the MD kernel infrastructure. It allows a common RAID driver to be used across all metadata formats, including Intel, DDF (common RAID disk data format), and native MD metadata.

2.1.6 YaST Install and Boot Support for MDRAID External Metadata

The YaST® installer tool added support for MDRAID External Metadata for RAID 0, 1, 10, 5, and 6. The installer can detect RAID arrays and whether the platform RAID capabilities are enabled. If RAID is enabled in the platform BIOS for Intel Matrix Storage Manager, it offers options for DMRAID, MDRAID (recommended), or none. The `initrd` was also modified to support assembling BIOS-based RAID arrays.

2.1.7 Improved Shutdown for MDRAID Arrays that Contain the Root File System

Shutdown scripts were modified to wait until all of the MDRAID arrays are marked clean. The operating system shutdown process now waits for a dirty-bit to be cleared until all MDRAID volumes have finished write operations.

Changes were made to the startup script, shutdown script, and the `initrd` to consider whether the root (`/`) file system (the system volume that contains the operating system and application files) resides on a software RAID array. The metadata handler for the array is started early in the shutdown process to monitor the final root file system environment during the shutdown. The handler is excluded from the general `killall` events. The process also allows for writes to be quiesced and for the array's metadata dirty-bit (which indicates whether an array needs to be resynchronized) to be cleared at the end of the shutdown.

2.1.8 MD over iSCSI Devices

The YaST installer now allows MD to be configured over iSCSI devices.

If RAID arrays are needed on boot, the iSCSI initiator software is loaded before `boot.md` so that the iSCSI targets are available to be auto-configured for the RAID.

For a new install, Libstorage creates an `/etc/mdadm.conf` file and adds the line `AUTO -all`. During an update, the line is not added. If `/etc/mdadm.conf` contains the line

```
AUTO -all
```

then no RAID arrays are auto-assembled unless they are explicitly listed in `/etc/mdadm.conf`.

2.1.9 MD-SGPIO

The MD-SGPIO utility is a standalone application that monitors RAID arrays via `sysfs(2)`. Events trigger an LED change request that controls blinking for LED lights that are associated with each slot in an enclosure or a drive bay of a storage subsystem. It supports two types of LED systems:

- 2-LED systems (Activity LED, Status LED)
- 3-LED systems (Activity LED, Locate LED, Fail LED)

2.1.10 Resizing LVM 2 Mirrors

The `lvresize`, `lvextend`, and `lvreduce` commands that are used to resize logical volumes were modified to allow the resizing of LVM 2 mirrors. Previously, these commands reported errors if the logical volume was a mirror.

2.1.11 Updating Storage Drivers for Adapters on IBM Servers

Update the following storage drivers to use the latest available versions to support storage adapters on IBM servers:

- Adaptec™: `aacraid`, `aic94xx`
- Emulex™: `lpfc`
- LSI™: `mptas`, `megaraid_sas`

The `mptsas` driver now supports native EEH (Enhanced Error Handler) recovery, which is a key feature for all of the IO devices for Power platform customers.

- qLogic™: `qla2xxx`, `qla3xxx`, `qla4xxx`

2.2 What's New in SLES 11

The features and behavior changes noted in this section were made for the SUSE® Linux Enterprise Server 11 release.

- Section 2.2.1, “EVMS2 Is Deprecated” (page 20)
- Section 2.2.2, “Ext3 as the Default File System” (page 21)
- Section 2.2.3, “JFS File System Is Deprecated” (page 21)
- Section 2.2.4, “OCFS2 File System Is in the High Availability Release” (page 21)
- Section 2.2.5, “/dev/disk/by-name Is Deprecated” (page 21)
- Section 2.2.6, “Device Name Persistence in the /dev/disk/by-id Directory” (page 22)
- Section 2.2.7, “Filters for Multipathed Devices” (page 22)
- Section 2.2.8, “User-Friendly Names for Multipathed Devices” (page 23)
- Section 2.2.9, “Advanced I/O Load-Balancing Options for Multipath” (page 23)
- Section 2.2.10, “Location Change for Multipath Tool Callouts” (page 23)
- Section 2.2.11, “Change from mpath to multipath for the mkinitrd -f Option” (page 24)

2.2.1 EVMS2 Is Deprecated

The Enterprise Volume Management Systems (EVMS2) storage management solution is deprecated. All EVMS management modules have been removed from the SUSE Linux Enterprise Server 11 packages. Your EVMS-managed devices should be automatically recognized and managed by Linux Volume Manager 2 (LVM2) when you upgrade your system. For more information, see *Evolution of Storage and Volume Management in SUSE Linux Enterprise* [<http://www.novell.com/linux/volumemanagement/strategy.html>].

For information about managing storage with EVMS2 on SUSE Linux Enterprise Server 10, see the *SUSE Linux Enterprise Server 10 SP3: Storage Administration Guide* [http://www.novell.com/documentation/sles10/stor_admin/data/bookinfo.html].

2.2.2 Ext3 as the Default File System

The Ext3 file system has replaced ReiserFS as the default file system recommended by the YaST tools at installation time and when you create file systems. ReiserFS is still supported. For more information, see *File System Future Directions* [<http://www.novell.com/linux/techspecs.html?tab=0>] on the *SUSE Linux Enterprise 10 File System Support* Web page.

2.2.3 JFS File System Is Deprecated

The JFS file system is no longer supported. The JFS utilities were removed from the distribution.

2.2.4 OCFS2 File System Is in the High Availability Release

The OCFS2 file system is fully supported as part of the SUSE Linux Enterprise High Availability Extension.

2.2.5 /dev/disk/by-name Is Deprecated

The `/dev/disk/by-name` path is deprecated in SUSE Linux Enterprise Server 11 packages.

2.2.6 Device Name Persistence in the /dev/disk/by-id Directory

In SUSE Linux Enterprise Server 11, the default multipath setup relies on `udev` to overwrite the existing symbolic links in the `/dev/disk/by-id` directory when multipathing is started. Before you start multipathing, the link points to the SCSI device by using its `scsi-xxx` name. When multipathing is running, the symbolic link points to the device by using its `dm-uuid-xxx` name. This ensures that the symbolic links in the `/dev/disk/by-id` path persistently point to the same device regardless of whether multipathing is started or not. The configuration files (such as `lvm.conf` and `md.conf`) do not need to be modified because they automatically point to the correct device.

See the following sections for more information about how this behavior change affects other features:

- Section 2.2.7, “Filters for Multipathed Devices” (page 22)
- Section 2.2.8, “User-Friendly Names for Multipathed Devices” (page 23)

2.2.7 Filters for Multipathed Devices

The deprecation of the `/dev/disk/by-name` directory (as described in Section 2.2.5, “`/dev/disk/by-name` Is Deprecated” (page 21)) affects how you set up filters for multipathed devices in the configuration files. If you used the `/dev/disk/by-name` device name path for the multipath device filters in the `/etc/lvm/lvm.conf` file, you need to modify the file to use the `/dev/disk/by-id` path. Consider the following when setting up filters that use the `by-id` path:

- The `/dev/disk/by-id/scsi-*` device names are persistent and created for exactly this purpose.
- Do not use the `/dev/disk/by-id/dm-*` name in the filters. These are symbolic links to the Device-Mapper devices, and result in reporting duplicate PVs in response to a `pvscan` command. The names appear to change from `LVM-pvuuid` to `dm-uuid` and back to `LVM-pvuuid`.

For information about setting up filters, see Section 7.2.3, “Using LVM2 on Multipath Devices” (page 60).

2.2.8 User-Friendly Names for Multipath Devices

A change in how multipath device names are handled in the `/dev/disk/by-id` directory (as described in Section 2.2.6, “Device Name Persistence in the `/dev/disk/by-id` Directory” (page 22)) affects your setup for user-friendly names because the two names for the device differ. You must modify the configuration files to scan only the device mapper names after multipathing is configured.

For example, you need to modify the `lvm.conf` file to scan using the multipath device names by specifying the `/dev/disk/by-id/dm-uuid-.*-mpath-.*` path instead of `/dev/disk/by-id`.

2.2.9 Advanced I/O Load-Balancing Options for Multipath

The following advanced I/O load-balancing options are available for Device Mapper Multipath, in addition to round-robin:

- Least-pending
- Length-load-balancing
- Service-time

For information, see Section “Understanding Priority Groups and Attributes” (page 88).

2.2.10 Location Change for Multipath Tool Callouts

The `mpath_*` `prio_callouts` for the Device Mapper Multipath tool have been moved to shared libraries in `/lib/libmultipath/lib*`. By using shared libraries, the

callouts are loaded into memory on daemon startup. This helps avoid a system deadlock on an all-paths-down scenario where the programs need to be loaded from the disk, which might not be available at this point.

2.2.11 Change from mpath to multipath for the mkinitrd -f Option

The option for adding Device Mapper Multipath services to the `initrd` has changed from `-f mpath` to `-f multipath`.

To make a new `initrd`, the command is now:

```
mkinitrd -f multipath
```


3

Planning a Storage Solution

Consider what your storage needs are and how you can effectively manage and divide your storage space to best meet your needs. Use the information in this section to help plan your storage deployment for file systems on your SUSE® Linux Enterprise Server 11 server.

- Section 3.1, “Partitioning Devices” (page 25)
- Section 3.2, “Multipath Support” (page 25)
- Section 3.3, “Software RAID Support” (page 26)
- Section 3.4, “File System Snapshots” (page 26)
- Section 3.5, “Backup and Antivirus Support” (page 26)

3.1 Partitioning Devices

For information about using the YaST Expert Partitioner, see “Using the YaST Partitioner” in the *SUSE Linux Enterprise Server 11 Installation and Administration Guide*.

3.2 Multipath Support

Linux supports using multiple I/O paths for fault-tolerant connections between the server and its storage devices. Linux multipath support is disabled by default. If you

use a multipath solution that is provided by your storage subsystem vendor, you do not need to configure the Linux multipath separately.

3.3 Software RAID Support

Linux supports hardware and software RAID devices. If you use hardware RAID devices, software RAID devices are unnecessary. You can use both hardware and software RAID devices on the same server.

To maximize the performance benefits of software RAID devices, partitions used for the RAID should come from different physical devices. For software RAID 1 devices, the mirrored partitions cannot share any disks in common.

3.4 File System Snapshots

Linux supports file system snapshots.

3.5 Backup and Antivirus Support

- Section 3.5.1, “Open Source Backup” (page 26)
- Section 3.5.2, “Commercial Backup and Antivirus Support” (page 27)

3.5.1 Open Source Backup

Open source tools for backing up data on Linux include `tar`, `cpio`, and `rsync`. See the man pages for these tools for more information.

- PAX: POSIX File System Archiver. It supports `cpio` and `tar`, which are the two most common forms of standard archive (backup) files. See the man page for more information.
- Amanda: The Advanced Maryland Automatic Network Disk Archiver. See www.amanda.org [<http://www.amanda.org/>].

3.5.2 Commercial Backup and Antivirus Support

Novell® Open Enterprise Server (OES) 2 Support Pack 1 for Linux is a product that includes SUSE Linux Enterprise Server (SLES) 10 Support Pack 2. Antivirus and backup software vendors who support OES 2 SP1 also support SLES 10 SP2. You can visit the vendor Web sites to find out about their scheduled support of SLES 11.

For a current list of possible backup and antivirus software vendors, see *Novell Open Enterprise Server Partner Support: Backup and Antivirus Support* [http://www.novell.com/products/openenterpriseserver/partners_communities.html]. This list is updated quarterly.

LVM Configuration

This section briefly describes the principles behind Logical Volume Manager (LVM) and its basic features that make it useful under many circumstances. The YaST LVM configuration can be reached from the YaST Expert Partitioner. This partitioning tool enables you to edit and delete existing partitions and create new ones that should be used with LVM.

WARNING

Using LVM might be associated with increased risk, such as data loss. Risks also include application crashes, power failures, and faulty commands. Save your data before implementing LVM or reconfiguring volumes. Never work without a backup.

- Section 4.1, “Understanding the Logical Volume Manager” (page 30)
- Section 4.2, “Creating LVM Partitions” (page 32)
- Section 4.3, “Creating Volume Groups” (page 32)
- Section 4.4, “Configuring Physical Volumes” (page 33)
- Section 4.5, “Configuring Logical Volumes” (page 34)
- Section 4.6, “Direct LVM Management” (page 36)
- Section 4.7, “Resizing an LVM Partition” (page 36)

4.1 Understanding the Logical Volume Manager

LVM enables flexible distribution of hard disk space over several file systems. It was developed because the need to change the segmentation of hard disk space might arise only after the initial partitioning has already been done during installation. Because it is difficult to modify partitions on a running system, LVM provides a virtual pool (volume group or VG) of memory space from which logical volumes (LVs) can be created as needed. The operating system accesses these LVs instead of the physical partitions. Volume groups can span more than one disk, so that several disks or parts of them can constitute one single VG. In this way, LVM provides a kind of abstraction from the physical disk space that allows its segmentation to be changed in a much easier and safer way than through physical repartitioning.

Figure 4.1 *Physical Partitioning versus LVM*

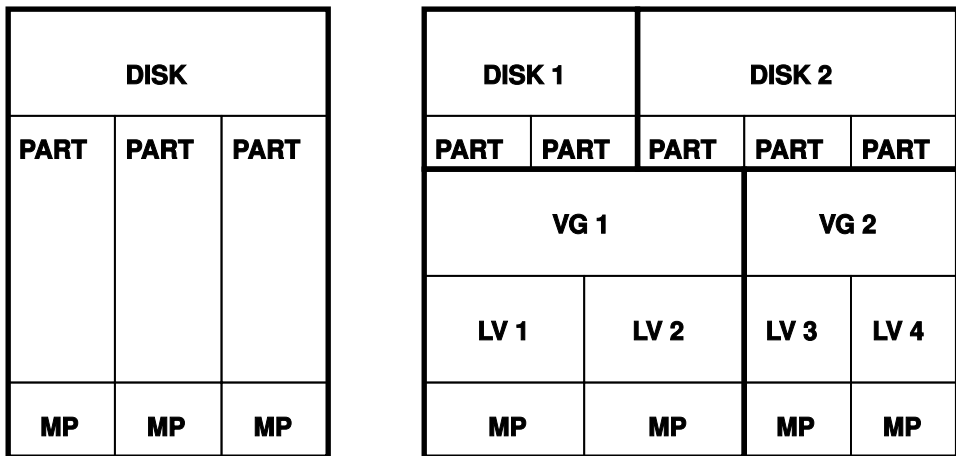


Figure 4.1, “Physical Partitioning versus LVM” (page 30) compares physical partitioning (left) with LVM segmentation (right). On the left side, one single disk has been divided into three physical partitions (PART), each with a mount point (MP) assigned so that the operating system can access them. On the right side, two disks have been divided into two and three physical partitions each. Two LVM volume groups (VG 1 and VG 2) have been defined. VG 1 contains two partitions from DISK 1 and one from DISK 2. VG 2 contains the remaining two partitions from DISK 2. In LVM, the physical disk

partitions that are incorporated in a volume group are called physical volumes (PVs). Within the volume groups, four logical volumes (LV 1 through LV 4) have been defined, which can be used by the operating system via the associated mount points. The border between different logical volumes need not be aligned with any partition border. See the border between LV 1 and LV 2 in this example.

LVM features:

- Several hard disks or partitions can be combined in a large logical volume.
- Provided the configuration is suitable, an LV (such as `/usr`) can be enlarged when the free space is exhausted.
- Using LVM, it is possible to add hard disks or LVs in a running system. However, this requires hot-swappable hardware that is capable of such actions.
- It is possible to activate a *striping mode* that distributes the data stream of a logical volume over several physical volumes. If these physical volumes reside on different disks, this can improve the reading and writing performance just like RAID 0.
- The snapshot feature enables consistent backups (especially for servers) in the running system.

With these features, using LVM already makes sense for heavily used home PCs or small servers. If you have a growing data stock, as in the case of databases, music archives, or user directories, LVM is especially useful. It allows file systems that are larger than the physical hard disk. Another advantage of LVM is that up to 256 LVs can be added. However, keep in mind that working with LVM is different from working with conventional partitions. Instructions and further information about configuring LVM is available in the official *LVM HOWTO* [<http://tldp.org/HOWTO/LVM-HOWTO/>].

Starting from kernel version 2.6, LVM version 2 is available, which is downward-compatible with the previous LVM and enables the continued management of old volume groups. When creating new volume groups, decide whether to use the new format or the downward-compatible version. LVM 2 does not require any kernel patches. It makes use of the device mapper integrated in kernel 2.6. This kernel only supports LVM version 2. Therefore, when talking about LVM, this section always refers to LVM version 2.

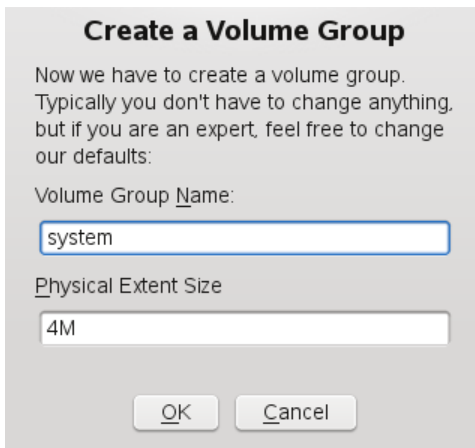
4.2 Creating LVM Partitions

You create an LVM partition by first clicking *Create > Do not format* then selecting *0x8E Linux LVM* as the partition identifier. After creating all the partitions to use with LVM, click *LVM* to start the LVM configuration.

4.3 Creating Volume Groups

If no volume group exists on your system yet, you are prompted to add one (see Figure 4.2, “Creating a Volume Group” (page 32)). It is possible to create additional groups with *Add group*, but usually one single volume group is sufficient. `system` is suggested as a name for the volume group in which the SUSE® Linux Enterprise Server system files are located. The physical extent size defines the size of a physical block in the volume group. All the disk space in a volume group is handled in chunks of this size. This value is normally set to 4 MB and allows for a maximum size of 256 GB for physical and logical volumes. The physical extent size should only be increased, for example, to 8, 16, or 32 MB, if you need logical volumes larger than 256 GB.

