SUSE Linux Enterprise Server 12 SP5

Storage Administration Guide

Storage Administration Guide

SUSE Linux Enterprise Server 12 SP5

Provides information about how to manage storage devices on a SUSE Linux Enterprise Server.

Publication Date: November 07, 2024

https://documentation.suse.com

Contents

About This Guide

This guide provides information about how to manage storage devices on SUSE Linux Enterprise Server 12 SP5. For information about partitioning and managing devices, see *Book "Deployment Guide", Chapter 13 "Advanced Disk Setup"*. This guide is intended for system administrators.

1 Available documentation

Online documentation

Our documentation is available online at https://documentation.suse.com. Browse or download the documentation in various formats.

Note: Latest updates

The latest updates are usually available in the English-language version of this documentation.

SUSE Knowledgebase

If you have run into an issue, also check out the Technical Information Documents (TIDs) that are available online at https://www.suse.com/support/kb/. Search the SUSE Knowledgebase for known solutions driven by customer need.

Release notes

For release notes, see https://www.suse.com/releasenotes/.

In your system

For offline use, the release notes are also available under `/usr/share/doc/release-notes` on your system. The documentation for individual packages is available at `/usr/share/doc/packages`.

Many commands are also described in their *manual pages*. To view them, run `man`, followed by a specific command name. If the `man` command is not installed on your system, install it with `sudo zypper install man`.

2 Improving the documentation

Your feedback and contributions to this documentation are welcome. The following channels for giving feedback are available:

Service requests and support

For services and support options available for your product, see https://www.suse.com/support/.

To open a service request, you need a SUSE subscription registered at SUSE Customer Center. Go to https://scc.suse.com/support/requests, log in, and click *Create New*.

Bug reports

Report issues with the documentation at https://bugzilla.suse.com/.

To simplify this process, click the *Report an issue* icon next to a headline in the HTML version of this document. This preselects the right product and category in Bugzilla and adds a link to the current section. You can start typing your bug report right away.

A Bugzilla account is required.

Contributions

To contribute to this documentation, click the *Edit source document* icon next to a headline in the HTML version of this document. This will take you to the source code on GitHub, where you can open a pull request.

A GitHub account is required.

Note: *Edit source document* only available for English

The *Edit source document* icons are only available for the English version of each document. For all other languages, use the *Report an issue* icons instead.

For more information about the documentation environment used for this documentation, see the repository's README.

Mail

You can also report errors and send feedback concerning the documentation to `doc-team@suse.com`. Include the document title, the product version, and the publication date of the document. Additionally, include the relevant section number and title (or provide the URL) and provide a concise description of the problem.

3 Documentation conventions

The following notices and typographic conventions are used in this document:

- `/etc/passwd`: Directory names and file names
- *`PLACEHOLDER`*: Replace *`PLACEHOLDER`* with the actual value
- `PATH`: An environment variable
- **`ls`**, `--help`: Commands, options, and parameters
- `user`: The name of a user or group
- `package_name`: The name of a software package
- **`Alt`**, **`Alt`**–**`F1`**: A key to press or a key combination. Keys are shown in uppercase as on a keyboard.
- *File*, *File* › *Save As*: menu items, buttons
- `AMD/Intel` This paragraph is only relevant for the AMD64/Intel 64 architectures. The arrows mark the beginning and the end of the text block.
  `IBM Z, POWER` This paragraph is only relevant for the architectures `IBM Z` and `POWER`. The arrows mark the beginning and the end of the text block.
- *Chapter 1, "Example chapter"*: A cross-reference to another chapter in this guide.
- Commands that must be run with `root` privileges. You can also prefix these commands with the **`sudo`** command to run them as a non-privileged user:

```
root # command
tux > sudo command
```

- Commands that can be run by non-privileged users:

```
tux > command
```

- Commands can be split into two or multiple lines by a backslash character (`\`) at the end of a line. The backslash informs the shell that the command invocation will continue after the line's end:

```
tux > echo a b \
```

```
c d
```

- A code block that shows both the command (preceded by a prompt) and the respective output returned by the shell:

```
tux > command
output
```

- Notices

Warning: Warning notice

Vital information you must be aware of before proceeding. Warns you about security issues, potential loss of data, damage to hardware, or physical hazards.

Important: Important notice

Important information you should be aware of before proceeding.

Note: Note notice

Additional information, for example about differences in software versions.

Tip: Tip notice

Helpful information, like a guideline or a piece of practical advice.

- Compact Notices

Additional information, for example about differences in software versions.

Helpful information, like a guideline or a piece of practical advice.

4 Support

Find the support statement for SUSE Linux Enterprise Server and general information about technology previews below. For details about the product lifecycle, see https://www.suse.com/lifecycle.

If you are entitled to support, find details on how to collect information for a support ticket at https://documentation.suse.com/sles-15/html/SLES-all/cha-adm-support.html.

4.1 Support statement for SUSE Linux Enterprise Server

To receive support, you need an appropriate subscription with SUSE. To view the specific support offers available to you, go to https://www.suse.com/support/ and select your product.