Figure 4.2 *Creating a Volume Group*



Create a Volume Group

Now we have to create a volume group.
Typically you don't have to change anything,
but if you are an expert, feel free to change
our defaults:

Volume Group Name:

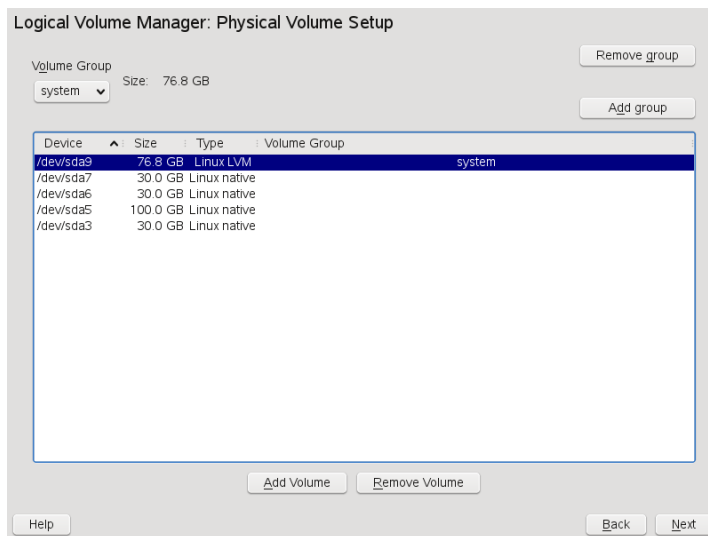
Physical Extent Size

4.4 Configuring Physical Volumes

After a volume group has been created, the next dialog (see Figure 4.3, “Physical Volume Setup” (page 33)) lists all partitions with either the “Linux LVM” or “Linux native” type. No swap or DOS partitions are shown. If a partition is already assigned to a volume group, the name of the volume group is shown in the list. Unassigned partitions are indicated with “--”.

If there are several volume groups, set the current volume group in the selection box to the upper left. The buttons in the upper right enable creation of additional volume groups and deletion of existing volume groups. Only volume groups that do not have any partitions assigned can be deleted. All partitions that are assigned to a volume group are also referred to as a physical volumes.

Figure 4.3 *Physical Volume Setup*

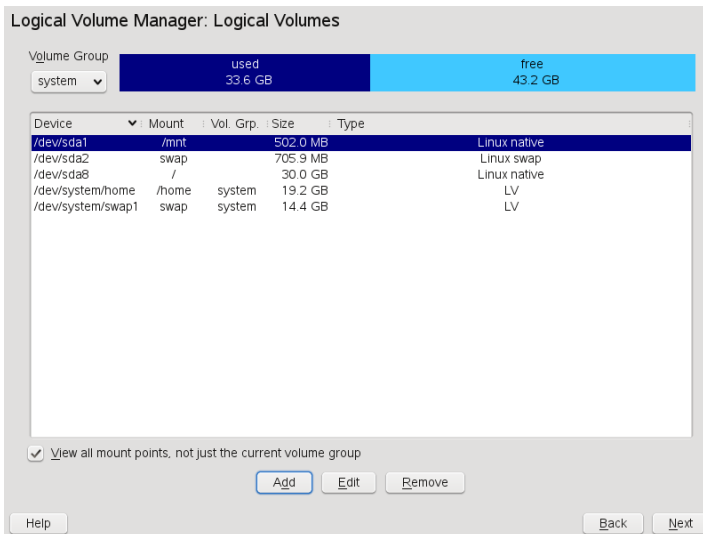


To add a previously unassigned partition to the selected volume group, first click the partition, then click *Add Volume*. At this point, the name of the volume group is entered next to the selected partition. Assign all partitions reserved for LVM to a volume group. Otherwise, the space on the partition remains unused. Before exiting the dialog, every volume group must be assigned at least one physical volume. After assigning all physical volumes, click *Next* to proceed to the configuration of logical volumes.

4.5 Configuring Logical Volumes

After the volume group has been filled with physical volumes, use the next dialog (see Figure 4.4, “Logical Volume Management” (page 34)) to define the logical volumes the operating system should use. Set the current volume group in a selection box to the upper left. Next to it, the free space in the current volume group is shown. The list below contains all logical volumes in that volume group. All normal Linux partitions to which a mount point is assigned, all swap partitions, and all already existing logical volumes are listed here. You can use *Add*, *Edit*, and *Remove* options to manage the logical volumes as needed until all space in the volume group has been exhausted. Assign at least one logical volume to each volume group.

Figure 4.4 *Logical Volume Management*



To create a new logical volume (see Figure 4.5, “Creating Logical Volumes” (page 35)), click *Add* and fill out the pop-up that opens. For partitioning, specify the size, file system, and mount point. Normally, a file system, such as Reiserfs or Ext2, is created on a logical volume and is then designated a mount point. The files stored on this logical volume can be found at this mount point on the installed system. Additionally it is possible to distribute the data stream in the logical volume among several physical volumes (striping). If these physical volumes reside on different hard disks, this generally results in a better reading and writing performance (like RAID 0). However, a

striping LV with n stripes can only be created correctly if the hard disk space required by the LV can be distributed evenly to n physical volumes. If, for example, only two physical volumes are available, a logical volume with three stripes is impossible.

WARNING

YaST has no chance at this point to verify the correctness of your entries concerning striping. Any mistake made here is apparent only later when the LVM is implemented on disk.

Figure 4.5 *Creating Logical Volumes*

Create Logical Volume

Logical volume name
home
(e.g. var, opt)

Size: (e.g., 4.0 GB 210.0 MB)
10.8 GB
max = 43.2 GB max

Stripes
1

Stripe Size
64

Fstab Options

Mount Point
/home

Format

Do not format

Format

File system
Reiser

Options

Encrypt file system

OK Cancel

If you have already configured LVM on your system, the existing logical volumes can be specified now. Before continuing, assign appropriate mount points to these logical volumes too. Click *Next* to return to the YaST Expert Partitioner and finish your work there.

4.6 Direct LVM Management

If you already have configured LVM and only want to change something, there is an alternative method available. In the YaST Control Center, select *System > Partitioner*. You can manage your LVM system by using the methods already described.

4.7 Resizing an LVM Partition

The `lvresize`, `lvextend`, and `lvreduce` commands are used to resize logical volumes. See the man pages for each of these commands for syntax and options information.

You can also increase the size of a logical volume by using the YaST partitioner. YaST uses `parted(8)` to grow the partition.

To extend an LV there must be enough unallocated space available on the VG.

LVs can be extended or shrunk while they are being used, but this may not be true for a filesystem on them. Extending or shrinking the LV does not automatically modify the size of file systems in the volume. You must use a different command to grow the filesystem afterwards. For information about resizing file systems, see Chapter 5, *Resizing File Systems* (page 39).

Make sure you use the right sequence:

- If you extend an LV, you must extend the LV before you attempt to grow the filesystem.
- If you shrink an LV, you must shrink the filesystem before you attempt to shrink the LV.

To extend the size of a logical volume:

- 1 Open a terminal console, log in as the `root` user.
- 2 If the logical volume contains file systems that are hosted for a virtual machine (such as a Xen VM), shut down the VM.

3 Dismount the file systems on the logical volume.

4 At the terminal console prompt, enter the following command to grow the size of the logical volume:

```
lvextend -L +size /dev/vgname/lvname
```

For *size*, specify the amount of space you want to add to the logical volume, such as 10GB. Replace */dev/vgname/lvname* with the Linux path to the logical volume, such as */dev/vg1/v1*. For example:

```
lvextend -L +10GB /dev/vg1/v1
```

For example, to extend an LV with a (mounted and active) ReiserFS on it by 10GB:

```
lvextend -L +10G /dev/vgname/lvname  
resize_reiserfs -s +10GB -f /dev/vg-name/lv-name
```

For example, to shrink an LV with a ReiserFS on it by 5GB:

```
umount /mountpoint-of-LV  
resize_reiserfs -s -5GB /dev/vgname/lvname  
lvreduce /dev/vgname/lvname  
mount /dev/vgname/lvname /mountpoint-of-LV
```


Resizing File Systems

When your data needs grow for a volume, you might need to increase the amount of space allocated to its file system.

- Section 5.1, “Guidelines for Resizing” (page 39)
- Section 5.2, “Increasing an Ext2 or Ext3 File System” (page 41)
- Section 5.3, “Increasing the Size of a Reiser File System” (page 42)
- Section 5.4, “Decreasing the Size of an Ext2 or Ext3 File System” (page 43)
- Section 5.5, “Decreasing the Size of a Reiser File System” (page 44)

5.1 Guidelines for Resizing

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.

Consider the following guidelines when planning to resize a file system.

- Section 5.1.1, “File Systems that Support Resizing” (page 40)

- Section 5.1.2, “Increasing the Size of a File System” (page 40)
- Section 5.1.3, “Decreasing the Size of a File System” (page 41)

5.1.1 File Systems that Support Resizing

The file system must support resizing in order to take advantage of increases in available space for the volume. In SUSE® Linux Enterprise Server 11, file system resizing utilities are available for file systems Ext2, Ext3, and ReiserFS. The utilities support increasing and decreasing the size as follows:

Table 5.1 *File System Support for Resizing*

File System	Utility	Increase Size (Grow)	Decrease Size (Shrink)
Ext2	resize2fs	Yes, offline only	Yes, offline only
Ext3	resize2fs	Yes, online or of-line	Yes, online or of-line
ReiserFS	resize_reiserfs	Yes, online or of-line	Yes, offline only

5.1.2 Increasing the Size of a File System

You can grow a file system to the maximum space available on the device, or specify an exact size. Make sure to grow the size of the device or logical volume before you attempt to increase the size of the file system.

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 device size because the file system size cannot extend beyond the space available.

5.1.3 Decreasing the Size of a File System

When decreasing the size of the file system on a 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 device size because the file system size cannot extend beyond the space available.

If you plan to also decrease the size of the logical volume that holds the file system, make sure to decrease the size of the file system before you attempt to decrease the size of the device or logical volume.

5.2 Increasing an Ext2 or Ext3 File System

Ext2 and Ext3 file systems can be resized when mounted or unmounted with the `resize2fs` command.

- 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 device called `/dev/sda1`, enter

```
resize2fs /dev/sda1
```

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/sda1 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 can 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 device named `/dev/sda1` at mount point `/home`, enter

```
mount -t ext2 /dev/sda1 /home
```

- 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.

5.3 Increasing the Size of a Reiser File System

A ReiserFS file system can be increased in size while mounted or unmounted.

- 1 Open a terminal console, then log in as the `root` user or equivalent.
- 2 Increase the size of the file system on the device called `/dev/sda2`, using one of the following methods:

- To extend the file system size to the maximum available size of the device, enter

```
resize_reiserfs /dev/sda2
```

When no size is specified, this increases the volume to the full size of the partition.

- To extend the file system to a specific size, enter

```
resize_reiserfs -s size /dev/sda2
```

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/sda2` by 500 MB:

```
resize_reiserfs -s +500M /dev/sda2
```

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 device `/dev/sda2` at mount point `/home`, enter

```
mount -t reiserfs /dev/sda2 /home
```

- 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.

5.4 Decreasing the Size of an Ext2 or Ext3 File System

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 device such as `/dev/sda1` by entering

```
resize2fs /dev/sda1 <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 device named `/dev/sda1` at mount point `/home`, enter

```
mount -t ext2 /dev/md0 /home
```

- 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.

5.5 Decreasing the Size of a Reiser File System

Reiser file systems can be reduced in size 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 decrease in size 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 a device called `/dev/sda1` by entering

```
resize_reiserfs -s size /dev/sda2
```

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/sda2
```

Wait until the resizing is completed before continuing.

4 Mount the file system by entering

```
mount -t reiserfs /dev/sda2 /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.

Using UUIDs to Mount Devices

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.

- Section 6.1, “Naming Devices with udev” (page 47)
- Section 6.2, “Understanding UUIDs” (page 48)
- Section 6.3, “Using UUIDs in the Boot Loader and `/etc/fstab` File (x86)” (page 49)
- Section 6.4, “Using UUIDs in the Boot Loader and `/etc/fstab` File (IA64)” (page 52)
- Section 6.5, “Additional Information” (page 53)

6.1 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 to 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`.

6.2 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 6.2.1, “Using UUIDs to Assemble or Activate File System Devices” (page 48)
- Section 6.2.2, “Finding the UUID for a File System Device” (page 49)

6.2.1 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 manufacturer 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 file system 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`) 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.

6.2.2 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
```

6.3 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.

Before you begin, make a copy of `/boot/grub/menu.lst` file and the `/etc/fstab` file.

- 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.lst` 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

If you make a mistake, you can boot the server without the SAN connected, and fix the error by using the backup copy of the `/boot/grub/menu.lst` file as a guide.

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

Do not leave stray characters or spaces in the file.

6.4 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`.

Before you begin, make a copy of the `/boot/efi/SuSE/elilo.conf` file.

- 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

Do not leave stray characters or spaces in the file.

6.5 Additional Information

For more information about using `udev` (8) for managing devices, see “Dynamic Kernel Device Management with `udev`” [http://www.novell.com/documentation/sles11/book_sle_admin/data/cha_udev.html] in the *SUSE® Linux Enterprise Server 11 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
```


Managing Multipath I/O for Devices

7

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

- Section 7.1, “Understanding Multipathing” (page 56)
- Section 7.2, “Planning for Multipathing” (page 57)
- Section 7.3, “Multipath Management Tools” (page 67)
- Section 7.4, “Configuring the System for Multipathing” (page 75)
- Section 7.5, “Enabling and Starting Multipath I/O Services” (page 86)
- Section 7.6, “Configuring Path Failover Policies and Priorities” (page 86)
- Section 7.7, “Tuning the Failover for Specific Host Bus Adapters” (page 100)
- Section 7.8, “Configuring Multipath I/O for the Root Device” (page 100)
- Section 7.9, “Configuring Multipath I/O for an Existing Software RAID” (page 105)
- Section 7.10, “Scanning for New Devices without Rebooting” (page 108)
- Section 7.11, “Scanning for New Partitioned Devices without Rebooting” (page 111)
- Section 7.12, “Viewing Multipath I/O Status” (page 113)
- Section 7.13, “Managing I/O in Error Situations” (page 114)

- Section 7.14, “Resolving Stalled I/O” (page 116)
- Section 7.15, “Additional Information” (page 116)
- Section 7.16, “What’s Next” (page 117)

7.1 Understanding Multipathing

- Section 7.1.1, “What Is Multipathing?” (page 56)
- Section 7.1.2, “Benefits of Multipathing” (page 56)

7.1.1 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.

7.1.2 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.

7.2 Planning for Multipathing

- Section 7.2.1, “Guidelines for Multipathing” (page 57)
- Section 7.2.2, “Using By-ID Names for Multipathed Devices” (page 59)
- Section 7.2.3, “Using LVM2 on Multipath Devices” (page 60)
- Section 7.2.4, “Using mdadm with Multipath Devices” (page 61)
- Section 7.2.5, “Using --noflush with Multipath Devices” (page 62)
- Section 7.2.6, “SAN Timeout Settings When the Root Device Is Multipathed” (page 62)
- Section 7.2.7, “Partitioning Multipath Devices” (page 63)
- Section 7.2.8, “Supported Architectures for Multipath I/O” (page 64)
- Section 7.2.9, “Supported Storage Arrays for Multipathing” (page 64)

7.2.1 Guidelines for Multipathing

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

- Section “Prerequisites” (page 58)
- Section “Vendor-Provided Multipath Solutions” (page 58)
- Section “Disk Management Tasks” (page 58)
- Section “Software RAIDs” (page 59)
- Section “High-Availability Solutions” (page 59)
- Section “Volume Managers” (page 59)
- Section “Virtualization Environments” (page 59)

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 7.2.9, “Supported Storage Arrays for Multipathing” (page 64).
- 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 multipathing for the logical device as seen by the server.

Vendor-Provided Multipath Solutions

For some storage arrays, the vendor provides 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-MPIO 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 DRBD 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.

7.2.2 Using By-ID Names for Multipathed Devices

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

If the `user-friendly names` option is enabled in the `/etc/multipath.conf` file, you can use the `/dev/disk/by-id/dm-uuid-.*-mpath-.*` device name because this name is aliased to the device ID. For information, see Section “Configuring User-Friendly Names or Alias Names in `/etc/multipath.conf`” (page 81).

7.2.3 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 so that it does not scan and use the physical paths, but only accesses the multipath I/O storage through the multipath I/O layer. If you are using user-friendly names, make sure to specify the path so that it scans only the device mapper names for the device (`/dev/disk/by-id/dm-uuid-.*-mpath-.*`) after multipathing is configured.

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

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

If `/etc/lvm/lvm.conf` does not exist, you can create one based on your current LVM configuration by entering the following at a terminal console prompt:

```
lvm dumpconfig > /etc/lvm/lvm.conf
```

- 2 Change the `filter` and `types` entries in `/etc/lvm/lvm.conf` as follows:

```
filter = [ "a|/dev/disk/by-id/.*|", "r|.*|" ]
types = [ "device-mapper", 1 ]
```

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

If you are using user-friendly names, specify the path as follows so that only the device mapper names are scanned after multipathing is configured:

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

- 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 the following at a terminal console prompt:

```
mkinitrd -f multipath
```

7 Reboot the server to apply the changes.

7.2.4 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/*
```

If you are using user-friendly names, specify the path as follows so that only the device mapper names are scanned after multipathing is configured:

```
DEVICE /dev/disk/by-id/dm-uuid-.*-mpath-.*
```

7.2.5 Using `--noflush` with Multipath Devices

The `--noflush` option 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
```

7.2.6 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_loss_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 7.4.5, “Creating and Configuring the `/etc/multipath.conf` File” (page 79).

7.2.7 Partitioning Multipath Devices

Behavior changes for how multipathed devices are handled might affect your configuration if you are upgrading.

- Section “SUSE Linux Enterprise Server 11” (page 63)
- Section “SUSE Linux Enterprise Server 10” (page 63)
- Section “SUSE Linux Enterprise Server 9” (page 63)

SUSE Linux Enterprise Server 11

In SUSE® Linux Enterprise Server 11, the default multipath setup relies on `udev` to overwrite the existing symbolic links in the `/dev/disk/by-id` directory when multipathing is started. Before you start multipathing, the link points to the SCSI device by using its `scsi-xxx` name. When multipathing is running, the symbolic link points to the device by using its `dm-uuid-xxx` name. This ensures that the symbolic links in the `/dev/disk/by-id` path persistently point to the same device regardless of whether multipath is started or not. The configuration files (such as `lvm.conf` and `md.conf`) do not need to be modified because they automatically point to the correct device.

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 `udev` to fill in the `/dev/disk/by-*` symlinks. The main benefit is that you can call `kpartx` with the new parameters without rebooting 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-MPIO does not automatically detect and reflect these changes. The device must be reinitialized, which usually requires a reboot.

7.2.8 Supported Architectures for Multipath I/O

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

7.2.9 Supported Storage Arrays for Multipathing

The multipathing drivers and tools 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 Arrays That Are Automatically Detected for Multipathing” (page 64)
- Section “Tested Storage Arrays for Multipathing Support” (page 66)
- Section “Storage Arrays that Require Specific Hardware Handlers” (page 66)

Storage Arrays That 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 7.4.5, “Creating and Configuring the `/etc/multipath.conf` File” (page 79).

Testing of the IBM zSeries* device with multipathing has shown that the `dev_loss_tmo` parameter should be set to 90 seconds, and the `fast_io_fail_tmo` parameter should be set to 5 seconds. If you are using zSeries devices, you must manually create and configure the `/etc/multipath.conf` file to specify the values. For information, see Section “Configuring Default Settings for zSeries in `/etc/multipath.conf`” (page 84).

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 7.4.5, “Creating and Configuring the `/etc/multipath.conf` File” (page 79).

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 66).
- 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 66).
- 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 vendor 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 Arrays That Are Automatically Detected for Multipathing” (page 64).

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 66).

7.3 Multipath Management Tools

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

- Section 7.3.1, “Device Mapper Multipath Module” (page 68)
- Section 7.3.2, “Multipath I/O Management Tools” (page 70)
- Section 7.3.3, “Using MDADM for Multipathed Devices” (page 72)
- Section 7.3.4, “The Linux `multipath(8)` Command” (page 73)

7.3.1 Device Mapper Multipath Module

The Device Mapper Multipath (DM-MP) module provides the multipathing capability for Linux. DM-MPIO is the preferred solution for multipathing on SUSE Linux Enterprise Server 11. It is the only multipathing option shipped with the product that is completely supported by Novell® and SUSE.

DM-MPIO 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-MPIO framework is extensible in two ways:

- Using specific hardware handlers. For information, see Section “Storage Arrays that Require Specific Hardware Handlers” (page 66).
- Using load-balancing algorithms that are more sophisticated than the round-robin algorithm

The user-space component of DM-MPIO 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-MPIO 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 you try to access them.

Table 7.1 *Multipath I/O Features of Storage Arrays*

Features of Storage Arrays	Description
Active/passive controllers	<p>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 primary controller fails, the second controller takes over, and it serves all LUNs.</p> <p>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.</p>
Active/active controllers	<p>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.</p>
Load balancing	<p>The Device Mapper Multipath driver automatically load balances traffic across all active paths.</p>

Features of Storage Arrays	Description
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. Multipathing is supported for the <code>/boot</code> device in SUSE Linux Enterprise Server 11 and later.

Device Mapper Multipath detects every path for a multipathed device as a separate SCSI device. The SCSI device names take the form `/dev/sdN`, 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 7.4.5, “Creating and Configuring the `/etc/multipath.conf` File” (page 79).

7.3.2 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 7.2 *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 <code>multipath</code> .
devmap-name	Provides a meaningful device name to <code>udev</code> 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 Specifications > Package 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:

- Enter 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
```

7.3.3 Using MDADM for Multipathed Devices

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 of MDADM and LVM where the configuration files (`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/*
```

If you are using user-friendly names, specify the path as follows so that only the device mapper names are scanned after multipathing is configured:

```
DEVICE /dev/disk/by-id/dm-uuid-.*-mpath-.*
```

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.conf` file, see Section 7.2.3, “Using LVM2 on Multipath Devices” (page 60).

7.3.4 The Linux `multipath(8)` Command

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

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 ]
```

General Examples

```
multipath
```

Configure all multipath devices.

```
multipath devicename
```

Configure 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 suppress a multipath map, and its device-mapped partitions.

```
multipath -d
```

Display potential multipath devices, but do not create any devices and do not update device maps (dry run).

`multipath -v2 -d`

Configure multipath devices and display multipath map information for them. The `-v2` option in `multipath -v2 -d` shows only local disks.

`multipath -v2 devicename`

Configure a specific multipath device and display multipath map information for it. The `-v2` option shows only local disks.

`multipath -v3 -d`

Configure multipath devices and display multipath map information for them. The `-v3` option shows the full path list.

`multipath -v3 devicename`

Configure a specific multipath device and display multipath map information for it. The `-v3` option shows the full path list.

`multipath -ll`

Display the status of all multipath devices.

`multipath -ll devicename`

Display the status of a specified multipath device.

`multipath -F`

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

`multipath -F devicename`

Flush unused multipath device maps for a specified multipath device. This unresolves the multiple paths; it does not delete the device.

`multipath -p [failover | multibus | group_by_serial | group_by_prio | group_by_node_name]`

Set the group policy by specifying one of the group policy options that are described in Table 7.3, “Group Policy Options for the `multipath -p` Command” (page 75):

Table 7.3 *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-controller, or per-multipath option in the <code>/etc/multipath.conf</code> configuration file.
group_by_node_name	One priority group per target node name. Target node names are fetched in the <code>/sys/class/fc_transport/target*/node_name</code> location.

7.4 Configuring the System for Multipathing

- Section 7.4.1, “Preparing SAN Devices for Multipathing” (page 76)
- Section 7.4.2, “Partitioning Multipathed Devices” (page 77)
- Section 7.4.3, “Configuring the Server for Multipathing” (page 77)
- Section 7.4.4, “Adding multipathd to the Boot Sequence” (page 78)
- Section 7.4.5, “Creating and Configuring the `/etc/multipath.conf` File” (page 79)

7.4.1 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 7.2.6, “SAN Timeout Settings When the Root Device Is Multipathed” (page 62).

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.

7.4.2 Partitioning Multipathed Devices

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

- Section “SUSE Linux Enterprise Server 10” (page 77)
- Section “SUSE Linux Enterprise Server 9” (page 77)

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-MPIO automatically recognizes the partitions and indicates them by appending `p1` to `pn` to the device’s ID, such as

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

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

7.4.3 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`.

You need to add the necessary 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 startup, add it here:

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

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

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

7.4.4 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 “Using YaST to Add multipathd” (page 78)
- Section “Using the Command Line to Add multipathd” (page 79)

Using YaST to Add multipathd

- 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.

Using the Command Line to Add multipathd

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

```
insserv multipathd
```

7.4.5 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 80)
- Section “Verifying the Setup in the `etc/multipath.conf` File” (page 80)
- Section “Configuring User-Friendly Names or Alias Names in `/etc/multipath.conf`” (page 81)
- Section “Blacklisting Non-Multipathed Devices in `/etc/multipath.conf`” (page 83)
- Section “Configuring Default Multipath Behavior in `/etc/multipath.conf`” (page 84)
- Section “Configuring Default Settings for zSeries in `/etc/multipath.conf`” (page 84)
- Section “Applying the `/etc/multipath.conf` File Changes” (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 66)), 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 -v3 -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.

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

A multipath device can be identified by either its WWID (Worldwide Identifier) or an alias that you assign for it. The WWID 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 the ID is preferred.

The multipath device names in the `/dev/mapper` directory reference the ID of the LUN and are always consistent because they use the `/var/lib/multipath/bindings` file to track the association. These device names are user-friendly names such as `/dev/disk/by-id/dm-uuid-.*-mpath-.*`.

You can use the `ALIAS` directive in the `/etc/multipath.conf` file to specify your own device names. Alias names override the use of `ID` and `/dev/disk/by-id/dm-uuid-.*-mpath-.*` names.

IMPORTANT

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

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

- 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` directive and its ending bracket.
- 4 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 Optionally specify your own user-friendly names for devices using the `alias` directive in the `multipath` section.

For example:

```
multipath {
    wwid 26353900f02796769
    alias sdd410
}
```

- 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 are 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

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 blacklists 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.

After you do this, 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). The sample files should contain the correct path of `/sbin/scsi_id` in SLES 11 SP1 and later.

Configuring Default Settings for zSeries in /etc/multipath.conf

Testing of the IBM zSeries device with multipathing has shown that the `dev_loss_tmo` parameter should be set to 90 seconds, and the `fast_io_fail_tmo` parameter should be set to 5 seconds. If you are using zSeries devices, modify the `/etc/multipath.conf` file to specify the values as follows:

```
defaults {
    dev_loss_tmo 90
}
```

```
    fast_io_fail_tmo 5
}
```

The `dev_loss_tmo` parameter sets the number of seconds to wait before marking a multipath link as bad. When the path fails, any current I/O on that failed path fails. The default value varies according to the device driver being used. The valid range of values is 0 to 600 seconds. To use the driver's internal timeouts, set the value to zero (0) or to any value greater than 600.

The `fast_io_fail_tmo` parameter sets the length of time to wait before failing I/O when a link problem is detected. I/O that reaches the driver fails. If I/O is in a blocked queue, the I/O does not fail until the `dev_loss_tmo` time elapses and the queue is unblocked.

Applying the `/etc/multipath.conf` File Changes

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.

7.5 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 to start them manually:

- 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
```

7.6 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.

- Section 7.6.1, “Configuring the Path Failover Policies” (page 87)
- Section 7.6.2, “Configuring Failover Priorities” (page 88)
- Section 7.6.3, “Using a Script to Set Path Priorities” (page 96)
- Section 7.6.4, “Configuring ALUA (`mpath_prio_alua`)” (page 97)
- Section 7.6.5, “Reporting Target Path Groups” (page 99)

7.6.1 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 7.4 *Group Policy Options for the `multipath -p` Command*

Policy Option	Description
<code>failover</code>	One path per priority group.
<code>multibus</code>	All paths in one priority group.
<code>group_by_serial</code>	One priority group per detected serial number.
<code>group_by_prio</code>	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 <code>/etc/multipath.conf</code> configuration file.
<code>group_by_node_name</code>	One priority group per target node name. Target node names are fetched in the <code>/sys/class/fc_transport/target*/node_name</code> location.

7.6.2 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 95)
- Section “Configuring for Single Path Failover” (page 96)
- Section “Grouping I/O Paths for Round-Robin Load Balancing” (page 96)

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 the priority is. An external program is used to assign priorities for each path. For a given device, the paths with the same priorities belong to the same priority group.

Table 7.5 *Multipath Attributes*

Multipath Attribute	Description	Values
<code>user_friendly_names</code>	Specifies whether to use IDs or to use the <code>/var/lib/</code>	yes: Autogenerate user-friendly names as aliases for the multipath devices instead of the actual ID.

Multipath Attribute	Description	Values
	multipath/ bindings file to assign a persistent and unique alias to the multipath devices in the form of /dev/mapper/mpathN.	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, hd, md, dm, sr, scd, st, ram, raw, loop.	For an example, see Section “Blacklisting Non-Multipathed Devices in /etc/multipath.conf” (page 83).
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.
failback	<p>Specifies whether to monitor the failed path recovery, and indicates the timing for group failback after failed paths return to service.</p> <p>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,</p>	<p>immediate: When a path recovers, enable the path immediately.</p> <p>n (> 0): When the path recovers, wait <i>n</i> seconds before enabling the path. Specify an integer value greater than 0.</p> <p>manual: (Default) The failed path is not monitored for recovery. The administrator runs the <code>multipath</code> command to update enabled paths and priority groups.</p>

Multipath Attribute	Description	Values
getuid	and changes the active priority group when the priority of the primary path exceeds the secondary group. The default program and arguments to call to obtain a unique path identifier. Should be specified with an absolute path.	<pre data-bbox="755 487 1032 506">/sbin/scsi_id -g -u -s</pre> <p data-bbox="755 535 1166 594">This is the default location and arguments.</p> <p data-bbox="755 633 862 659">Example:</p> <pre data-bbox="755 708 1131 753">getuid "/sbin/scsi_id -g -u -d /dev/%n"</pre>
no_path_retry	Specifies the behaviors to use on path failure.	<p data-bbox="755 808 1185 935">n (> 0): Specifies the number of retries until <code>multipath</code> stops the queuing and fails the path. Specify an integer value greater than 0.</p> <p data-bbox="755 971 1185 1029">fail: Specified immediate failure (no queuing).</p> <p data-bbox="755 1065 1185 1127">queue : Never stop queuing (queue forever until the path comes alive).</p>
path_grouping_policy	Specifies the path grouping policy for a multipath device hosted by a given controller.	<p data-bbox="755 1169 1185 1260">failover: One path is assigned per priority group so that only one path at a time is used.</p> <p data-bbox="755 1295 1185 1421">multibus: (Default) All valid paths are in one priority group. Traffic is load-balanced across all active paths in the group.</p>

Multipath Attribute	Description	Values
path_checker	Determines the state of the path.	<p>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.</p> <p>group_by_serial: Paths are grouped by the SCSI target serial number (controller node WWN).</p> <p>group_by_node_name: One priority group is assigned per target node name. Target node names are fetched in <code>/sys/class/fc_transport/target*/node_name</code>.</p> <p>directio: (Default in <code>multipath-tools</code> 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 <code>/var/log/messages</code>.</p> <p>readsector0: (Default in <code>multipath-tools</code> version 0.4.7 and earlier) Reads the first sector of the device. Logs failure messages in <code>/var/log/messages</code>.</p> <p>tur: Issues a SCSI test unit ready command to the device. This is the preferred setting if the LUN supports it. On failure, the command does not fill up <code>/var/log/messages</code> with messages.</p>

Multipath Attribute	Description	Values
path_selector	Specifies the path-selector algorithm to use for load balancing.	<p>Some SAN vendors provide custom path_checker options:</p> <ul style="list-style-type: none"> • emc_clariion: Queries the EMC Clariion EVPD page 0xC0 to determine the path state. • 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. <p>round-robin 0: (Default) The load-balancing algorithm used to balance traffic across all active paths in a priority group.</p> <p>Beginning in SUSE Linux Enterprise Server 11, the following additional I/O balancing options are available:</p> <p>least-pending: Provides a least-pending-I/O dynamic load balancing policy for bio based device mapper multipath. This load balancing policy considers the number of unserved requests pending on a path and selects the path with least count of pending service requests.</p> <p>This policy is especially useful when the SAN environment has heteroge-</p>

Multipath Attribute	Description	Values
pg_timeout	Specifies path group timeout handling.	<p>neous components. For example, when there is one 8 GB HBA and one 2 GB HBA connected to the same server, the 8 GB HBA could be utilized better with this algorithm.</p> <p>length-load-balancing: A dynamic load balancer that balances the number of in-flight I/O on paths similar to the least-pending option.</p> <p>service-time: A service-time oriented load balancer that balances I/O on paths according to the latency.</p> <p>NONE (internal default)</p>
<p>prio_callout</p> <p>Multipath prio_callouts are located in shared libraries in <code>/lib/libmultipath/lib*</code>. By using shared libraries, the callouts are loaded into memory on daemon startup.</p>	<p>Specifies the program and arguments to use to determine the layout of the multipath map.</p> <p>When queried by the <code>multipath</code> command, the specified <code>mpath_prio_*</code> callout program returns the priority for a given path in relation to the entire multipath layout.</p> <p>When it is used with the <code>path_grouping_policy</code> of <code>group_by_prio</code>, all paths with the same</p>	<p>If no <code>prio_callout</code> attribute is used, all paths are equal. This is the default.</p> <p>/bin/true: Use this value when the <code>group_by_priority</code> is not being used.</p> <p>The <code>prioritizer</code> programs generate path priorities when queried by the <code>multipath</code> command. The program names must begin with <code>mpath_prio_</code> and are named by the device type or balancing method used. Current prioritizer programs include the following:</p> <p>mpath_prio_alua %n: Generates path priorities based on the SCSI-3 ALUA settings.</p>

Multipath Attribute	Description	Values
	<p>priority are grouped into one multipath group. The group with the highest aggregate priority becomes the active group.</p> <p>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.</p> <p>The <code>mpath_prio_*</code> program can also be a custom script created by a vendor or administrator for a specified setup.</p> <p>A <code>%n</code> in the command line expands to the device name in the <code>/dev</code> directory.</p> <p>A <code>%b</code> expands to the device number in <code>major:minor</code> format in the <code>/dev</code> directory.</p> <p>A <code>%d</code> expands to the device ID in the <code>/dev/</code></p>	<p>mpath_prio_balance_units: Generates the same priority for all paths.</p> <p>mpath_prio_emc %n: Generates the path priority for EMC arrays.</p> <p>mpath_prio_hds_modular %b: Generates the path priority for Hitachi HDS Modular storage arrays.</p> <p>mpath_prio_hp_sw %n: Generates the path priority for Compaq/HP controller in active/standby mode.</p> <p>mpath_prio_netapp %n: Generates the path priority for NetApp arrays.</p> <p>mpath_prio_random %n: Generates a random priority for each path.</p> <p>mpath_prio_rdac %n: Generates the path priority for LSI/Engenio RDAC controller.</p> <p>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.</p> <p>mpath_prio_spec.sh %n: Provides the path of a user-created script that generates the priorities for 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 in-</p>

Multipath Attribute	Description	Values
<code>rr_min_io</code>	<p><code>disk/by-id</code> directory.</p> <p>If devices are hot-pluggable, use the <code>%d</code> flag instead of <code>%n</code>. This addresses the short time that elapses between the time when devices are available and when <code>udev</code> creates the device nodes.</p> <p>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 <code>path_selector</code> setting.</p>	<p>stead.) 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 7.6.3, "Using a Script to Set Path Priorities" (page 96).</p> <p>n (>0): Specify an integer value greater than 0.</p> <p>1000: Default.</p>
<code>rr_weight</code>	<p>Specifies the weighting method to use for paths.</p>	<p>uniform: Default. All paths have the same round-robin weights.</p> <p>priorities: Each path's weight is determined by the path's priority times the <code>rr_min_io</code> setting.</p>

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.

7.6.3 Using a Script to Set Path Priorities

You can create a script that interacts with Device Mapper Multipath (DM-MPIO) 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:

```
0x10000000c95eb4 0x200200a0b8122c6e 2:0:0:0 sdb  
3600a0b8000122c6d0000000453174fc 50
```

```
0x10000000c95eb4 0x200200a0b8122c6e 2:0:0:1 sdc  
3600a0b8000fd632000000045317563 2
```

```
0x10000000c95eb4 0x200200a0b8122c6e 2:0:0:2 sdd  
3600a0b8000122c6d0000000345317524 50
```



```
0x10000000c95eb4 0x200200a0b8122c6e 2:0:0:3 sde  
3600a0b8000fd6320000000245317593 2
```

```
0x10000000c95eb4 0x200300a0b8122c6e 2:0:1:0 sdi  
3600a0b8000122c6d00000000453174fc 5
```

```
0x10000000c95eb4 0x200300a0b8122c6e 2:0:1:1 sdj  
3600a0b8000fd6320000000045317563 51
```

```
0x10000000c95eb4 0x200300a0b8122c6e 2:0:1:2 sdk  
3600a0b8000122c6d0000000345317524 5
```