The support levels are defined as follows:

L1

Problem determination, which means technical support designed to provide compatibility information, usage support, ongoing maintenance, information gathering and basic troubleshooting using available documentation.

L2

Problem isolation, which means technical support designed to analyze data, reproduce customer problems, isolate a problem area and provide a resolution for problems not resolved by Level 1 or prepare for Level 3.

L3

Problem resolution, which means technical support designed to resolve problems by engaging engineering to resolve product defects which have been identified by Level 2 Support.

For contracted customers and partners, SUSE Linux Enterprise Server is delivered with L3 support for all packages, except for the following:

- Technology previews.
- Sound, graphics, fonts, and artwork.
- Packages that require an additional customer contract.

- Some packages shipped as part of the module *Workstation Extension* are L2-supported only.
- Packages with names ending in `-devel` (containing header files and similar developer resources) will only be supported together with their main packages.

SUSE will only support the usage of original packages. That is, packages that are unchanged and not recompiled.

4.2 Technology previews

Technology previews are packages, stacks, or features delivered by SUSE to provide glimpses into upcoming innovations. Technology previews are included for your convenience to give you a chance to test new technologies within your environment. We would appreciate your feedback. If you test a technology preview, please contact your SUSE representative and let them know about your experience and use cases. Your input is helpful for future development.

Technology previews have the following limitations:

- Technology previews are still in development. Therefore, they may be functionally incomplete, unstable, or otherwise *not* suitable for production use.
- Technology previews are *not* supported.
- Technology previews may only be available for specific hardware architectures.
- Details and functionality of technology previews are subject to change. As a result, upgrading to subsequent releases of a technology preview may be impossible and require a fresh installation.
- SUSE may discover that a preview does not meet customer or market needs, or does not comply with enterprise standards. Technology previews can be removed from a product at any time. SUSE does not commit to providing a supported version of such technologies in the future.

For an overview of technology previews shipped with your product, see the release notes at https://www.suse.com/releasenotes.

I File Systems and Mounting

1 Overview of File Systems in Linux

SUSE Linux Enterprise Server ships with different file systems from which to choose, including Btrfs, Ext4, Ext3, Ext2, ReiserFS and XFS. Each file system has its own advantages and disadvantages. For a side-by-side feature comparison of the major file systems in SUSE Linux Enterprise Server, see https://www.suse.com/releasenotes/x86_64/SUSE-SLES/12-SP5/#TechInfo.Filesystems (File System Support and Sizes). This chapter contains an overview of how these file systems work and what advantages they offer.

With SUSE Linux Enterprise 12, Btrfs is the default file system for the operating system and XFS is the default for all other use cases. SUSE also continues to support the Ext family of file systems, ReiserFS and OCFS2. By default, the Btrfs file system will be set up with subvolumes. Snapshots will be automatically enabled for the root file system using the snapper infrastructure. For more information about snapper, refer to *Book "Administration Guide", Chapter 7 "System Recovery and Snapshot Management with Snapper"*.

Professional high-performance setups might require a highly available storage system. To meet the requirements of high-performance clustering scenarios, SUSE Linux Enterprise Server includes OCFS2 (Oracle Cluster File System 2) and the Distributed Replicated Block Device (DRBD) in the High Availability add-on. These advanced storage systems are not covered in this guide. For information, see the *SUSE Linux Enterprise High Availability Administration Guide* at http://www.suse.com/doc.

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.

Unless stated otherwise in this section, all the steps required to set up or change partitions and file systems can be performed by using the YaST Partitioner (which is also strongly recommended). For information, see *Book "Deployment Guide", Chapter 13 "Advanced Disk Setup"*.

1.1 Terminology

metadata

A data structure that is internal to the file system. It ensures 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 one reason 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 a variety of 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 start-up. Instead, only the journal is replayed.

1.2 Btrfs

Btrfs is a copy-on-write (COW) file system developed by Chris Mason. It is based on COW-friendly B-trees developed by Ohad Rodeh. Btrfs is a logging-style file system. Instead of journaling the block changes, it writes them in a new location, then links the change in. Until the last write, the new changes are not committed.

1.2.1 Key Features

Btrfs provides fault tolerance, repair, and easy management features, such as the following:

- Writable snapshots that allow you to easily roll back your system if needed after applying updates, or to back up files.
- Subvolume support: Btrfs creates a default subvolume in its assigned pool of space. It allows you to create additional subvolumes that act as individual file systems within the same pool of space. The number of subvolumes is limited only by the space allocated to the pool.
- The online check and repair functionality `scrub` is available as part of the Btrfs command line tools. It verifies the integrity of data and metadata, assuming the tree structure is fine. You can run scrub periodically on a mounted file system; it runs as a background process during normal operation.
- Different RAID levels for metadata and user data.
- Different checksums for metadata and user data to improve error detection.
- Integration with Linux Logical Volume Manager (LVM) storage objects.
- Integration with the YaST Partitioner and AutoYaST on SUSE Linux Enterprise Server. This also includes creating a Btrfs file system on Multiple Devices (MD) and Device Mapper (DM) storage configurations.
- Offline migration from existing Ext2, Ext3, and Ext4 file systems.
- Boot loader support for `/boot`, allowing to boot from a Btrfs partition.
- Multivolume Btrfs is supported in RAID0, RAID1, and RAID10 profiles in SUSE Linux Enterprise Server 12 SP5. Higher RAID levels are not supported yet, but might be enabled with a future service pack.
- Use Btrfs commands to set up transparent compression.