```
0x10000000c95eb4 0x200300a0b8122c6e 2:0:1:3 sdl  
3600a0b8000fd6320000000245317593 51
```

To continue the example mentioned in Table 7.5, “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.

7.6.4 Configuring ALUA (`mpath_prio_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-MPIO to group SCSI devices with the same priority together. This path priority tool is based on ALUA (Asynchronous Logical Unit Access).

- Section “Syntax” (page 98)
- Section “Prerequisite” (page 98)
- Section “Options” (page 98)
- Section “Return Values” (page 99)

Syntax

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

Prerequisite

SCSI devices.

Options

`-d directory`

Specifies the Linux directory path where the listed device node names can be found. The default directory is `/dev`. When you use this option, 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 [device...]`

Specifies the SCSI device (or multiple devices) that 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 by 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 7.6, “ALUA Priorities for Device Mapper Multipath” (page 99) shows the priority values returned by the `mpath_prio_alua` command.

Table 7.6 *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.
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 ($6 \times 10 = 60$) and the optimized path group has a single path ($1 \times 50 = 50$), 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`.

7.6.5 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)`.

7.7 Tuning the Failover for Specific Host Bus Adapters

When using multipath I/O, you want any host bus adapter (HBA) failure or cable failures to be reported faster than when multipathing is not in use. Configure time-out settings for your HBA to disable failover at the HBA level to allow the failure to propagate up to the multipath I/O layer as fast as possible where the I/O can be redirected to another, healthy path.

To disable the HBA handling of failover, modify the driver's options in the `/etc/modprobe.conf.local` file. Refer to the HBA vendor's documentation for information about how to disable failover settings for your driver.

For example, for the QLogic `qla2xxx` family of host bus adapters, the following setting is recommended:

```
options qla2xxx qlport_down_retry=1
```

7.8 Configuring Multipath I/O for the Root Device

IMPORTANT

In the SUSE Linux Enterprise Server 10 SP1 initial release and earlier, 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.

DM-MPIO is now available and supported for `/boot` and `/root` in SUSE Linux Enterprise Server 11. In addition, the YaST partitioner in the YaST2 installer supports enabling multipath during the install.

-
- Section 7.8.1, “Enabling Multipath I/O to Install SLES on a Multipath Storage LUN” (page 101)

- Section 7.8.2, “Enabling Multipath I/O to Install SLES on an Active/Passive Multipath Storage LUN” (page 101)
- Section 7.8.3, “Enabling Multipath I/O for an Existing Root Device” (page 104)
- Section 7.8.4, “Disabling Multipath I/O on the Root Device” (page 105)

7.8.1 Enabling Multipath I/O to Install SLES on a Multipath Storage LUN

The multipathd daemon is not automatically active during the system installation. You can start it by using the *Configure Multipath* option in the YaST partitioner.

- 1 During the install on the YaST2 Installation Settings page, click on *Partitioning* to open the YaST partitioner.
- 2 Select *Custom Partitioning (for experts)*.
- 3 Select the *Hard Disks* main icon, click the *Configure* button, then select *Configure Multipath*.
- 4 Start multipath.

YaST2 starts to rescan the disks and shows available multipath devices (such as `/dev/mapper/3600a0b80000f4593000012ae4ab0ae65`). This is the device that should be used for all further processing.

- 5 Click *Next* to continue with the installation.

7.8.2 Enabling Multipath I/O to Install SLES on an Active/Passive Multipath Storage LUN

The multipathd daemon is not automatically active during the system installation. You can start it by using the *Configure Multipath* option in the YaST partitioner.

- 1 During the install on the YaST2 Installation Settings page, click on *Partitioning* to open the YaST partitioner.
- 2 Select *Custom Partitioning (for experts)*.
- 3 Select the *Hard Disks* main icon, click the *Configure* button, then select *Configure Multipath*.
- 4 Start multipath.

YaST2 starts to rescan the disks and shows available multipath devices (such as `/dev/mapper/3600a0b80000f4593000012ae4ab0ae65`). This is the device that should be used for all further processing. Write down the device path and UUID; you need it later.

- 5 Click *Next* to continue with the installation.
- 6 After all settings are done and the installation finished, YaST2 starts to write the boot loader information, and displays a countdown to restart the system. Stop the counter by clicking the *Stop* button and press CTRL+ALT+F5 to access a console.
- 7 Use the console to determine if a passive path was entered in the `/boot/grub/device.map` file for the `hd0` entry.

This is necessary because the installation does not distinguish between active and passive paths.

- 7a Mount the root device to `/mnt` by entering

```
mount /dev/mapper/UUID_part2 /mnt
```

For example, enter

```
mount /dev/mapper/3600a0b80000f4593000012ae4ab0ae65_part2 /mnt
```

- 7b Mount the boot device to `/mnt/boot` by entering

```
mount /dev/mapper/UUID_part1 /mnt/boot
```

For example, enter

```
mount /dev/mapper/3600a0b80000f4593000012ae4ab0ae65_part1 /mnt/boot
```

7c Open `/mnt/boot/grub/device.map` file by entering

```
less /mnt/boot/grub/device.map
```

7d In the `/mnt/boot/grub/device.map` file, determine if the `hd0` entry points to a passive path, then do one of the following:

- **Active path:** No action is needed; skip Step 8 (page 103) and continue with Step 9 (page 104).
- **Passive path:** The configuration must be changed and the boot loader must be reinstalled. Continue with Step 8 (page 103).

8 If the `hd0` entry points to a passive path, change the configuration and reinstall the boot loader:

8a At the console, enter the following commands at the console prompt:

```
mount -o bind /dev /mnt/dev
mount -o bind /sys /mnt/sys
mount -o bind /proc /mnt/proc
chroot
```

8b At the console, run `multipath -ll`, then check the output to find the active path.

Passive paths are flagged as `ghost`.

8c In the `/mnt/boot/grub/device.map` file, change the `hd0` entry to an active path, save the changes, and close the file.

- 8d** In case the selection was to boot from MBR, `/etc/grub.conf` should look like the following:

```
setup --stage2=/boot/grub/stage2 (hd0) (hd0,0)
quit
```

- 8e** Reinstall the boot loader by entering

```
grub < /etc/grub.conf
```

- 8f** Enter the following commands:

```
exit
umount /mnt/*
umount /mnt
```

- 9** Return to the YaST graphical environment by pressing CTRL+ALT+F7.

- 10** Click *OK* to continue with the installation reboot.

7.8.3 Enabling Multipath I/O for an 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.

7.8.4 Disabling Multipath I/O on the Root Device

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

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

7.9 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-MPIO 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 multipath
```

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
```

7.10 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
<code>-l</code>	Activates scanning for LUNs 0-7. [Default: 0]
<code>-L NUM</code>	Activates scanning for LUNs 0 to NUM. [Default: 0]

Option	Description
-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.
<hr/>	
WARNING	
Use with caution, this option is dangerous.	
<hr/>	
--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.

Option	Description
	<code>--channels=A[-B] [,C[-D]]</code>
<code>--ids=LIST</code>	Scans only target IDs in LIST, where LIST is a comma-separated list of single values and ranges. No spaces are allowed.
	<code>--ids=A[-B] [,C[-D]]</code>
<code>--luns=LIST</code>	Scans only LUNs in LIST, where LIST is a comma-separated list of single values and ranges. No spaces are allowed.
	<code>--luns=A[-B] [,C[-D]]</code>

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 110) through Step 3 (page 110) 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-MPIO configuration. At a terminal console prompt, enter

```
multipath
```

You can now configure the new device for multipathing.

7.11 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 7.10, “Scanning for New Devices without Rebooting” (page 108).

- 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-MPIO 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 112) 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
```

7.12 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 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 device

7.13 Managing I/O in Error Situations

You might need to configure multipathing to queue I/O if all paths fail concurrently by enabling `queue_if_no_path`. Otherwise, I/O fails immediately if all paths are gone. In certain scenarios, where the driver, the HBA, or the fabric experience spurious errors, DM-MPIO should be configured to queue all I/O where those errors lead to a loss of all paths, and never propagate errors upward.

When you use multipathed devices in a cluster, you might choose to disable `queue_if_no_path`. This automatically fails the path instead of queuing the I/O, and escalates the I/O error to cause a failover of the cluster resources.

Because enabling `queue_if_no_path` 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.

7.14 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
```

7.15 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:

- *How to Setup/Use Multipathing on SLES* [http://support.novell.com/techcenter/sdb/en/2005/04/sles_multipathing.html]
- *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]

- *Dynamically Adding Storage for Use with Multipath I/O (Technical Information Document 3000817)* [https://secure-support.novell.com/KanisaPlatform/Publishing/911/3000817_f.SAL_Public.html]
- *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]

7.16 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 8, *Software RAID Configuration* (page 119)
- Chapter 10, *Managing Software RAIDs 6 and 10 with mdadm* (page 135)

Software RAID Configuration

The purpose of RAID (redundant array of independent disks) is to combine several hard disk partitions into one large virtual hard disk to optimize performance, data security, or both. Most RAID controllers use the SCSI protocol because it can address a larger number of hard disks in a more effective way than the IDE protocol and is more suitable for parallel processing of commands. There are some RAID controllers that support IDE or SATA hard disks. Software RAID provides the advantages of RAID systems without the additional cost of hardware RAID controllers. However, this requires some CPU time and has memory requirements that make it unsuitable for real high performance computers.

IMPORTANT

Software RAID is not supported underneath clustered file systems such as OCFS2, because RAID does not support concurrent activation. If you want RAID for OCFS2, you need the RAID to be handled by the storage subsystem.

SUSE® Linux Enterprise offers the option of combining several hard disks into one soft RAID system. RAID implies several strategies for combining several hard disks in a RAID system, each with different goals, advantages, and characteristics. These variations are commonly known as *RAID levels*.

- Section 8.1, “Understanding RAID Levels” (page 120)
- Section 8.2, “Soft RAID Configuration with YaST” (page 122)
- Section 8.3, “Troubleshooting” (page 124)

- Section 8.4, “For More Information” (page 125)

8.1 Understanding RAID Levels

This section describes common RAID levels 0, 1, 2, 3, 4, 5, and nested RAID levels.

- Section 8.1.1, “RAID 0” (page 120)
- Section 8.1.2, “RAID 1” (page 120)
- Section 8.1.3, “RAID 2 and RAID 3” (page 121)
- Section 8.1.4, “RAID 4” (page 121)
- Section 8.1.5, “RAID 5” (page 121)
- Section 8.1.6, “Nested RAID Levels” (page 121)

8.1.1 RAID 0

This level improves the performance of your data access by spreading out blocks of each file across multiple disk drives. Actually, this is not really a RAID, because it does not provide data backup, but the name *RAID 0* for this type of system has become the norm. With RAID 0, two or more hard disks are pooled together. The performance is very good, but the RAID system is destroyed and your data lost if even one hard disk fails.

8.1.2 RAID 1

This level provides adequate security for your data, because the data is copied to another hard disk 1:1. This is known as *hard disk mirroring*. If a disk is destroyed, a copy of its contents is available on another mirrored disk. All disks except one could be damaged without endangering your data. However, if damage is not detected, damaged data might be mirrored to the correct disk and the data is corrupted that way. The writing performance suffers a little in the copying process compared to when using single disk access (10 to 20 % slower), but read access is significantly faster in comparison to any one of the normal physical hard disks, because the data is duplicated so can be scanned

in parallel. RAID 1 generally provides nearly twice the read transaction rate of single disks and almost the same write transaction rate as single disks.

8.1.3 RAID 2 and RAID 3

These are not typical RAID implementations. Level 2 stripes data at the bit level rather than the block level. Level 3 provides byte-level striping with a dedicated parity disk and cannot service simultaneous multiple requests. Both levels are rarely used.

8.1.4 RAID 4

Level 4 provides block-level striping just like Level 0 combined with a dedicated parity disk. If a data disk fails, the parity data is used to create a replacement disk. However, the parity disk might create a bottleneck for write access. Nevertheless, Level 4 is sometimes used.

8.1.5 RAID 5

RAID 5 is an optimized compromise between Level 0 and Level 1 in terms of performance and redundancy. The hard disk space equals the number of disks used minus one. The data is distributed over the hard disks as with RAID 0. *Parity blocks*, created on one of the partitions, are there for security reasons. They are linked to each other with XOR, enabling the contents to be reconstructed by the corresponding parity block in case of system failure. With RAID 5, no more than one hard disk can fail at the same time. If one hard disk fails, it must be replaced as soon as possible to avoid the risk of losing data.

8.1.6 Nested RAID Levels

Several other RAID levels have been developed, such as RAIDn, RAID 10, RAID 0+1, RAID 30, and RAID 50. Some of them being proprietary implementations created by hardware vendors. These levels are not very widespread, and are not explained here.

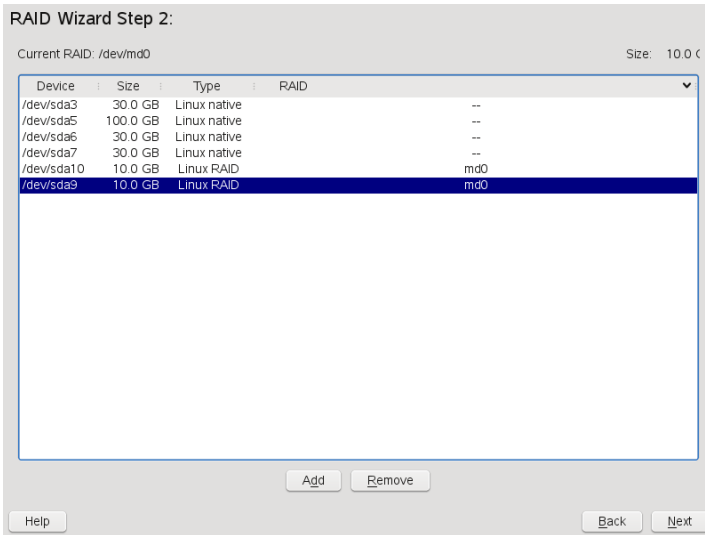
8.2 Soft RAID Configuration with YaST

The YaST soft RAID configuration can be reached from the YaST Expert Partitioner. This partitioning tool enables you to edit and delete existing partitions and create new ones that should be used with soft RAID.

You can create RAID partitions by first clicking *Create > Do not format* then selecting *0xFD Linux RAID* as the partition identifier. For RAID 0 and RAID 1, at least two partitions are needed—for RAID 1, usually exactly two and no more. If RAID 5 is used, at least three partitions are required. It is recommended to use only partitions of the same size because each segment can contribute only the same amount of space as the smallest sized partition. The RAID partitions should be stored on different hard disks to decrease the risk of losing data if one is defective (RAID 1 and 5) and to optimize the performance of RAID 0. After creating all the partitions to use with RAID, click *RAID > Create RAID* to start the RAID configuration.

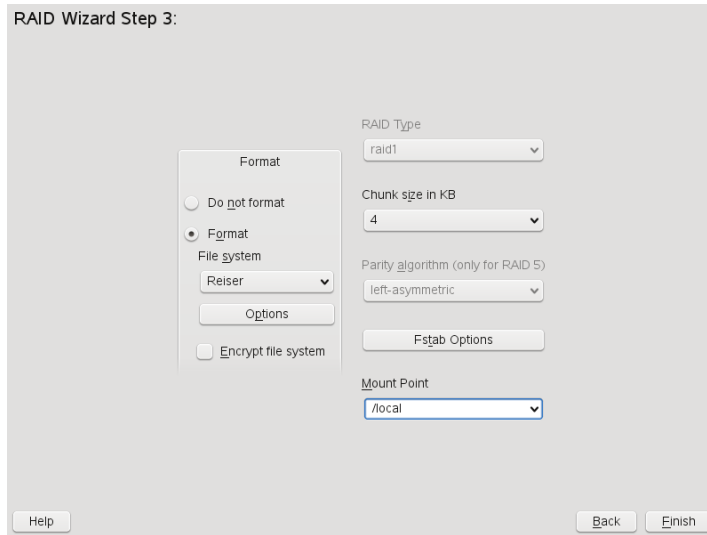
In the next dialog, choose among RAID levels 0, 1, and 5, then click *Next*. The following dialog (see Figure 8.1, “RAID Partitions” (page 123)) lists all partitions with either the *Linux RAID* or *Linux native* type. No swap or DOS partitions are shown. If a partition is already assigned to a RAID volume, the name of the RAID device (for example, `/dev/md0`) is shown in the list. Unassigned partitions are indicated with “--”.

Figure 8.1 RAID Partitions



To add a previously unassigned partition to the selected RAID volume, first select the partition then click *Add*. At this point, the name of the RAID device is displayed next to the selected partition. Assign all partitions reserved for RAID. Otherwise, the space on the partition remains unused. After assigning all partitions, click *Next* to proceed to the settings dialog where you can fine-tune the performance (see Figure 8.2, “File System Settings” (page 124)).

Figure 8.2 *File System Settings*



As with conventional partitioning, set the file system to use as well as encryption and the mount point for the RAID volume. After completing the configuration with *Finish*, see the `/dev/md0` device and others indicated with *RAID* in the Expert Partitioner.

8.3 Troubleshooting

Check the `/proc/mdstats` file to find out whether a RAID partition has been damaged. In the event of a system failure, shut down your Linux system and replace the defective hard disk with a new one partitioned the same way. Then restart your system and enter the command `mdadm /dev/mdX --add /dev/sdX`. Replace X with your particular device identifiers. This integrates the hard disk automatically into the RAID system and fully reconstructs it.

Although you can access all data during the rebuild, you might encounter some performance issues until the RAID has been fully rebuilt.

8.4 For More Information

Configuration instructions and more details for soft RAID can be found in the HOWTOs at:

- *The Software RAID HOWTO* [<http://en.tldp.org/HOWTO/Software-RAID-HOWTO.html>]
- *The Software RAID HOWTO* in the `/usr/share/doc/packages/mdadm/Software-RAID.HOWTO.html` file

Linux RAID mailing lists are also available, such as linux-raid [<http://marc.theaimsgroup.com/?l=linux-raid>].

Configuring Software RAID for the Root Partition

In SUSE® Linux Enterprise Server 11, the Device Mapper RAID tool has been integrated into the YaST Partitioner. You can use the partitioner at install time to create a software RAID for the system device that contains your root (/) partition.

- Section 9.1, “Prerequisites for the Software RAID” (page 127)
- Section 9.2, “Enabling iSCSI Initiator Support at Install Time” (page 128)
- Section 9.3, “Enabling Multipath I/O Support at Install Time” (page 129)
- Section 9.4, “Creating a Software RAID Device for the Root (/) Partition” (page 129)

9.1 Prerequisites for the Software RAID

Make sure your configuration meets the following requirements:

- You need two or more hard drives, depending on the type of software RAID you plan to create.
- **RAID 0 (Striping):** RAID 0 requires two or more devices. RAID 0 offers no fault tolerance benefits, and it is not recommended for the system device.
- **RAID 1 (Mirroring):** RAID 1 requires two devices.

- **RAID 5 (Redundant Striping):** RAID 5 requires three or more devices.
- The hard drives should be similarly sized. The RAID assumes the size of the smallest drive.
- The block storage devices can be any combination of local (in or directly attached to the machine), Fibre Channel storage subsystems, or iSCSI storage subsystems.
- If you are using hardware RAID devices, do not attempt to run software RAIDs on top of it.
- If you are using iSCSI target devices, enable the iSCSI initiator support before you create the RAID device.
- If your storage subsystem provides multiple I/O paths between the server and its directly attached local devices, Fibre Channel devices, or iSCSI devices that you want to use in the software RAID, you must enable the multipath support before you create the RAID device.

9.2 Enabling iSCSI Initiator Support at Install Time

If there are iSCSI target devices that you want to use for the root (/) partition, you must enable the iSCSI Initiator software to make those devices available to you before you create the software RAID device.

- 1 Proceed with the YaST install of SUSE Linux Enterprise 11 until you reach the Installation Settings page.
- 2 Click *Partitioning* to open the Preparing Hard Disk page, click *Custom Partitioning (for experts)*, then click *Next*.
- 3 On the Expert Partitioner page, expand *Hard Disks* in the *System View* panel to view the default proposal.
- 4 On the *Hard Disks* page, select *ConfigureConfigure iSCSI*, then click *Continue* when prompted to continue with initializing the iSCSI initiator configuration.

9.3 Enabling Multipath I/O Support at Install Time

If there are multiple I/O paths to the devices you want to use to create a software RAID device for the root (/) partition, you must enable multipath support before you create the software RAID device.

- 1 Proceed with the YaST install of SUSE Linux Enterprise 11 until you reach the Installation Settings page.
- 2 Click *Partitioning* to open the Preparing Hard Disk page, click *Custom Partitioning (for experts)*, then click *Next*.
- 3 On the Expert Partitioner page, expand *Hard Disks* in the *System View* panel to view the default proposal.
- 4 On the *Hard Disks* page, select *Configure Configure Multipath*, then click *Yes* when prompted to activate multipath.

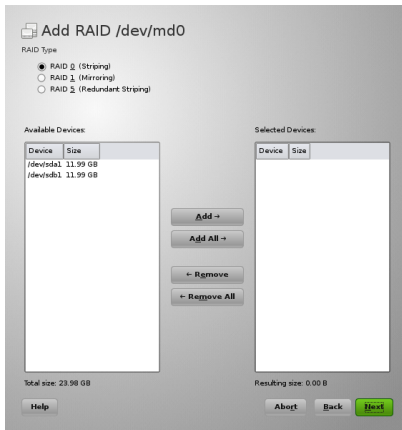
This re-scans the devices and resolves the multiple paths so that each device is listed only once in the list of hard disks.

9.4 Creating a Software RAID Device for the Root (/) Partition

- 1 Proceed with the YaST install of SUSE Linux Enterprise 11 until you reach the Installation Settings page.
- 2 Click *Partitioning* to open the Preparing Hard Disk page, click *Custom Partitioning (for experts)*, then click *Next*.
- 3 On the Expert Partitioner page, expand *Hard Disks* in the *System View* panel to view the default proposal, select the proposed partitions, then click *Delete*.
- 4 Create a swap partition.

- 4a** On the Expert Partitioner page under *Hard Disks*, select the device you want to use for the swap partition, then click *Add* on the *Hard Disk Partitions* tab.
 - 4b** Under *New Partition Type*, select *Primary Partition*, then click *Next*.
 - 4c** Under *New Partition Size*, specify the size to use, then click *Next*.
 - 4d** Under *Format Options*, select *Format partition*, then select *Swap* from the drop-down list.
 - 4e** Under *Mount Options*, select *Mount partition*, then select *swap* from the drop-down list.
 - 4f** Click *Finish*.
- 5** Set up the *0xFD Linux RAID* format for each of the devices you want to use for the software RAID.
 - 5a** On the Expert Partitioner page under *Hard Disks*, select the device you want to use in the RAID, then click *Add* on the *Hard Disk Partitions* tab.
 - 5b** Under *New Partition Type*, select *Primary Partition*, then click *Next*.
 - 5c** Under *New Partition Size*, specify to use the maximum size, then click *Next*.
 - 5d** Under *Format Options*, select *Do not format partition*, then select *0xFD Linux RAID* from the drop-down list.
 - 5e** Under *Mount Options*, select *Do not mount partition*.
 - 5f** Click *Finish*.
 - 5g** Repeat Step 5a (page 130) to Step 5f (page 130) for each device that you plan to use in the software RAID
- 6** Create the RAID device.
 - 6a** In the *System View* panel, select *RAID*, then click *Add RAID* on the RAID page.

The devices that you prepared in Step 5 (page 130) are listed in *Available Devices*.



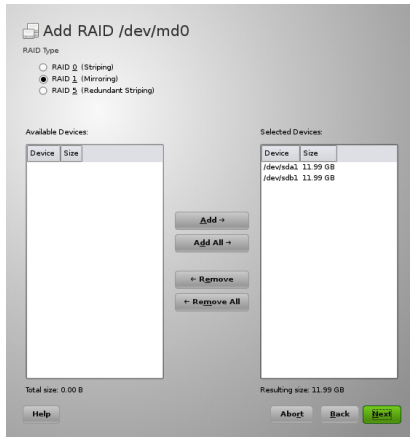
- 6b** Under *RAID Type*, select *RAID 0 (Striping)*, *RAID 1 (Mirroring)*, or *RAID 5 (Redundant Striping)*.

For example, select RAID 1 (Mirroring).

- 6c** In the *Available Devices* panel, select the devices you want to use for the RAID, then click *Add* to move the devices to the *Selected Devices* panel.

Specify two or more devices for a RAID 1, two devices for a RAID 0, or at least three devices for a RAID 5.

To continue the example, two devices are selected for RAID 1.



6d Click *Next*.

6e Under *RAID Options*, select the chunk size from the drop-down list.

The default chunk size for a RAID 1 (Mirroring) is 4 KB.

The default chunk size for a RAID 0 (Striping) is 32 KB.

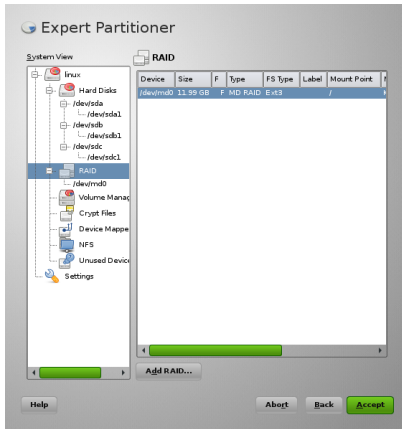
Available chunk sizes are 4 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB, 2 MB, or 4 MB.

6f Under *Formatting Options*, select *Format partition*, then select the file system type (such as Ext3) from the *File system* drop-down list.

6g Under *Mounting Options*, select *Mount partition*, then select */* from the *Mount Point* drop-down list.

6h Click *Finish*.

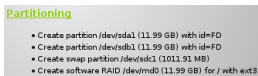
The software RAID device is managed by Device Mapper, and creates a device under the `/dev/md0` path.



7 On the Expert Partitioner page, click *Accept*.

The new proposal appears under *Partitioning* on the Installation Settings page.

For example, the setup for the



8 Continue with the install.

Whenever you reboot your server, Device Mapper is started at boot time so that the software RAID is automatically recognized, and the operating system on the root (/) partition can be started.

Managing Software RAID 6 and 10 with mdadm

10

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 RAID 0, 1, 4, and 5. The `mdadm` tool provides the functionality of legacy programs `mdtools` and `raidtools`.

- Section 10.1, “Creating a RAID 6” (page 135)
- Section 10.2, “Creating Nested RAID 10 Devices with `mdadm`” (page 137)
- Section 10.3, “Creating a Complex RAID 10 with `mdadm`” (page 142)
- Section 10.4, “Creating a Degraded RAID Array” (page 147)

10.1 Creating a RAID 6

- Section 10.1.1, “Understanding RAID 6” (page 135)
- Section 10.1.2, “Creating a RAID 6” (page 136)

10.1.1 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 two of

the hard disk drives fail during the data recovery process, the system continues to be operational, with no data loss.

RAID 6 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 10.1 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

10.1.2 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
```