1.2.2 The Root File System Setup on SUSE Linux Enterprise Server

By default, SUSE Linux Enterprise Server is set up using Btrfs and snapshots for the root partition. Snapshots allow you to easily roll back your system if needed after applying updates, or to back up files. Snapshots can easily be managed with the SUSE Snapper infrastructure as

explained in *Book "Administration Guide", Chapter 7 "System Recovery and Snapshot Management with Snapper"*. For general information about the SUSE Snapper project, see the Snapper Portal wiki at OpenSUSE.org (http://snapper.io).

When using a snapshot to roll back the system, it must be ensured that data such as user's home directories, Web and FTP server contents or log files do not get lost or overwritten during a roll back. This is achieved by using Btrfs subvolumes on the root file system. Subvolumes can be excluded from snapshots. The default root file system setup on SUSE Linux Enterprise Server as proposed by YaST during the installation contains the following subvolumes. They are excluded from snapshots for the reasons given below.

`/boot/grub2/i386-pc`, `/boot/grub2/x86_64-efi`, `/boot/grub2/powerpc-ieee1275`, `/boot/grub2/s390x-emu`

A rollback of the boot loader configuration is not supported. The directories listed above are architecture-specific. The first two directories are present on AMD64/Intel 64 machines, the latter two on IBM POWER and on IBM IBM Z, respectively.

`/home`

If `/home` does not reside on a separate partition, it is excluded to avoid data loss on rollbacks.

`/opt`, `/var/opt`

Third-party products usually get installed to `/opt`. It is excluded to avoid uninstalling these applications on rollbacks.

`/srv`

Contains data for Web and FTP servers. It is excluded to avoid data loss on rollbacks.

`/tmp`, `/var/tmp`, `/var/cache`, `/var/crash`

All directories containing temporary files and caches are excluded from snapshots.

`/usr/local`

This directory is used when manually installing software. It is excluded to avoid uninstalling these installations on rollbacks.

`/var/lib/libvirt/images`

The default location for virtual machine images managed with libvirt. Excluded to ensure virtual machine images are not replaced with older versions during a rollback. By default, this subvolume is created with the option `no copy on write`.

`/var/lib/mailman`, `/var/spool`

Directories containing mails or mail queues are excluded to avoid a loss of mails after a rollback.

`/var/lib/named`

Contains zone data for the DNS server. Excluded from snapshots to ensure a name server can operate after a rollback.

`/var/lib/mariadb`, `/var/lib/mysql`, `/var/lib/pgqsl`

These directories contain database data. By default, these subvolumes are created with the option `no copy on write`.

`/var/log`

Log file location. Excluded from snapshots to allow log file analysis after the rollback of a broken system. By default, `/var/log` has the NoCOW attribute set, disabling copy-on-write, which improves performance and reduces the number of duplicate blocks. Verify with **`lsattr`**:

```
tux > lsattr -l /var/
/var/log      No_COW
```

Warning: Support for Rollbacks

Rollbacks are only supported by the SUSE support if you do not remove any of the pre-configured subvolumes. You may, however, add additional subvolumes using the YaST Partitioner.

1.2.2.1 Mounting Compressed Btrfs File Systems

Note: GRUB 2 and LZO Compressed Root

GRUB 2 cannot read an lzo compressed root. You need a separate `/boot` partition to use compression.

Since SLE12 SP1, compression for Btrfs file systems is supported. Use the `compress` or `compress-force` option and select the compression algorithm, `lzo` or `zlib` (the default). The zlib compression has a higher compression ratio while lzo is faster and takes less CPU load.

For example:

```
root # mount -o compress /dev/sdx /mnt
```

In case you create a file, write to it, and the compressed result is greater or equal to the uncompressed size, Btrfs will skip compression for future write operations forever for this file. If you do not like this behavior, use the `compress-force` option. This can be useful for files that have some initial non-compressable data.

Note, compression takes effect for new files only. Files that were written without compression are not compressed when the file system is mounted with the `compress` or `compress-force` option. Furthermore, files with the `nodatacow` attribute never get their extents compressed:

```
root # chattr +C FILE
root # mount -o nodatacow  /dev/sdx /mnt
```

In regard to encryption, this is independent from any compression. After you have written some data to this partition, print the details:

```
root # btrfs filesystem show /mnt
btrfs filesystem show /mnt
Label: 'Test-Btrfs'  uuid: 62f0c378-e93e-4aa1-9532-93c6b780749d
        Total devices 1 FS bytes used 3.22MiB
      devid    1 size 2.00GiB used 240.62MiB path /dev/sdb1
```

If you want this to be permanent, add the `compress` or `compress-force` option into the `/etc/fstab` configuration file. For example:

```
UUID=1a2b3c4d /home btrfs subvol=@/home,compress 0 0
```

1.2.2.2 Mounting Subvolumes

A system rollback from a snapshot on SUSE Linux Enterprise Server is performed by booting from the snapshot first. This allows you to check the snapshot while running before doing the rollback. Being able to boot from snapshots is achieved by mounting the subvolumes (which would normally not be necessary).

In addition to the subvolumes listed in *Section 1.2.2, "The Root File System Setup on SUSE Linux Enterprise Server"* a volume named `@` exists. This is the default subvolume that will be mounted as the root partition (`/`). The other subvolumes will be mounted into this volume.

When booting from a snapshot, not the `@` subvolume will be used, but rather the snapshot. The parts of the file system included in the snapshot will be mounted read-only as `/`. The other subvolumes will be mounted writable into the snapshot. This state is temporary by default: the

previous configuration will be restored with the next reboot. To make it permanent, execute the `snapper rollback` command. This will make the snapshot that is currently booted the new *default* subvolume, which will be used after a reboot.

1.2.2.3 Checking for Free Space

File system usage is usually checked by running the `df` command. On a Btrfs file system, the output of `df` can be misleading, because in addition to the space the raw data allocates, a Btrfs file system also allocates and uses space for metadata.

Consequently a Btrfs file system may report being out of space even though it seems that plenty of space is still available. In that case, all space allocated for the metadata is used up. Use the following commands to check for used and available space on a Btrfs file system:

`btrfs filesystem show`

```
tux > sudo btrfs filesystem show /
Label: 'ROOT'  uuid: 52011c5e-5711-42d8-8c50-718a005ec4b3
        Total devices 1 FS bytes used 10.02GiB
        devid    1 size 20.02GiB used 13.78GiB path /dev/sda3
```

Shows the total size of the file system and its usage. If these two values in the last line match, all space on the file system has been allocated.

`btrfs filesystem df`

```
tux > sudo btrfs filesystem df /
Data, single: total=13.00GiB, used=9.61GiB
System, single: total=32.00MiB, used=16.00KiB
Metadata, single: total=768.00MiB, used=421.36MiB
GlobalReserve, single: total=144.00MiB, used=0.00B
```

Shows values for allocated (`total`) and used space of the file system. If the values for `total` and `used` for the metadata are almost equal, all space for metadata has been allocated.

`btrfs filesystem usage`

```
tux > sudo btrfs filesystem usage /
Overall:
    Device size:                  20.02GiB
    Device allocated:             13.78GiB
    Device unallocated:            6.24GiB
    Device missing:                  0.00B
    Used:                         10.02GiB
```

```
    Free (estimated):              9.63GiB      (min: 9.63GiB)
    Data ratio:                       1.00
    Metadata ratio:                   1.00
    Global reserve:              144.00MiB      (used: 0.00B)

             Data     Metadata  System
Id Path      single   single    single   Unallocated
-- --------- -------- --------- -------- -----------
 1 /dev/sda3 13.00GiB 768.00MiB 32.00MiB     6.24GiB
-- --------- -------- --------- -------- -----------
   Total     13.00GiB 768.00MiB 32.00MiB     6.24GiB
   Used       9.61GiB 421.36MiB 16.00KiB
```

Shows data similar to that of the two previous commands combined.

For more information refer to **`man 8 btrfs-filesystem`** and https://btrfs.wiki.kernel.org/index.php/FAQ.

1.2.3 Migration from Ext and ReiserFS File Systems to Btrfs

You can migrate data volumes from existing Ext (Ext2, Ext3, or Ext4) or ReiserFS to the Btrfs file system. The conversion process occurs offline and in place on the device. The file system needs at least 15% of available free space on the device.

To convert the file system to Btrfs, take the file system offline, then enter:

```
sudo btrfs-convert DEVICE
```

To roll back the migration to the original file system, take the file system offline, then enter:

```
sudo btrfs-convert -r DEVICE
```

Warning: Root File System Conversion not Supported

Converting the root file system to Btrfs is not supported. Either keep the existing file system or re-install the whole system from scratch.

Important: Possible Loss of Data

When rolling back to the original file system, all data will be lost that you added after the conversion to Btrfs. That is, only the original data is converted back to the previous file system.

1.2.4 Btrfs Administration

Btrfs is integrated in the YaST Partitioner and AutoYaST. It is available during the installation to allow you to set up a solution for the root file system. You can use the YaST Partitioner after the installation to view and manage Btrfs volumes.

Btrfs administration tools are provided in the `btrfsprogs` package. For information about using Btrfs commands, see the **`man 8 btrfs`**, **`man 8 btrfsck`**, and **`man 8 mkfs.btrfs`** commands. For information about Btrfs features, see the *Btrfs wiki* at http://btrfs.wiki.kernel.org.