10.2 Creating Nested RAID 10 Devices with mdadm

- Section 10.2.1, “Understanding Nested RAID Devices” (page 137)
- Section 10.2.2, “Creating Nested RAID 10 (1+0) with mdadm” (page 139)
- Section 10.2.3, “Creating Nested RAID 10 (0+1) with mdadm” (page 141)

10.2.1 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 10.2 *Nested RAID Levels*

RAID Level	Description	Performance and Fault Tolerance
10 (1+0)	RAID 0 (stripe) built with RAID 1 (mirror) arrays	<p>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.</p> <p>You can optionally configure a spare for each underlying mirrored array, or configure a spare to serve a spare group that serves all mirrors.</p>
10 (0+1)	RAID 1 (mirror) built with RAID 0 (stripe) arrays	<p>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.</p> <p>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</p>

RAID Level	Description	Performance and Fault Tolerance
		<p>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.</p> <p>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.</p>

10.2.2 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 7, *Managing Multipath I/O for Devices* (page 55).

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 10.3 Scenario for Creating a RAID 10 (1+0) by Nesting

Raw Devices	RAID 1 (mirror)	RAID 1+0 (striped mirrors)
/dev/sdb1 /dev/sdc1	/dev/md0	/dev/md2
/dev/sdd1	/dev/md1	

Raw Devices	RAID 1 (mirror)	RAID 1+0 (striped mirrors)
<hr/>		
/dev/sde1		
<hr/>		

- 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 140):

```
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`.

10.2.3 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 7, *Managing Multipath I/O for Devices* (page 55).

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, then 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 10.4 Scenario for Creating a RAID 10 (0+1) by Nesting

Raw Devices	RAID 0 (stripe)	RAID 0+1 (mirrored stripes)
/dev/sdb1 /dev/sdc1	/dev/md0	/dev/md2
/dev/sdd1 /dev/sde1	/dev/md1	

- 1 Open a terminal console, then log in as the root user or equivalent.
- 2 Create two 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 141):

```
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`.

10.3 Creating a Complex RAID 10 with mdadm

- Section 10.3.1, “Understanding the mdadm RAID10” (page 143)
- Section 10.3.2, “Creating a RAID 10 with mdadm” (page 146)

10.3.1 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 “Comparing the Complex RAID10 and Nested RAID 10 (1+0)” (page 143)
- Section “Number of Replicas in the mdadm RAID10” (page 144)
- Section “Number of Devices in the mdadm RAID10” (page 144)
- Section “Near Layout” (page 145)
- Section “Far Layout” (page 145)

Comparing the Complex RAID10 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 10.5 *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 an mdadm RAID10 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 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	sdcl	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	sdcl	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 starts 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.

The speed of the `raid10` for writing is similar to other mirrored RAID types, like `raid1` and `raid10` using near layout, as the elevator of the file system schedules the writes in

a more optimal way than raw writing. Using raid10 in the far layout well-suited for mirrored writing applications.

Far layout with an even number of disks and two replicas:

```
sda1 sdb1 sdc1 sde1
0 1 2 3
3 5 6 7
. . .
3 1 2 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
```

10.3.2 Creating a RAID 10 with mdadm

The RAID10 option for mdadm creates a RAID 10 device without nesting. For information about RAID10-, see Section 10.3, “Creating a Complex RAID 10 with mdadm” (page 142).

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 10.6 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`.

10.4 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 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

11

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 section, make sure that you have a valid backup of all of the data.

- Section 11.1, “Understanding the Resizing Process” (page 151)
- Section 11.2, “Increasing the Size of a Software RAID” (page 153)
- Section 11.3, “Decreasing the Size of a Software RAID” (page 159)

11.1 Understanding the Resizing Process

Resizing an existing software RAID device involves increasing or decreasing the space contributed by each component partition.

- Section 11.1.1, “Guidelines for Resizing a Software RAID” (page 152)
- Section 11.1.2, “Overview of Tasks” (page 152)

11.1.1 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 11, file system resizing utilities are available for file systems Ext2, Ext3, and ReiserFS. The utilities support increasing and decreasing the size as follows:

Table 11.1 *File System Support for Resizing*

File System	Utility	Increase Size	Decrease Size
Ext2 or Ext3	resize2fs	Yes, offline only	Yes, offline only
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.

11.1.2 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 11.2 *Tasks Involved in Resizing a RAID*

Tasks	Description	Order If In- creas- ing Size	Order If De- creas- ing 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, and ReiserFS.	3	1

11.2 Increasing the Size of a Software RAID

Before you begin, review the guidelines in Section 11.1, “Understanding the Resizing Process” (page 151).

- Section 11.2.1, “Increasing the Size of Component Partitions” (page 154)
- Section 11.2.2, “Increasing the Size of the RAID Array” (page 156)
- Section 11.2.3, “Increasing the Size of the File System” (page 157)

11.2.1 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 11.3 *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 155) by doing one of the following:

- Increase the size of the partition, using a disk partitioner such as `fdisk(8)`, `cgdisk(8)`, or `parted(8)`. This option 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 154) through Step 5 (page 155) 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 11.2.2, “Increasing the Size of the RAID Array” (page 156).

11.2.2 Increasing the Size of the RAID Array

After you have resized each of the component partitions in the RAID (see Section 11.2.1, “Increasing the Size of Component Partitions” (page 154)), 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 11.2.3, “Increasing the Size of the File System” (page 157).

- If your array was not resized as you expected, you must reboot, then try this procedure again.

11.2.3 Increasing the Size of the File System

After you increase the size of the array (see Section 11.2.2, “Increasing the Size of the RAID Array” (page 156)), 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 `resize2fs` command.

- 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 can 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.

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 increases 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.

11.3 Decreasing the Size of a Software RAID

Before you begin, review the guidelines in Section 11.1, “Understanding the Resizing Process” (page 151).

- Section 11.3.1, “Decreasing the Size of the File System” (page 160)
- Section 11.3.2, “Decreasing the Size of Component Partitions” (page 162)
- Section 11.3.3, “Decreasing the Size of the RAID Array” (page 164)

11.3.1 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 decreasing 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.

ReiserFS

ReiserFS file systems can be decreased in size 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 decrease in size 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.

11.3.2 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 11.4 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 155) 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 154) through Step 5 (page 155) 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 11.3.3, “Decreasing the Size of the RAID Array” (page 164).

11.3.3 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. Use the `--grow` option to force it to read the change in available disk size. 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:
 - Decrease the size of the array to the maximum available size by entering

```
mdadm --grow /dev/md0 -z max
```

- Decrease 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.

iSNS for Linux

Storage area networks (SANs) can contain many disk drives that are dispersed across complex networks. This can make device discovery and device ownership difficult. iSCSI initiators must be able to identify storage resources in the SAN and determine whether they have access to them.

Internet Storage Name Service (iSNS) is a standards-based service that is available beginning with SUSE® Linux Enterprise Server (SLES) 10 Support Pack 2. iSNS facilitates the automated discovery, management, and configuration of iSCSI devices on a TCP/IP network. iSNS provides intelligent storage discovery and management services comparable to those found in Fibre Channel networks.

IMPORTANT

iSNS should be used only in secure internal networks.

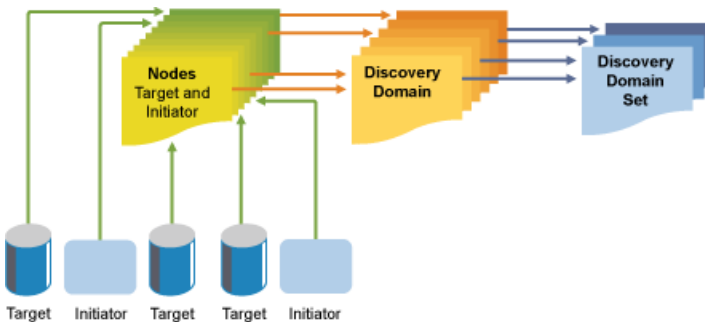
- Section 12.1, “How iSNS Works” (page 168)
- Section 12.2, “Installing iSNS Server for Linux” (page 169)
- Section 12.3, “Configuring iSNS Discovery Domains” (page 171)
- Section 12.4, “Starting iSNS” (page 177)
- Section 12.5, “Stopping iSNS” (page 178)
- Section 12.6, “For More Information” (page 178)

12.1 How iSNS Works

For an iSCSI initiator to discover iSCSI targets, it needs to identify which devices in the network are storage resources and what IP addresses it needs to access them. A query to an iSNS server returns a list of iSCSI targets and the IP addresses that the initiator has permission to access.

Using iSNS, you create iSNS discovery domains and discovery domain sets. You then group or organize iSCSI targets and initiators into discovery domains and group the discovery domains into discovery domain sets. By dividing storage nodes into domains, you can limit the discovery process of each host to the most appropriate subset of targets registered with iSNS, which allows the storage network to scale by reducing the number of unnecessary discoveries and by limiting the amount of time each host spends establishing discovery relationships. This lets you control and simplify the number of targets and initiators that must be discovered.

Figure 12.1 *iSNS Discovery Domains and Discovery Domain Sets*



Both iSCSI targets and iSCSI initiators use iSNS clients to initiate transactions with iSNS servers by using the iSNS protocol. They then register device attribute information in a common discovery domain, download information about other registered clients, and receive asynchronous notification of events that occur in their discovery domain.

iSNS servers respond to iSNS protocol queries and requests made by iSNS clients using the iSNS protocol. iSNS servers initiate iSNS protocol state change notifications and store properly authenticated information submitted by a registration request in an iSNS database.

Some of the benefits provided by iSNS for Linux include:

- Provides an information facility for registration, discovery, and management of networked storage assets.
- Integrates with the DNS infrastructure.
- Consolidates registration, discovery, and management of iSCSI storage.
- Simplifies storage management implementations.
- Improves scalability compared to other discovery methods.

An example of the benefits iSNS provides can be better understood through the following scenario:

Suppose you have a company that has 100 iSCSI initiators and 100 iSCSI targets. Depending on your configuration, all iSCSI initiators could potentially try to discover and connect to any of the 100 iSCSI targets. This could create discovery and connection difficulties. By grouping initiators and targets into discovery domains, you can prevent iSCSI initiators in one department from discovering the iSCSI targets in another department. The result is that the iSCSI initiators in a specific department only discover those iSCSI targets that are part of the department's discovery domain.

12.2 Installing iSNS Server for Linux

iSNS Server for Linux is included with SLES 10 SP2 and later, but is not installed or configured by default. You must install the iSNS package modules (`isns` and `yast2-isns` modules) and configure the iSNS service.