1.2.5 Btrfs Quota Support for Subvolumes

The Btrfs root file system subvolumes `/var/log`, `/var/crash` and `/var/cache` can use all of the available disk space during normal operation, and cause a system malfunction. To help avoid this situation, SUSE Linux Enterprise Server now offers Btrfs quota support for subvolumes. If you set up the root file system by using the respective YaST proposal, it is prepared accordingly: quota groups (`qgroup`) for all subvolumes are already set up. To set a quota for a subvolume in the root file system, proceed as follows:

Note: Btrfs Quota Groups Can Incur Degraded Performance

On SUSE Linux Enterprise Server 12 SP5, using Btrfs quota groups can degrade file system performance.

1. Enable quota support:

   ```
   sudo btrfs quota enable /
   ```

2. Get a list of subvolumes:

   ```
   sudo btrfs subvolume list /
   ```

   Quotas can only be set for existing subvolumes.

3. Set a quota for one of the subvolumes that was listed in the previous step. A subvolume can either be identified by path (for example `/var/tmp`) or by `0/SUBVOLUME ID` (for example `0/272`). The following example sets a quota of five GB for `/var/tmp`.

   ```
   sudo btrfs qgroup limit 5G /var/tmp
   ```

The size can either be specified in bytes (5000000000), kilobytes (5000000K), megabytes (5000M), or gigabytes (5G). The resulting values in bytes slightly differ, since 1024 Bytes = 1 KiB, 1024 KiB = 1 MiB, etc.

4. To list the existing quotas, use the following command. The column `max_rfer` shows the quota in bytes.

```
sudo btrfs qgroup show -r /
```

Tip: Nullifying a Quota

In case you want to nullify an existing quota, set a quota size of `none`:

```
sudo btrfs qgroup limit none /var/tmp
```

To disable quota support for a partition and all its subvolumes, use **`btrfs quota disable`**:

```
sudo btrfs quota disable /
```

See the **`man 8 btrfs-qgroup`** and **`man 8 btrfs-quota`** for more details. The *UseCases* page on the Btrfs wiki (https://btrfs.wiki.kernel.org/index.php/UseCases) also provides more information.

1.2.6 Btrfs send/receive

Btrfs allows to make snapshots to capture the state of the file system. Snapper, for example, uses this feature to create snapshots before and after system changes, allowing a rollback. However, together with the send/receive feature, snapshots can also be used to create and maintain copies of a file system in a remote location. This feature can, for example, be used to do incremental backups.

A **`btrfs send`** operation calculates the difference between two read-only snapshots from the same subvolume and sends it to a file or to STDOUT. A **`Btrfs receive`** operation takes the result of the send command and applies it to a snapshot.

1.2.6.1 Prerequisites

To use Btrfs's send/receive feature, the following requirements need to be met:

- A Btrfs file system is required on the source side (`send`) and on the target side (`receive`).
- Btrfs send/receive operates on snapshots, therefore the respective data needs to reside in a Btrfs subvolume.
- Snapshots on the source side need to be read-only.
- SUSE Linux Enterprise 12 SP2 or better. Earlier versions of SUSE Linux Enterprise do not support send/receive.

1.2.6.2 Incremental Backups

The following procedure shows the basic usage of Btrfs send/receive using the example of creating incremental backups of `/data` (source side) in `/backup/data` (target side). `/data` needs to be a subvolume.

PROCEDURE 1.1: INITIAL SETUP

1. Create the initial snapshot (called `snapshot_0` in this example) on the source side and make sure it is written to the disk:

   ```
   sudo btrfs subvolume snapshot -r /data /data/bkp_data
   sync
   ```

   A new subvolume `/data/bkp_data` is created. It will be used as the basis for the next incremental backup and should be kept as a reference.

2. Send the initial snapshot to the target side. Since this is the initial send/receive operation, the complete snapshot needs to be sent:

   ```
   sudo bash -c 'btrfs send /data/bkp_data | btrfs receive /backup'
   ```

   A new subvolume `/backup/bkp_data` is created on the target side.

When the initial setup has been finished, you can create incremental backups and send the differences between the current and previous snapshots to the target side. The procedure is always the same:

1. Create a new snapshot on the source side.
2. Send the differences to the target side.
3. Optional: Rename and/or clean up snapshots on both sides.

PROCEDURE 1.2: PERFORMING AN INCREMENTAL BACKUP

1. Create a new snapshot on the source side and make sure it is written to the disk. In the following example the snapshot is named bkp_data_*CURRENT_DATE*:

   ```
   sudo btrfs subvolume snapshot -r /data /data/bkp_data_$(date +%F)
   sync
   ```

   A new subvolume, for example `/data/bkp_data_2016-07-07`, is created.

2. Send the difference between the previous snapshot and the one you have created to the target side. This is achieved by specifying the previous snapshot with the option `-p *SNAPSHOT*`.

   ```
   sudo bash -c 'btrfs send -p /data/bkp_data /data/bkp_data_2016-07-07 \
   | btrfs receive /backup'
   ```

   A new subvolume `/backup/bkp_data_2016-07-07` is created.