NOTE

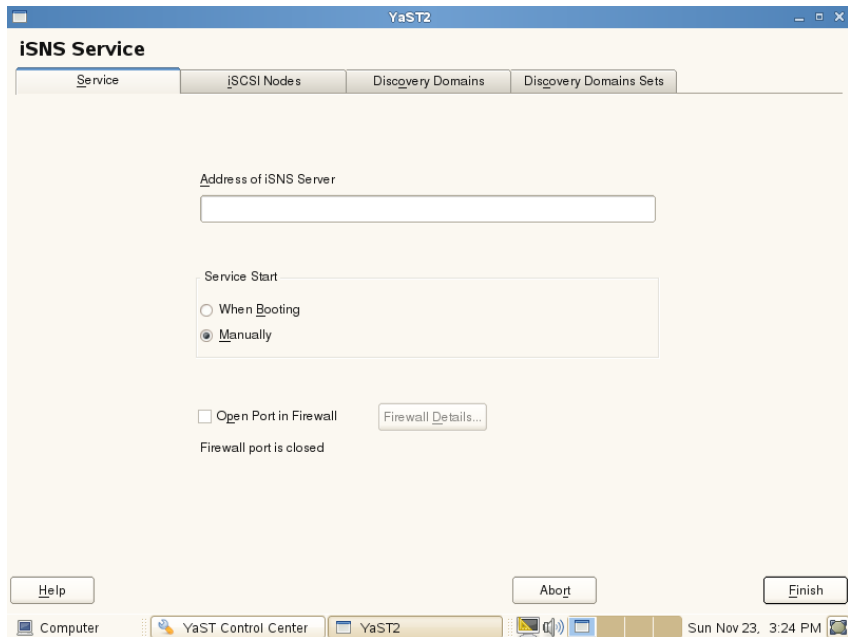
iSNS can be installed on the same server where iSCSI target or iSCSI initiator software is installed. Installing both the iSCSI target software and iSCSI initiator software on the same server is not supported.

To install iSNS for Linux:

- 1 Start YaST and select *Network Services* *iSNS Server*.
- 2 When prompted to install the `isns` package, click *Install*.

- 3 Follow the install dialog instructions to provide the SUSE Linux Enterprise Server 11 installation disks.

When the installation is complete, the iSNS Service configuration dialog opens automatically to the *Service* tab.



- 4 In *Address of iSNS Server*, specify the DNS name or IP address of the iSNS Server.

- 5 In *Service Start*, select one of the following:

- **When Booting:** The iSNS service starts automatically on server startup.
- **Manually (Default):** The iSNS service must be started manually by entering `rcisns start` or `/etc/init.d/isns start` at the server console of the server where you install it.

- 6 Specify the following firewall settings:

- **Open Port in Firewall:** Select the check box to open the firewall and allow access to the service from remote computers. The firewall port is closed by default.
- **Firewall Details:** If you open the firewall port, the port is open on all network interfaces by default. Click *Firewall Details* to select interfaces on which to open the port, select the network interfaces to use, then click *OK*.

7 Click *Finish* to apply the configuration settings and complete the installation.

8 Continue with Section 12.3, “Configuring iSNS Discovery Domains” (page 171).

12.3 Configuring iSNS Discovery Domains

In order for iSCSI initiators and targets to use the iSNS service, they must belong to a discovery domain.

IMPORTANT

The SNS service must be installed and running to configure iSNS discovery domains. For information, see Section 12.4, “Starting iSNS” (page 177).

- Section 12.3.1, “Creating iSNS Discovery Domains” (page 171)
- Section 12.3.2, “Creating iSNS Discovery Domain Sets” (page 173)
- Section 12.3.3, “Adding iSCSI Nodes to a Discovery Domain” (page 175)
- Section 12.3.4, “Adding Discovery Domains to a Discovery Domain Set” (page 177)

12.3.1 Creating iSNS Discovery Domains

A default discovery domain named *default DD* is automatically created when you install the iSNS service. The existing iSCSI targets and initiators that have been configured to use iSNS are automatically added to the default discovery domain.

To create a new discovery domain:

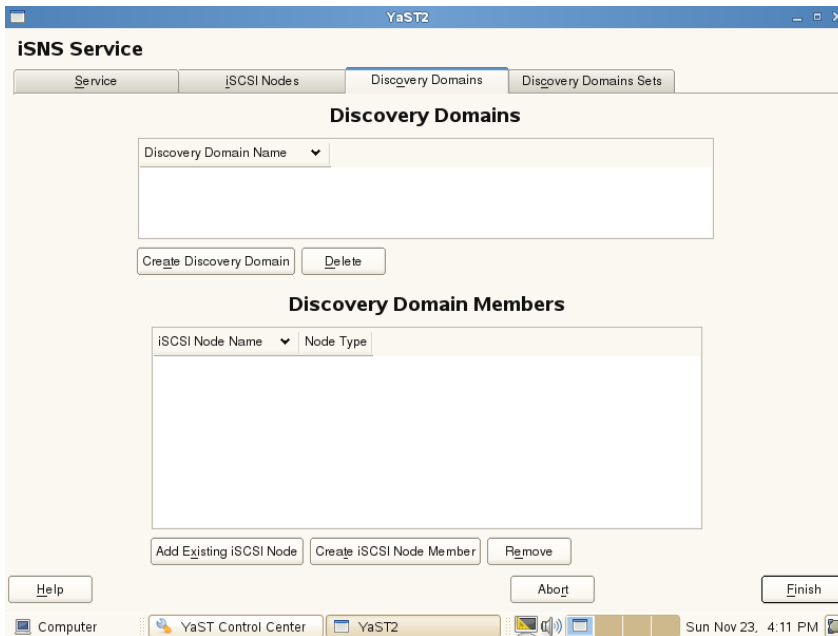
- 1 Start YaST and under *Network Services*, select *iSNS Server*.
- 2 Click the *Discovery Domains* tab.

The *Discovery Domains* area lists all discovery domains. You can create new discovery domains, or delete existing ones. Deleting a domain removes the members from the domain, but it does not delete the iSCSI node members.

The *Discovery Domain Members* area lists all iSCSI nodes assigned to a selected discovery domain. Selecting a different discovery domain refreshes the list with members from that discovery domain. You can add and delete iSCSI nodes from a selected discovery domain. Deleting an iSCSI node removes it from the domain, but it does not delete the iSCSI node.

Creating an iSCSI node allows a node that is not yet registered to be added as a member of the discovery domain. When the iSCSI initiator or target registers this node, then it becomes part of this domain.

When an iSCSI initiator performs a discovery request, the iSNS service returns all iSCSI node targets that are members of the same discovery domain.



3 Click the *Create Discovery Domain* button.

You can also select an existing discovery domain and click the *Delete* button to remove that discovery domain.

4 Specify the name of the discovery domain you are creating, then click *OK*.

5 Continue with Section 12.3.2, “Creating iSNS Discovery Domain Sets” (page 173).

12.3.2 Creating iSNS Discovery Domain Sets

Discovery domains must belong to a discovery domain set. You can create a discovery domain and add nodes to that discovery domain, but it is not active and the iSNS service does not function unless you add the discovery domain to a discovery domain set. A default discovery domain set named *default DDS* is automatically created when you install iSNS and the default discovery domain is automatically added to that domain set.

To create a discovery domain set:

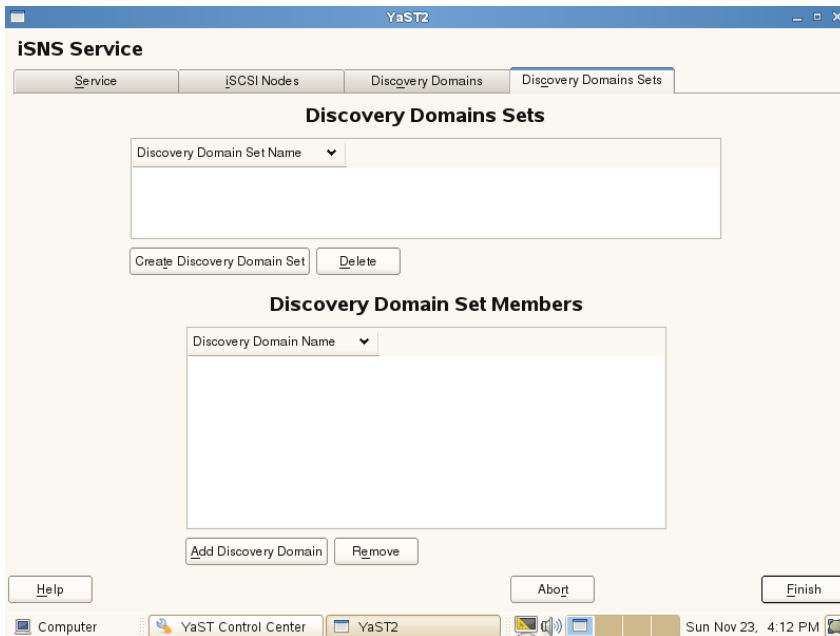
- 1 Start YaST and under *Network Services*, select *iSNS Server*.
- 2 Click the *Discovery Domain Sets* tab.

The *Discovery Domain Sets* area lists all of the discover domain sets. A discovery domain must be a member of a discovery domain set in order to be active.

In an iSNS database, a discovery domain set contains discovery domains, which in turn contains iSCSI node members.

The *Discovery Domain Set Members* area lists all discovery domains that are assigned to a selected discovery domain set. Selecting a different discovery domain set refreshes the list with members from that discovery domain set. You can add and delete discovery domains from a selected discovery domain set. Removing a discovery domain removes it from the domain set, but it does not delete the discovery domain.

Adding an discovery domain to a set allows a not yet registered iSNS discovery domain to be added as a member of the discovery domain set.



3 Click the *Create Discovery Domain Set* button.

You can also select an existing discovery domain set and click the *Delete* button to remove that discovery domain set.

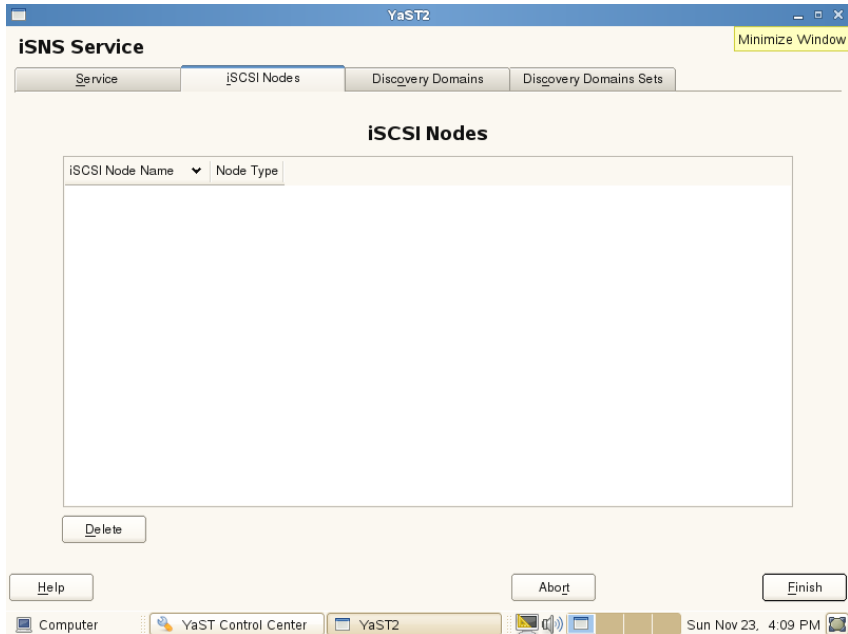
4 Specify the name of the discovery domain set you are creating, then click *OK*.

5 Continue with Section 12.3.3, “Adding iSCSI Nodes to a Discovery Domain” (page 175).

12.3.3 Adding iSCSI Nodes to a Discovery Domain

1 Start YaST and under *Network Services*, select *iSNS Server*.

2 Click the *iSCSI Nodes* tab.



- 3 Review the list of nodes to make sure that the iSCSI targets and initiators that you want to use the iSNS service are listed.

If an iSCSI target or initiator is not listed, you might need to restart the iSCSI service on the node. You can do this by running the `rcopen-iscsi restart` command to restart an initiator or the `rciscsitarget restart` command to restart a target.

You can select an iSCSI node and click the *Delete* button to remove that node from the iSNS database. This is useful if you are no longer using an iSCSI node or have renamed it.

The iSCSI node is automatically added to the list (iSNS database) again when you restart the iSCSI service or reboot the server unless you remove or comment out the iSNS portion of the iSCSI configuration file.

- 4 Click the *Discovery Domains* tab, select the desired discovery domain, then click the *Display Members* button.

- 5 Click *Add existing iSCSI Node*, select the node you want to add to the domain, then click *Add Node*.
- 6 Repeat Step 5 (page 177) for as many nodes as you want to add to the discovery domain, then click *Done* when you are finished adding nodes.

An iSCSI node can belong to more than one discovery domain.

- 7 Continue with Section 12.3.4, “Adding Discovery Domains to a Discovery Domain Set” (page 177).

12.3.4 Adding Discovery Domains to a Discovery Domain Set

- 1 Start YaST and under *Network Services*, select *iSNS Server*.
- 2 Click the *Discovery Domains Set* tab.
- 3 Select *Create Discovery Domain Set* to add a new set to the list of discovery domain sets.
- 4 Choose a discovery domain set to modify.
- 5 Click *Add Discovery Domain*, select the discovery domain you want to add to the discovery domain set, then click *Add Discovery Domain*.
- 6 Repeat the last step for as many discovery domains as you want to add to the discovery domain set, then click *Done*.

A discovery domain can belong to more than one discovery domain set.

12.4 Starting iSNS

iSNS must be started at the server where you install it. Enter one of the following commands at a terminal console as the `root` user:

```
rcisns start
```

```
/etc/init.d/isns start
```

You can also use the `stop`, `status`, and `restart` options with iSNS.

iSNS can also be configured to start automatically each time the server is rebooted:

- 1 Start YaST and under *Network Services*, select *iSNS Server*.
- 2 With the *Service* tab selected, specify the IP address of your iSNS server, then click *Save Address*.
- 3 In the *Service Start* section of the screen, select *When Booting*.

You can also choose to start the iSNS server manually. You must then use the `rcisns start` command to start the service each time the server is restarted.

12.5 Stopping iSNS

iSNS must be stopped at the server where it is running. Enter one of the following commands at a terminal console as the `root` user:

```
rcisns stop
```

```
/etc/init.d/isns stop
```

12.6 For More Information

For information, see the *Linux iSNS for iSCSI project* [<http://sourceforge.net/projects/linuxisns/>]. The electronic mailing list for this project is *Linux iSNS - Discussion* [http://sourceforge.net/mailarchive/forum.php?forum_name=linuxisns-discussion].

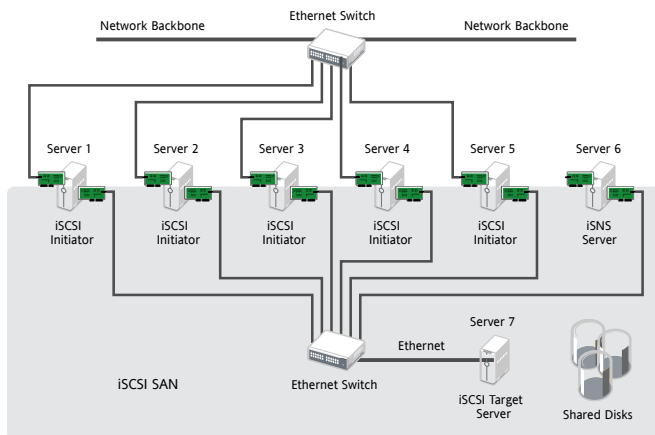
General information about iSNS is available in *RFC 4171: Internet Storage Name Service* [<http://www.ietf.org/rfc/rfc4171>].

Mass Storage over IP Networks: iSCSI

13

One of the central tasks in computer centers and when operating servers is providing hard disk capacity for server systems. Fibre Channel is often used for this purpose. iSCSI (Internet SCSI) solutions provide a lower-cost alternative to Fibre Channel that can leverage commodity servers and Ethernet networking equipment. Linux iSCSI provides iSCSI initiator and target software for connecting Linux servers to central storage systems.

Figure 13.1 *iSCSI SAN with an iSNS Server*



iSCSI is a storage networking protocol that facilitates data transfers of SCSI packets over TCP/IP networks between block storage devices and servers. iSCSI target software runs on the target server and defines the logical units as iSCSI target devices. iSCSI

initiator software runs on different servers and connects to the target devices to make the storage devices available on that server.

IMPORTANT

It is not supported to run iSCSI target software and iSCSI initiator software on the same server in a production environment.

The iSCSI target and initiator servers communicate by sending SCSI packets at the IP level in your LAN. When an application running on the initiator server starts an inquiry for an iSCSI target device, the operating system produces the necessary SCSI commands. The SCSI commands are then embedded in IP packets and encrypted as necessary by software that is commonly known as the *iSCSI initiator*. The packets are transferred across the internal IP network to the corresponding iSCSI remote station, called the *iSCSI target*.

Many storage solutions provide access over iSCSI, but it is also possible to run a Linux server that provides an iSCSI target. In this case, it is important to set up a Linux server that is optimized for file system services. The iSCSI target accesses block devices in Linux. Therefore, it is possible to use RAID solutions to increase disk space as well as a lot of memory to improve data caching. For more information about RAID, also see Chapter 8, *Software RAID Configuration* (page 119).

- Section 13.1, “Installing iSCSI” (page 180)
- Section 13.2, “Setting Up an iSCSI Target” (page 182)
- Section 13.3, “Configuring iSCSI Initiator” (page 191)

13.1 Installing iSCSI

YaST includes entries for iSCSI Target and iSCSI Initiator software, but the packages are not installed by default.

IMPORTANT

It is not supported to run iSCSI target software and iSCSI initiator software on the same server in a production environment.

- Section 13.1.1, “Installing iSCSI Target Software” (page 181)
- Section 13.1.2, “Installing the iSCSI Initiator Software” (page 181)

13.1.1 Installing iSCSI Target Software

Install the iSCSI target software on the server where you want to create iSCSI target devices.

- 1 Open YaST, and log in as the `root` user.
- 2 Select *Network Services* *iSCSI Target*.
- 3 When you are prompted to install the `iscsitarget` package, click *Install*.
- 4 Follow the on-screen install instructions, and provide the installation media as needed.

When the installation is complete, YaST opens to the iSCSI Target Overview page with the *Service* tab selected.

- 5 Continue with Section 13.2, “Setting Up an iSCSI Target” (page 182).

13.1.2 Installing the iSCSI Initiator Software

Install the iSCSI initiator software on each server where you want to access the target devices that you set up on the iSCSI target server.

- 1 Open YaST, and log in as the `root` user.
- 2 Select *Network Services* *iSCSI Initiator*.
- 3 When you are prompted to install the `open-iscsi` package, click *Install*.
- 4 Follow the on-screen install instructions, and provide the installation media as needed.

When the installation is complete, YaST opens to the iSCSI Initiator Overview page with the *Service* tab selected.

- 5 Continue with Section 13.3, “Configuring iSCSI Initiator” (page 191).

13.2 Setting Up an iSCSI Target

SUSE® Linux Enterprise Server comes with an open source iSCSI target solution that evolved from the Ardis iSCSI target. A basic setup can be done with YaST, but to take full advantage of iSCSI, a manual setup is required.