3. As a result four snapshots, two on each side, exist:

   `/data/bkp_data`
   `/data/bkp_data_2016-07-07`
   `/backup/bkp_data`
   `/backup/bkp_data_2016-07-07`

Now you have three options for how to proceed:

- Keep all snapshots on both sides. With this option you can roll back to any snapshot on both sides while having all data duplicated at the same time. No further action is required. When doing the next incremental backup, keep in mind to use the next-to-last snapshot as parent for the send operation.
- Only keep the last snapshot on the source side and all snapshots on the target side. Also allows to roll back to any snapshot on both sides—to do a rollback to a specific snapshot on the source side, perform a send/receive operation of a complete snapshot from the target side to the source side. Do a delete/move operation on the source side.
- Only keep the last snapshot on both sides. This way you have a backup on the target side that represents the state of the last snapshot made on the source side. It is not possible to roll back to other snapshots. Do a delete/move operation on the source and the target side.

a. To only keep the last snapshot on the source side, perform the following commands:

```
sudo btrfs subvolume delete /data/bkp_data
sudo mv /data/bkp_data_2016-07-07 /data/bkp_data
```

The first command will delete the previous snapshot, the second command renames the current snapshot to `/data/bkp_data`. This ensures that the last snapshot that was backed up is always named `/data/bkp_data`. As a consequence, you can also always use this subvolume name as a parent for the incremental send operation.

b. To only keep the last snapshot on the target side, perform the following commands:

```
sudo btrfs subvolume delete /backup/bkp_data
sudo mv /backup/bkp_data_2016-07-07 /backup/bkp_data
```

The first command will delete the previous backup snapshot, the second command renames the current backup snapshot to `/backup/bkp_data`. This ensures that the latest backup snapshot is always named `/backup/bkp_data`.

Tip: Sending to a Remote Target Side

To send the snapshots to a remote machine, use SSH:

```
btrfs send /data/bkp_data | ssh root@jupiter.example.com 'btrfs receive /backup'
```

1.2.7 Data Deduplication Support

Btrfs supports data deduplication by replacing identical blocks in the file system with logical links to a single copy of the block in a common storage location. SUSE Linux Enterprise Server provides the tool **`duperemove`** for scanning the file system for identical blocks. When used on a Btrfs file system, it can also be used to deduplicate these blocks. **`duperemove`** is not installed by default. To make it available, install the package `duperemove` .

Note: Use Cases

As of SUSE Linux Enterprise Server 12 SP5 **`duperemove`** is not suited to deduplicate the entire file system. It is intended to be used to deduplicate a set of 10 to 50 large files that possibly have lots of blocks in common, such as virtual machine images.

`duperemove` can either operate on a list of files or recursively scan a directory:

```
sudo duperemove OPTIONS file1 file2 file3
sudo duperemove -r OPTIONS directory
```

It operates in two modes: read-only and de-duping. When run in read-only mode (that is without the `-d` switch), it scans the given files or directories for duplicated blocks and prints them. This works on any file system.

Running **`duperemove`** in de-duping mode is only supported on Btrfs file systems. After having scanned the given files or directories, the duplicated blocks will be submitted for deduplication.

For more information see **`man 8 duperemove`**.

1.2.8 Deleting Subvolumes from the Root File System

You may need to delete one of the default Btrfs subvolumes from the root file system for specific purposes. One of them is transforming a subvolume—for example `@/home` or `@/srv`—into a file system on a separate device. The following procedure illustrates how to delete a Btrfs subvolume:

1. Identify the subvolume you need to delete (for example `@/opt`). Notice that the root path has always subvolume ID '5'.

```
tux > sudo btrfs subvolume list /
ID 256 gen 30 top level 5 path @
ID 258 gen 887 top level 256 path @/var
ID 259 gen 872 top level 256 path @/usr/local
ID 260 gen 886 top level 256 path @/tmp
ID 261 gen 60 top level 256 path @/srv
ID 262 gen 886 top level 256 path @/root
ID 263 gen 39 top level 256 path @/opt
[...]
```

2. Find the device name that hosts the root partition:

```
tux > sudo btrfs device usage /
/dev/sda1, ID: 1
  Device size:            23.00GiB
  Device slack:              0.00B
  Data,single:             7.01GiB
  Metadata,DUP:            1.00GiB
  System,DUP:             16.00MiB
  Unallocated:            14.98GiB
```

3. Mount the root file system (subvolume with ID 5) on a separate mount point (for example `/mnt`):

```
tux > sudo mount -o subvolid=5 /dev/sda1 /mnt
```

4. Delete the `@/opt` partition from the mounted root file system:

```
tux > sudo btrfs subvolume delete /mnt/@/opt
```

5. Unmount the previously mounted root file system:

```
tux > sudo umount /mnt
```

1.3 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. XFS is the default file system for data partitions in SUSE Linux Enterprise Server.

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.

High scalability

XFS offers high scalability by using 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 called *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

XFS offers 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.

1.3.1 XFS formats

SUSE Linux Enterprise Server supports the “on-disk format” (v5) of the XFS file system. The main advantages of this format are automatic checksums of all XFS metadata, file type support, and support for a larger number of access control lists for a file.

Note that this format is not supported by SUSE Linux Enterprise kernels older than version 3.12, by `xfsprogs` older than version 3.2.0, and GRUB 2 versions released before SUSE Linux Enterprise 12.

Important: The V4 is deprecated

XFS is deprecating file systems with the V4 format. This file system format was created by the command:

```
mkfs.xfs -m crc=0 DEVICE
```

The format was used in SLE 11 and older releases and currently it creates a warning message by **`dmesg`**:

```
Deprecated V4 format (crc=0) will not be supported after September 2030
```

If you see the message above in the output of the **`dmesg`** command, it is recommended that you update your file system to the V5 format:

1. Back up your data to another device.
2. Create the file system on the device.

   ```
   mkfs.xfs -m crc=1 DEVICE
   ```

3. Restore the data from the backup on the updated device.

1.4 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 several 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.

Solidity and Speed

Being an "old-timer", Ext2 underwent many improvements and was heavily tested. This might be the reason 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 only 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 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 and its metadata format, upgrades from Ext2 to Ext3 are very easy.

1.5 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:

1.5.1 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 re-creating 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. Perform a clean unmount of the Ext3 file system and remount it as an Ext2 file system.

1.5.2 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 to 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.

1.5.3 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, add the modules `ext3` and `jbd` in the `initrd`. Do so by

   a. opening or creating `/etc/dracut.conf.d/10-filesystem.conf` and adding the following line (mind the leading whitespace):

   ```
   force_drivers+=" ext3 jbd"
   ```

   b. and running the **`dracut -f`** command.

4. Reboot the system.

1.5.4 Ext3 File System Inode Size and Number of Inodes

An inode stores information about the file and its block location in the file system. To allow space in the inode for extended attributes and ACLs, the default inode size for Ext3 was increased from 128 bytes on SLES 10 to 256 bytes on SLES 11. As compared to SLES 10, when you make a new Ext3 file system on SLES 11, the default amount of space preallocated for the same number

of inodes is doubled, and the usable space for files in the file system is reduced by that amount. Thus, you must use larger partitions to accommodate the same number of inodes and files than were possible for an Ext3 file system on SLES 10.

When you create a new Ext3 file system, the space in the inode table is preallocated for the total number of inodes that can be created. The bytes-per-inode ratio and the size of the file system determine how many inodes are possible. When the file system is made, an inode is created for every bytes-per-inode bytes of space:

```
number of inodes = total size of the file system divided by the number of bytes per inode
```

The number of inodes controls the number of files you can have in the file system: one inode for each file. To address the increased inode size and reduced usable space available, the default for the bytes-per-inode ratio was increased from 8192 bytes on SLES 10 to 16384 bytes on SLES 11. The doubled ratio means that the number of files that can be created is one-half of the number of files possible for an Ext3 file system on SLES 10.

Important: Changing the Inode Size of an Existing Ext3 File System

After the inodes are allocated, you cannot change the settings for the inode size or bytes-per-inode ratio. No new inodes are possible without re-creating the file system with different settings, or unless the file system gets extended. When you exceed the maximum number of inodes, no new files can be created on the file system until some files are deleted.

When you make a new Ext3 file system, you can specify the inode size and bytes-per-inode ratio to control inode space usage and the number of files possible on the file system. If the blocks size, inode size, and bytes-per-inode ratio values are not specified, the default values in the `/etc/mked2fs.conf` file are applied. For information, see the `mke2fs.conf(5)` man page.

Use the following guidelines:

- **Inode size:** The default inode size is 256 bytes. Specify a value in bytes that is a power of 2 and equal to 128 or larger in bytes and up to the block size, such as 128, 256, 512, and so on. Use 128 bytes only if you do not use extended attributes or ACLs on your Ext3 file systems.

- **Bytes-per-inode ratio:** The default bytes-per-inode ratio is 16384 bytes. Valid bytes-per-inode ratio values must be a power of 2 equal to 1024 or greater in bytes, such as 1024, 2048, 4096, 8192, 16384, 32768, and so on. This value should not be smaller than the block size of the file system, because the block size is the smallest chunk of space used to store data. The default block size for the Ext3 file system is 4 KB.
  In addition, you should consider the number of files and the size of files you need to store. For example, if your file system will have many small files, you can specify a smaller bytes-per-inode ratio, which increases the number of inodes. If your file system will have very large files, you can specify a larger bytes-per-inode ratio, which reduces the number of possible inodes.
  Generally, it is better to have too many inodes than to run out of them. If you have too few inodes and very small files, you could reach the maximum number of files on a disk that is practically empty. If you have too many inodes and very large files, you might have free space reported but be unable to use it because you cannot create new files in space reserved for inodes.