- Section 13.2.1, “Preparing the Storage Space” (page 182)
- Section 13.2.2, “Creating iSCSI Targets with YaST” (page 184)
- Section 13.2.3, “Configuring an iSCSI Target Manually” (page 187)
- Section 13.2.4, “Configuring Online Targets with ietadm” (page 189)

13.2.1 Preparing the Storage Space

The iSCSI target configuration exports existing block devices to iSCSI initiators. You must prepare the storage space you want to use in the target devices by setting up unformatted partitions or devices by using the Partitioner in YaST, or by partitioning the devices from the command line.

IMPORTANT

After you set up a device or partition for use as an iSCSI target, you never access it directly via its local path. Do not specify a mount point for it when you create it.

- Section “Partitioning Devices” (page 183)
- Section “Partitioning Devices in a Virtual Environment” (page 183)

Partitioning Devices

- 1 Log in as the `root` user, then open YaST.
- 2 Select *SystemPartitioner*.
- 3 Click *Yes* to continue through the warning about using the Partitioner.
- 4 Click *Add* to create a partition, but do not format it, and do not mount it.

iSCSI targets can use unformatted partitions with Linux, Linux LVM, or Linux RAID file system IDs.

- 4a Select *Primary Partition*, then click *Next*.
 - 4b Specify the amount of space to use, then click *Next*.
 - 4c Select *Do not format*, then specify the file system ID type.
 - 4d Select *Do not mount*.
 - 4e Click *Finish*.
- 5 Repeat Step 4 (page 183) for each area that you want to use later as an iSCSI LUN.
 - 6 Click *Accept* to keep your changes, then close YaST.

Partitioning Devices in a Virtual Environment

You can use a Xen guest server as the iSCSI target server. You must assign the storage space you want to use for the iSCSI storage devices to the guest virtual machine, then access the space as virtual disks within the guest environment. Each virtual disk can be a physical block device, such as an entire disk, partition, or volume, or it can be a file-backed disk image where the virtual disk is a single image file on a larger physical disk on the Xen host server. For the best performance, create each virtual disk from a physical disk or a partition. After you set up the virtual disks for the guest virtual ma-

chine, start the guest server, then configure the new blank virtual disks as iSCSI target devices by following the same process as for a physical server.

File-baked disk images are created on the Xen host server, then assigned to the Xen guest server. By default, Xen stores file-backed disk images in the `/var/lib/xen/images/vm_name` directory, where `vm_name` is the name of the virtual machine.

For example, if you want to create the disk image `/var/lib/xen/images/vm_one/xen-0` with a size of 4 GB, first make sure that the directory is there, then create the image itself.

- 1 Log in to the host server as the `root` user.
- 2 At a terminal console prompt, enter the following commands

```
mkdir -p /var/lib/xen/images/vm_one
dd if=/dev/zero of=/var/lib/xen/images/vm_one/xen-0 seek=1M bs=4096
count=1
```

- 3 Assign the file system image to the guest virtual machine in the Xen configuration file.
- 4 Log in as the `root` user on the guest server, then use YaST to set up the virtual block device by using the process in Section “Partitioning Devices” (page 183).

13.2.2 Creating iSCSI Targets with YaST

- 1 Open YaST, and log in as the `root` user.
- 2 Select *Network Services* *iSCSI Target*.

YaST opens to the iSCSI Target Overview page with the *Service* tab selected

- 3 In the *Service Start* area, select one of the following:
 - **When booting:** Automatically start the initiator service on subsequent server reboots.
 - **Manually (default):** Start the service manually.

4 If you are using iSNS for target advertising, select the *iSNS Access Control* check box, then type the IP address.

5 If desired, open the firewall ports to allow access to the server from remote computers.

5a Select the *Open Port in Firewall* check box.

5b Specify the network interfaces where you want to open the port by clicking *Firewall Details*, selecting the check box next to a network interface to enable it, then clicking *OK* to accept the settings.

6 If authentication is required to connect to target devices you set up on this server, select the *Global* tab, deselect *No Authentication* to enable authentication, then specify the necessary credentials for incoming and outgoing authentication.

The *No Authentication* option is enabled by default. You can specify authentication for incoming, outgoing, or both incoming and outgoing. You can also specify multiple sets of credentials for incoming authentication by adding pairs of usernames and passwords to the list under *Incoming Authentication*.

7 Configure the iSCSI target devices.

7a Select the *Targets* tab.

7b If you have not already done so, select and delete the example iSCSI target from the list, then confirm the deletion by clicking *Continue*.

7c Click *Add* to add a new iSCSI target.

The iSCSI target automatically presents an unformatted partition or block device and completes the Target and Identifier fields.

7d You can accept this, or browse to select a different space.

You can also subdivide the space to create LUNs on the device by clicking *Add* and specifying sectors to allocate to that LUN. If you need additional options for these LUNs, select *Expert Settings*.

7e Click *Next*

7f Repeat Step 7c (page 185) to Step 7e (page 185) for each iSCSI target device you want to create.

7g (Optional) On the *Service* tab, click *Save* to export the information about the configured iSCSI targets to a file.

This makes it easier to later provide this information to consumers of the resources.

7h Click *Finish* to create the devices, then click *Yes* to restart the iSCSI software stack.

To configure the iSCSI target, run the *iSCSI Target* module in YaST. The configuration is split into three tabs. In the *Service* tab, select the start mode and the firewall settings. If you want to access the iSCSI target from a remote machine, select *Open Port in Firewall*. If an iSNS server should manage the discovery and access control, activate *iSNS Access Control* and enter the IP address of your iSNS server. You cannot use hostnames, but must use the IP address. For more about iSNS, read Chapter 12, *iSNS for Linux* (page 167).

The *Global* tab provides settings for the iSCSI server. The authentication set here is used for the discovery of services, not for accessing the targets. If you do not want to restrict the access to the discovery, use *No Authentication*.

If authentication is needed, there are two possibilities to consider. One is that an initiator must prove that it has the permissions to run a discovery on the iSCSI target. This is done with *Incoming Authentication*. The other is that the iSCSI target must prove to the initiator that it is the expected target. Therefore, the iSCSI target can also provide a username and password. This is done with *Outgoing Authentication*. Find more information about authentication in *RFC 3720* [<http://www.ietf.org/rfc/rfc3720.txt>].

The targets are defined in the *Targets* tab. Use *Add* to create a new iSCSI target. The first dialog asks for information about the device to export.

Target

The *Target* line has a fixed syntax that looks like the following:

```
iqn.yyyy-mm.<reversed domain name>
```

It always starts with `iqn.yyyy-mm` is the format of the date when this target is activated. Find more about naming conventions in *RFC 3722* [<http://www.ietf.org/rfc/rfc3722.txt>].

Identifier

The *Identifier* is freely selectable. It should follow some scheme to make the whole system more structured.

LUN

It is possible to assign several LUNs to a target. To do this, select a target in the *Targets* tab, then click *Edit*. Then, add new LUNs to an existing target.

Path

Add the path to the block device or file system image to export.

The next menu configures the access restrictions of the target. The configuration is very similar to the configuration of the discovery authentication. In this case, at least an incoming authentication should be setup.

Next finishes the configuration of the new target, and brings you back to the overview page of the *Target* tab. Activate your changes by clicking *Finish*.

13.2.3 Configuring an iSCSI Target Manually

Configure an iSCSI target in `/etc/ietd.conf`. All parameters in this file before the first *Target* declaration are global for the file. Authentication information in this portion has a special meaning—it is not global, but is used for the discovery of the iSCSI target.

If you have access to an iSNS server, you should first configure the file to tell the target about this server. The address of the iSNS server must always be given as an IP address. You cannot specify the DNS name for the server. The configuration for this functionality looks like the following:

```
iSNSServer 192.168.1.111
iSNSAccessControl no
```

This configuration makes the iSCSI target register itself with the iSNS server, which in turn provides the discovery for initiators. For more about iSNS, see Chapter 12, *iSMS*

for *Linux* (page 167). The access control for the iSNS discovery is not supported. Just keep `iSNSAccessControl no`.

All direct iSCSI authentication can be done in two directions. The iSCSI target can require the iSCSI initiator to authenticate with the `IncomingUser`, which can be added multiple times. The iSCSI initiator can also require the iSCSI target to authenticate. Use `OutgoingUser` for this. Both have the same syntax:

```
IncomingUser <username> <password>
OutgoingUser <username> <password>
```

The authentication is followed by one or more target definitions. For each target, add a `Target` section. This section always starts with a `Target` identifier followed, by definitions of logical unit numbers:

```
Target iqn.yyyy-mm.<reversed domain name>[:identifier]
    Lun 0 Path=/dev/mapper/system-v3
    Lun 1 Path=/dev/hda4
    Lun 2 Path=/var/lib/xen/images/xen-1,Type=fileio
```

In the `Target` line, `yyyy-mm` is the date when this target is activated, and `identifier` is freely selectable. Find more about naming conventions in *RFC 3722* [<http://www.ietf.org/rfc/rfc3722.txt>]. Three different block devices are exported in this example. The first block device is a logical volume (see also Chapter 4, *LVM Configuration* (page 29)), the second is an IDE partition, and the third is an image available in the local file system. All these look like block devices to an iSCSI initiator.

Before activating the iSCSI target, add at least one `IncomingUser` after the `Lun` definitions. It does the authentication for the use of this target.

To activate all your changes, restart the `iscsitarget` daemon with `rcopen-iscsi restart`. Check your configuration in the `/proc` file system:

```
cat /proc/net/iet/volume
tid:1 name:iqn.2006-02.com.example.iserv:systems
    lun:0 state:0 iotype:fileio path:/dev/mapper/system-v3
    lun:1 state:0 iotype:fileio path:/dev/hda4
    lun:2 state:0 iotype:fileio path:/var/lib/xen/images/xen-1
```

There are many more options that control the behavior of the iSCSI target. For more information, see the man page of `ietd.conf`.

Active sessions are also displayed in the `/proc` file system. For each connected initiator, an extra entry is added to `/proc/net/iet/session`:

```
cat /proc/net/iet/session
tid:1 name:iqn.2006-02.com.example.iserv:system-v3
    sid:562949957419520
initiator:iqn.2005-11.de.suse:cn=rome.example.com,01.9ff842f5645
    cid:0 ip:192.168.178.42 state:active hd:none dd:none
    sid:281474980708864 initiator:iqn.2006-02.de.suse:01.6f7259c88b70
    cid:0 ip:192.168.178.72 state:active hd:none dd:none
```

13.2.4 Configuring Online Targets with `ietadm`

When changes to the iSCSI target configuration are necessary, you must always restart the target to activate changes that are done in the configuration file. Unfortunately, all active sessions are interrupted in this process. To maintain an undisturbed operation, the changes should be done in the main configuration file `/etc/ietd.conf`, but also made manually to the current configuration with the administration utility `ietadm`.

To create a new iSCSI target with a LUN, first update your configuration file. The additional entry could be:

```
Target iqn.2006-02.com.example.iserv:system2
    Lun 0 Path=/dev/mapper/system-swap2
    IncomingUser joe secret
```

To set up this configuration manually, proceed as follows:

- 1 Create a new target with the command `ietadm --op new --tid=2 --params Name=iqn.2006-02.com.example.iserv:system2`.
- 2 Add a logical unit with `ietadm --op new --tid=2 --lun=0 --params Path=/dev/mapper/system-swap2`.
- 3 Set the username and password combination on this target with `ietadm --op new --tid=2 --user --params=IncomingUser=joe,Password=secret`.
- 4 Check the configuration with `cat /proc/net/iet/volume`.

It is also possible to delete active connections. First, check all active connections with the command `cat /proc/net/iet/session`. This might look like:

```
cat /proc/net/iet/session
tid:1 name:iqn.2006-03.com.example.iserv:system
      sid:281474980708864 initiator:iqn.1996-04.com.example:01.82725735af5
      cid:0 ip:192.168.178.72 state:active hd:none dd:none
```

To delete the session with the session ID 281474980708864, use the command `ietadm --op delete --tid=1 --sid=281474980708864 --cid=0`. Be aware that this makes the device inaccessible on the client system and processes accessing this device are likely to hang.

`ietadm` can also be used to change various configuration parameters. Obtain a list of the global variables with `ietadm --op show --tid=1 --sid=0`. The output looks like:

```
InitialR2T=Yes
ImmediateData=Yes
MaxConnections=1
MaxRecvDataSegmentLength=8192
MaxXmitDataSegmentLength=8192
MaxBurstLength=262144
FirstBurstLength=65536
DefaultTime2Wait=2
DefaultTime2Retain=20
MaxOutstandingR2T=1
DataPDUInOrder=Yes
DataSequenceInOrder=Yes
ErrorRecoveryLevel=0
HeaderDigest=None
DataDigest=None
OFMarker=No
IFMarker=No
OFMarkInt=Reject
IFMarkInt=Reject
```

All of these parameters can be easily changed. For example, if you want to change the maximum number of connections to two, use

```
ietadm --op update --tid=1 --params=MaxConnections=2.
```

In the file `/etc/ietd.conf`, the associated line should look like `MaxConnections 2`.

WARNING

The changes that you make with the `ietadm` utility are not permanent for the system. These changes are lost at the next reboot if they are not added to the `/etc/ietd.conf` configuration file. Depending on the usage of iSCSI in your network, this might lead to severe problems.

There are several more options available for the `ietadm` utility. Use `ietadm -h` to find an overview. The abbreviations there are target ID (`tid`), session ID (`sid`), and connection ID (`cid`). They can also be found in `/proc/net/iet/session`.

13.3 Configuring iSCSI Initiator

The iSCSI initiator, also called an iSCSI client, can be used to connect to any iSCSI target. This is not restricted to the iSCSI target solution explained in Section 13.2, “Setting Up an iSCSI Target” (page 182). The configuration of iSCSI initiator involves two major steps: the discovery of available iSCSI targets and the setup of an iSCSI session. Both can be done with YaST.

- Section 13.3.1, “Using YaST for the iSCSI Initiator Configuration” (page 191)
- Section 13.3.2, “Setting Up the iSCSI Initiator Manually” (page 195)
- Section 13.3.3, “The iSCSI Client Databases” (page 196)
- Section 13.3.4, “For More Information” (page 198)

13.3.1 Using YaST for the iSCSI Initiator Configuration

The iSCSI Initiator Overview in YaST is divided into three tabs:

- **Service:** The *Service* tab can be used to enable the iSCSI initiator at boot time. It also offers to set a unique *Initiator Name* and an iSNS server to use for the discovery. The default port for iSNS is 3205.

- **Connected Targets:** The *Connected Targets* tab gives an overview of the currently connected iSCSI targets. Like the *Discovered Targets* tab, it also gives the option to add new targets to the system.

On this page, you can select a target device, then toggle the start-up setting for each iSCSI target device:

- **Automatic:** This option is used for iSCSI targets that are to be connected when the iSCSI service itself starts up. This is the typical configuration.
- **Onboot:** This option is used for iSCSI targets that are to be connected during boot; that is, when root (/) is on iSCSI. As such, the iSCSI target device will be evaluated from the initrd on server boots.
- **Discovered Targets:** *Discovered Targets* provides the possibility of manually discovering iSCSI targets in the network.
- Section “Configuring the iSCSI Initiator” (page 192)
- Section “Discovering iSCSI Targets by Using iSNS” (page 193)
- Section “Discovering iSCSI Targets Manually” (page 194)
- Section “Setting the Start-up Preference for iSCSI Target Devices” (page 195)

Configuring the iSCSI Initiator

1 Open YaST, and log in as the `root` user.

2 Select *Network Services* *iSCSI Initiator*.

YaST opens to the iSCSI Initiator Overview page with the *Service* tab selected.

3 In the *Service Start* area, select one of the following:

- **When booting:** Automatically start the initiator service on subsequent server reboots.
- **Manually (default):** Start the service manually.

4 Specify or verify the *Initiator Name*.

Specify a well-formed initiator qualified name (IQN) for the iSCSI initiator on this server. The initiator name must be globally unique on your network. The IQN uses the following general format:

```
iqn.yyyy-mm.com.mycompany:n1:n2
```

where n1 and n2 are alphanumeric characters. For example:

```
iqn.1996-04.de.suse:01:9c83a3e15f64
```

The *Initiator Name* is automatically completed with the corresponding value from the `/etc/iscsi/initiatorname.iscsi` file on the server.

If the server has iBFT (iSCSI Boot Firmware Table) support, the *Initiator Name* is completed with the corresponding value in the IBFT, and you are not able to change the initiator name in this interface. Use the BIOS Setup to modify it instead. The iBFT is a block of information containing various parameters useful to the iSCSI boot process, including iSCSI target and initiator descriptions for the server.

5 Use either of the following methods to discover iSCSI targets on the network.

- **iSNS:** To use iSNS (Internet Storage Name Service) for discovering iSCSI targets, continue with Section “Discovering iSCSI Targets by Using iSNS” (page 193).
- **Discovered Targets:** To discover iSCSI target devices manually, continue with Section “Discovering iSCSI Targets Manually” (page 194).

Discovering iSCSI Targets by Using iSNS

Before you can use this option, you must have already installed and configured an iSNS server in your environment. For information, see Chapter 12, *iSNS for Linux* (page 167).

- 1 In YaST, select *iSCSI Initiator*, then select the *Service* tab.
- 2 Specify the IP address of the iSNS server and port.

The default port is 3205.

- 3 On the iSCSI Initiator Overview page, click *Finish* to save and apply your changes.

Discovering iSCSI Targets Manually

Repeat the following process for each of the iSCSI target servers that you want to access from the server where you are setting up the iSCSI initiator.

- 1 In YaST, select *iSCSI Initiator*, then select the *Discovered Targets* tab.
- 2 Click *Discovery* to open the *iSCSI Initiator Discovery* dialog.
- 3 Enter the IP address and change the port if needed.

The default port is 3260.

- 4 If authentication is required, deselect *No Authentication*, then specify the credentials the *Incoming* or *Outgoing* authentication.
- 5 Click *Next* to start the discovery and connect to the iSCSI target server.
- 6 If credentials are required, after a successful discovery, use *Login* to activate the target.

You are prompted for authentication credentials to use the selected iSCSI target.

- 7 Click *Next* to finish the configuration.

If everything went well, the target now appears in *Connected Targets*.

The virtual iSCSI device is now available.

- 8 On the iSCSI Initiator Overview page, click *Finish* to save and apply your changes.
- 9 You can find the local device path for the iSCSI target device by using the `lsscsi` command:

```
lsscsi
[1:0:0:0] disk IET VIRTUAL-DISK 0 /dev/sda
```

Setting the Start-up Preference for iSCSI Target Devices

- 1 In YaST, select *iSCSI Initiator*, then select the *Connected Targets* tab to view a list of the iSCSI target devices that are currently connected to the server.
- 2 Select the iSCSI target device that you want to manage.
- 3 Click *Toggle Start-Up* to modify the setting:
 - **Automatic:** This option is used for iSCSI targets that are to be connected when the iSCSI service itself starts up. This is the typical configuration.
 - **Onboot:** This option is used for iSCSI targets that are to be connected during boot; that is, when root (/) is on iSCSI. As such, the iSCSI target device will be evaluated from the `initrd` on server boots.
- 4 Click *Finish* to save and apply your changes.

13.3.2 Setting Up the iSCSI Initiator Manually

Both the discovery and the configuration of iSCSI connections require a running `iscsid`. When running the discovery the first time, the internal database of the iSCSI initiator is created in the directory `/var/lib/open-iscsi`.

If your discovery is password protected, provide the authentication information to `iscsid`. Because the internal database does not exist when doing the first discovery, it cannot be used at this time. Instead, the configuration file `/etc/iscsid.conf` must be edited to provide the information. To add your password information for the discovery, add the following lines to the end of `/etc/iscsid.conf`:

```
discovery.sendtargets.auth.authmethod = CHAP
```

```
discovery.sendtargets.auth.username = <username>
discovery.sendtargets.auth.password = <password>
```

The discovery stores all received values in an internal persistent database. In addition, it displays all detected targets. Run this discovery with the command `iscsiadm -m discovery --type=st --portal=<targetip>`. The output should look like:

```
149.44.171.99:3260,1 iqn.2006-02.com.example.iserv:systems
```

To discover the available targets on a iSNS server, use the command `iscsiadm --mode discovery --type isns --portal <targetip>`

For each target defined on the iSCSI target, one line appears. For more information about the stored data, see Section 13.3.3, “The iSCSI Client Databases” (page 196).

The special `--login` option of `iscsiadm` creates all needed devices:

```
iscsiadm -m node -n iqn.2006-02.com.example.iserv:systems --login
```

The newly generated devices show up in the output of `ls SCSI` and can now be accessed by `mount`.

13.3.3 The iSCSI Client Databases

All information that was discovered by the iSCSI initiator is stored in two database files that reside in `/var/lib/open-iscsi`. There is one database for the discovery of targets and one for the discovered nodes. When accessing a database, you first must select if you want to get your data from the discovery or from the node database. Do this with the `-m discovery` and `-m node` parameters of `iscsiadm`. Using `iscsiadm` just with one of these parameters gives an overview of the stored records:

```
iscsiadm -m discovery
149.44.171.99:3260,1 iqn.2006-02.com.example.iserv:systems
```

The target name in this example is

`iqn.2006-02.com.example.iserv:systems`. This name is needed for all actions that relate to this special data set. To examine the content of the data record

with the ID `iqn.2006-02.com.example.iserv:systems`, use the following command:

```
iscsiadm -m node --targetname iqn.2006-02.com.example.iserv:systems
node.name = iqn.2006-02.com.example.iserv:systems
node.transport_name = tcp
node.tpgt = 1
node.active_conn = 1
node.startup = manual
node.session.initial_cmds_n = 0
node.session.reopen_max = 32
node.session.auth.authmethod = CHAP
node.session.auth.username = joe
node.session.auth.password = *****
node.session.auth.username_in = <empty>
node.session.auth.password_in = <empty>
node.session.timeo.replacement_timeout = 0
node.session.err_timeo.abort_timeout = 10
node.session.err_timeo.reset_timeout = 30
node.session.iscsi.InitialR2T = No
node.session.iscsi.ImmediateData = Yes
....
```

To edit the value of one of these variables, use the command `iscsiadm` with the `update` operation. For example, if you want `iscsid` to log in to the iSCSI target when it initializes, set the variable `node.startup` to the value `automatic`:

```
iscsiadm -m node -n iqn.2006-02.com.example.iserv:systems --op=update
--name=node.startup --value=automatic
```

Remove obsolete data sets with the `delete` operation If the target `iqn.2006-02.com.example.iserv:systems` is no longer a valid record, delete this record with the following command:

```
iscsiadm -m node -n iqn.2006-02.com.example.iserv:systems --op=delete
```

IMPORTANT

Use this option with caution because it deletes the record without any additional confirmation prompt.

To get a list of all discovered targets, run the `iscsiadm -m node` command.

13.3.4 For More Information

The iSCSI protocol has been available for several years. There are many reviews and additional documentation comparing iSCSI with SAN solutions, doing performance benchmarks, or just describing hardware solutions. Important pages for more information about open-iscsi are:

- *Open-iSCSI Project* [<http://www.open-iscsi.org/>]
- *AppNote: iFolder on Open Enterprise Server Linux Cluster using iSCSI* [<http://www.novell.com/coolsolutions/appnote/15394.html>]

There is also some online documentation available. See the man pages for `iscsiadm`, `iscsid`, `ietd.conf`, and `ietd` and the example configuration file `/etc/iscsid.conf`.

Volume Snapshots

A file system snapshot is a copy-on-write technology that monitors changes to an existing volume's data blocks so that when a write is made to one of the blocks, the block's value at the snapshot time is copied to a snapshot volume. In this way, a point-in-time copy of the data is preserved until the snapshot volume is deleted.

- Section 14.1, “Understanding Volume Snapshots” (page 199)
- Section 14.2, “Creating Linux Snapshots with LVM” (page 201)
- Section 14.3, “Monitoring a Snapshot” (page 201)
- Section 14.4, “Deleting Linux Snapshots” (page 202)

14.1 Understanding Volume Snapshots

A file system snapshot contains metadata about and data blocks from an original volume that have changed since the snapshot was taken. When you access data via the snapshot, you see a point-in-time copy the original volume. There is no need to restore data from backup media or to overwrite the changed data.

In a Xen host environment, the virtual machine must be using an LVM logical volume as its storage back-end, as opposed to using a virtual disk file.

Linux snapshots allow you to create a backup from a point-in-time view of the file system. The snapshot is created instantly and persists until you delete it. You can backup the file system from the snapshot while the volume itself continues to be available for users. The snapshot initially contains some metadata about the snapshot, but no actual data from the original volume. Snapshot uses copy-on-write technology to detect when data changes in an original data block. It copies the value it held when the snapshot was taken to a block in the snapshot volume, then allows the new data to be stored in the original block. As blocks change from their original value, the snapshot size grows.

When you are sizing the snapshot, consider how much data is expected to change on the original volume and how long you plan to keep the snapshot. The amount of space that you allocate for a snapshot volume can vary, depending on the size of the original volume, how long you plan to keep the snapshot, and the number of data blocks that are expected to change during the snapshot's lifetime. The snapshot volume cannot be resized after it is created. As a guide, create a snapshot volume that is about 10% of the size of the original logical volume. If you anticipate that every block in the original volume will change at least one time before you delete the snapshot, then the snapshot volume should be at least as large as the original volume plus some additional space for metadata about the snapshot volume. Less space is required if the data changes infrequently or if the expected lifetime is sufficiently brief.

IMPORTANT

During the snapshot's lifetime, the snapshot must be mounted before its original volume can be mounted.

When you are done with the snapshot, it is important to remove it from the system. A snapshot eventually fills up completely as data blocks change on the original volume. When the snapshot is full, it is disabled, which prevents you from remounting the original volume.

Remove snapshots in a last created, first deleted order.

14.2 Creating Linux Snapshots with LVM

The Logical Volume Manager (LVM) can be used for creating snapshots of your file system.

- Open a terminal console, log in as the `root` user, then enter

```
lvcreate -s -L 1G -n snap_volume source_volume_path
```

For example:

```
lvcreate -s -L 1G -n linux01-snap /dev/lvm/linux01
```

The snapshot is created as the `/dev/lvm/linux01-snap` volume.

14.3 Monitoring a Snapshot

- Open a terminal console, log in as the `root` user, then enter

```
lvdisplay snap_volume
```

For example:

```
lvdisplay /dev/vg01/linux01-snap
```

```
--- Logical volume ---
LV Name                /dev/lvm/linux01
VG Name                vg01
LV UUID                QHVJYh-PR3s-A4SG-s4Aa-MyWN-Ra7a-HL47KL
LV Write Access        read/write
LV snapshot status     active destination for /dev/lvm/linux01
LV Status              available
# open                 0
LV Size                80.00 GB
Current LE             1024
COW-table size         8.00 GB
COW-table LE           512
Allocated to snapshot 30%
Snapshot chunk size    8.00 KB
Segments               1
```

Allocation	inherit
Read ahead sectors	0
Block device	254:5

14.4 Deleting Linux Snapshots

- Open a terminal console, log in as the `root` user, then enter

```
lvremove snap_volume_path
```

For example:

```
lvremove /dev/lvm/linux01-snap
```

Troubleshooting Storage Issues

15

This section describes how to work around known issues for devices, software RAIDs, multipath I/O, and volumes.

- Section 15.1, “Is DM-MPIO Available for the Boot Partition?” (page 203)

15.1 Is DM-MPIO Available for the Boot Partition?

Device Mapper Multipath I/O (DM-MPIO) is supported for the boot partition, beginning in SUSE® Linux Enterprise Server 10 Support Pack 1.



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 11. 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, “May 2010 (SLES 11 SP1)” (page 206)
- Section A.2, “February 23, 2010” (page 207)
- Section A.3, “January 20, 2010” (page 208)
- Section A.4, “December 1, 2009” (page 209)
- Section A.5, “October 20, 2009” (page 211)
- Section A.6, “August 3, 2009” (page 212)
- Section A.7, “June 22, 2009” (page 213)
- Section A.8, “May 21, 2009” (page 215)

A.1 May 2010 (SLES 11 SP1)

Updates were made to the following sections. The changes are explained below.

- Section A.1.1, “Managing Multipath I/O for Devices” (page 206)
- Section A.1.2, “Mass Storage over IP Networks: iSCSI” (page 207)
- Section A.1.3, “What’s New” (page 207)

A.1.1 Managing Multipath I/O for Devices

Location	Change
Section 7.2.3, “Using LVM2 on Multipath Devices” (page 60)	The example in Step 3 (page 60) was corrected.
Section 7.2.6, “SAN Timeout Settings When the Root Device Is Multipathed” (page 62)	This section is new.
Section 7.3.2, “Multipath I/O Management Tools” (page 70)	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 <i>SUSE Linux Enterprise Server Technical Specifications Package Descriptions</i> Web page [http://www.novell.com/products/server/techspecs.html], find your architecture and select <i>Packages Sorted by Name</i> , then search on “multipath-tools” to find the package list for that architecture.
Section 7.4.1, “Preparing SAN Devices for Multipathing” (page 76)	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 7.2.6, “SAN Timeout Settings When the Root Device Is Multipathed” (page 62).

Location	Change
Section 7.8.1, “Enabling Multipath I/O to Install SLES on a Multipath Storage LUN” (page 101)	This section is new.
Section 7.8.2, “Enabling Multipath I/O to Install SLES on an Active/Passive Multipath Storage LUN” (page 101)	This section is new.

A.1.2 Mass Storage over IP Networks: iSCSI

Location	Change
Step 7g (page 186) in Section 13.2.2, “Creating iSCSI Targets with YaST” (page 184)	In the <i>YaSTNetwork ServicesiSCSI Target</i> function, the <i>Save</i> option allows you to export the iSCSI target information, which makes it easier to provide this information to consumers of the resources.

A.1.3 What’s New

Location	Change
Section 2.1, “What’s New in SLES 11 SP1” (page 15)	This section is new.

A.2 February 23, 2010

Updates were made to the following sections. The changes are explained below.

- Section A.2.1, “Configuring Software RAID for the Root Partition” (page 208)
- Section A.2.2, “Managing Multipath I/O” (page 208)

A.2.1 Configuring Software RAID for the Root Partition

Location	Change
Section 9.1, “Prerequisites for the Software RAID” (page 127)	Corrected an error in the RAID 0 definition..

A.2.2 Managing Multipath I/O

Location	Change
Section 7.10, “Scanning for New Devices without Rebooting” (page 108)	Added information about using the <code>rescan -scsi-bus.sh</code> script to scan for devices without rebooting.
Section 7.11, “Scanning for New Partitioned Devices without Rebooting” (page 111)	Added information about using the <code>rescan -scsi-bus.sh</code> script to scan for devices without rebooting.

A.3 January 20, 2010

Updates were made to the following section. The changes are explained below.

- Section A.3.1, “Managing Multipath I/O” (page 209)

A.3.1 Managing Multipath I/O

Location	Change
Section “Configuring Default Multipath Behavior in <code>/etc/multipath.conf</code> ” (page 84)	In the <code>default_getuid</code> command line, use the path <code>/sbin/scsi_id</code> as shown in the above example instead of the sample path of <code>/lib/udev/scsi_id</code> that is found in the sample file <code>/usr/share/doc/packages/multipath-tools/multipath.conf.synthetic</code> file (and in the default and annotated sample files).
<code>getuid</code> in Table 7.5, “Multipath Attributes” (page 88)	Use the path <code>/sbin/scsi_id</code> instead of the path <code>/lib/udev/sbin_id</code> that is in the sample file <code>/usr/share/doc/packages/multipath-tools/multipath.conf.synthetic</code> file (and in the default and annotated sample files).

A.4 December 1, 2009

Updates were made to the following sections. The changes are explained below.

- Section A.4.1, “Managing Multipath I/O for Devices” (page 210)
- Section A.4.2, “Resizing File Systems” (page 210)
- Section A.4.3, “What’s New” (page 211)

A.4.1 Managing Multipath I/O for Devices

Location	Change
Section 7.2.3, “Using LVM2 on Multipath Devices” (page 60)	The <code>-f mpath</code> option changed to <code>-f multipath</code> : <code>mkinitrd -f multipath</code>
Section 7.9, “Configuring Multipath I/O for an Existing Software RAID” (page 105)	
prio_callout in Table 7.5, “Multipath Attributes” (page 88)	Multipath prio_callouts are located in shared libraries in <code>/lib/libmultipath/lib*</code> . By using shared libraries, the callouts are loaded into memory on daemon startup.

A.4.2 Resizing File Systems

Location	Change
Section 5.1.1, “File Systems that Support Resizing” (page 40)	The <code>resize2fs</code> utility supports online or offline resizing for the <code>ext3</code> file system.

A.4.3 What's New

Location	Change
Section 2.2.10, “Location Change for Multipath Tool Callouts” (page 23)	This section is new.
Section 2.2.11, “Change from mpath to multipath for the mkinitrd -f Option” (page 24)	This section is new.

A.5 October 20, 2009

Updates were made to the following sections. The changes are explained below.

- Section A.5.1, “LVM Configuration” (page 211)
- Section A.5.2, “Managing Multipath I/O for Devices” (page 212)
- Section A.5.3, “What’s New” (page 212)

A.5.1 LVM Configuration

Location	Change
Section 4.6, “Direct LVM Management” (page 36)	In the YaST Control Center, select <i>System > Partitioner</i> .

A.5.2 Managing Multipath I/O for Devices

Location	Change
Section “Blacklisting Non-Multipathed Devices in <code>/etc/multipath.conf</code> ” (page 83)	The keyword <code>devnode_blacklist</code> has been deprecated and replaced with the keyword <code>blacklist</code> .
Section “Configuring Default Multipath Behavior in <code>/etc/multipath.conf</code> ” (page 84)	Changed <code>getuid_callout</code> to <code>getuid</code> .
Section “Understanding Priority Groups and Attributes” (page 88)	Changed <code>getuid_callout</code> to <code>getuid</code> .
Section “Understanding Priority Groups and Attributes” (page 88)	Added descriptions of least-pending, length-load-balancing, and service-time options.

A.5.3 What’s New

Location	Change
Section 2.2.9, “Advanced I/O Load-Balancing Options for Multipath” (page 23)	This section is new.

A.6 August 3, 2009

Updates were made to the following section. The change is explained below.

- Section A.6.1, “Managing Multipath I/O” (page 213)

A.6.1 Managing Multipath I/O

Location	Change
Section 7.2.5, “Using --noflush with Multipath Devices” (page 62)	This section is new.

A.7 June 22, 2009

Updates were made to the following sections. The changes are explained below.

- Section A.7.1, “Managing Multipath I/O” (page 213)
- Section A.7.2, “Managing Software RAIDs 6 and 10 with mdadm” (page 214)
- Section A.7.3, “Mass Storage over IP Networks: iSCSI” (page 214)

A.7.1 Managing Multipath I/O

Location	Change
Section 7.8, “Configuring Multipath I/O for the Root Device” (page 100)	Added Step 4 (page 104) and Step 6 (page 105) for System Z.
Section 7.11, “Scanning for New Partitioned Devices without Rebooting” (page 111)	Corrected the syntax for the command lines in Step 2.

Location	Change
Section 7.11, “Scanning for New Partitioned Devices without Re-booting” (page 111)	Step 7 (page 112) replaces old Step 7 and Step 8.

A.7.2 Managing Software RAIDs 6 and 10 with mdadm

Location	Change
Section 10.4, “Creating a Degraded RAID Array” (page 147)	To see the rebuild progress while being refreshed every second, enter <pre>watch -n 1 cat /proc/mdstat</pre>

A.7.3 Mass Storage over IP Networks: iSCSI

Location	Change
Section 13.3.1, “Using YaST for the iSCSI Initiator Configuration” (page 191)	<p>Re-organized material for clarity.</p> <p>Added information about how to use the settings for the Start-up option for iSCSI target devices:</p> <ul style="list-style-type: none"> • Automatic: This option is used for iSCSI targets that are to be connected when the iSCSI service itself starts up. This is the typical configuration. • Onboot: This option is used for iSCSI targets that are to be connected during boot; that is, when root (/) is on iSCSI. As such,

Location	Change
	the iSCSI target device will be evaluated from the <code>initrd</code> on server boots.

A.8 May 21, 2009

Updates were made to the following section. The changes are explained below.

- Section A.8.1, “Managing Multipath I/O” (page 215)

A.8.1 Managing Multipath I/O

Location	Change
Section “Storage Arrays That Are Automatically Detected for Multipathing” (page 64)	Testing of the IBM zSeries device with multipathing has shown that the <code>dev_loss_tmo</code> parameter should be set to 90 seconds, and the <code>fast_io_fail_tmo</code> parameter should be set to 5 seconds. If you are using zSeries devices, you must manually create and configure the <code>/etc/multipath.conf</code> file to specify the values. For information, see Section “Configuring Default Settings for zSeries in <code>/etc/multipath.conf</code> ” (page 84).
Section 7.3.1, “Device Mapper Multipath Module” (page 68)	Multipathing is supported for the <code>/boot</code> device in SUSE Linux Enterprise Server 11 and later.
Section “Configuring Default Settings for zSeries in <code>/etc/multipath.conf</code> ” (page 84)	This section is new.

Location	Change
Section 7.8, “Configuring Multi-path I/O for the Root Device” (page 100)	DM-MP is now available and supported for /boot and /root in SUSE Linux Enterprise Server 11.