If you do not use extended attributes or ACLs on your Ext3 file systems, you can restore the SLES 10 behavior specifying 128 bytes as the inode size and 8192 bytes as the bytes-per-inode ratio when you make the file system. Use any of the following methods to set the inode size and bytes-per-inode ratio:

- **Modifying the default settings for all new Ext3 files:** In a text editor, modify the `defaults` section of the `/etc/mke2fs.conf` file to set the `inode_size` and `inode_ratio` to the desired default values. The values apply to all new Ext3 file systems. For example:

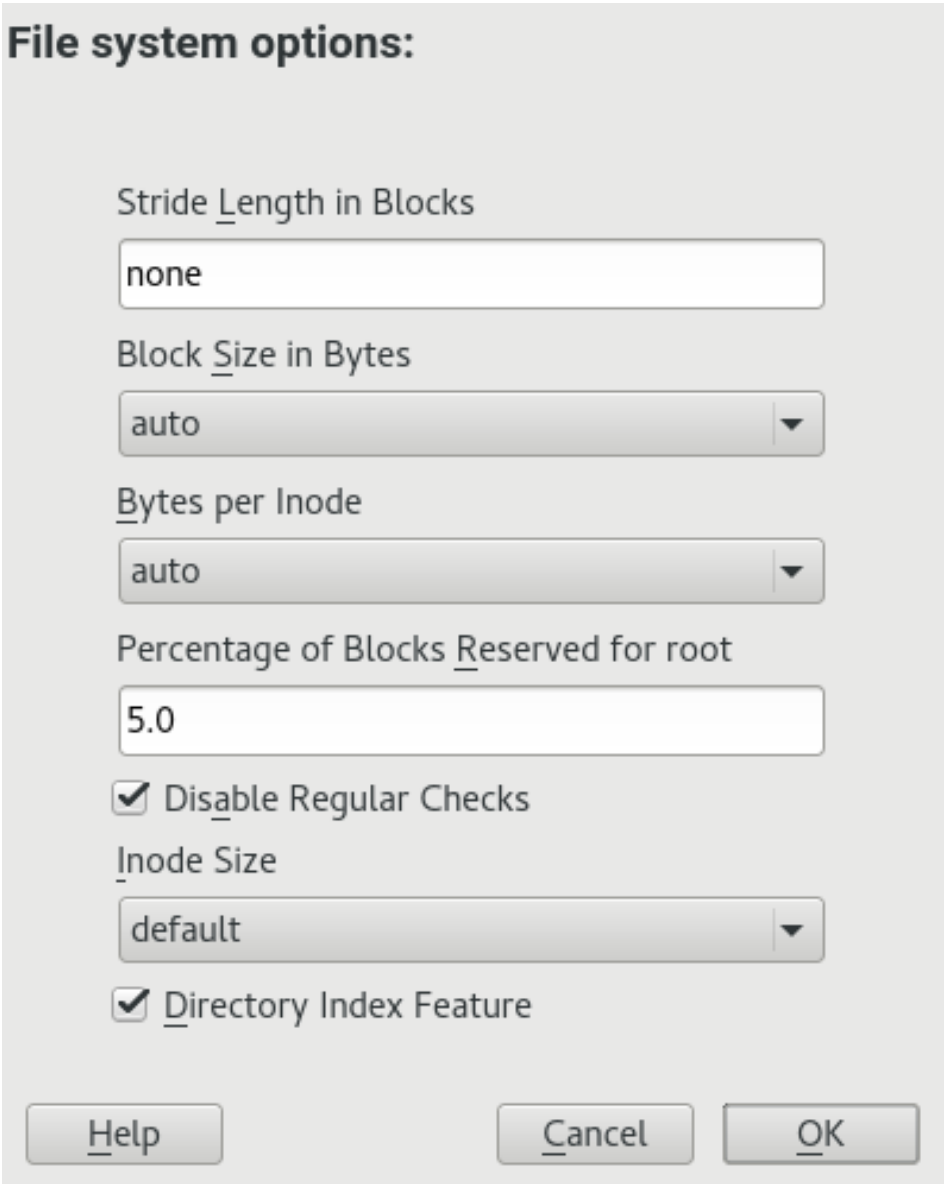
  ```
  blocksize = 4096
  inode_size = 128
  ```

```
inode_ratio = 8192
```

- **At the command line:** Pass the inode size (`-I 128`) and the bytes-per-inode ratio (`-i 8192`) to the **`mkfs.ext3(8)`** command or the **`mke2fs(8)`** command when you create a new Ext3 file system. For example, use either of the following commands:

```
sudo mkfs.ext3 -b 4096 -i 8092 -I 128 /dev/sda2
sudo mke2fs -t ext3 -b 4096 -i 8192 -I 128 /dev/sda2
```

- **During installation with YaST:** Pass the inode size and bytes-per-inode ratio values when you create a new Ext3 file system during the installation. In the YaST Partitioner on the *Edit Partition* page under *Formatting Options*, select *Format partitionExt3*, then click *Options*. In the *File system options* dialog, select the desired values from the *Block Size in Bytes*, *Bytes-per-inode*, and *Inode Size* drop-down box.
  For example, select 4096 for the *Block Size in Bytes* drop-down box, select 8192 from the *Bytes per inode* drop-down box, select 128 from the *Inode Size* drop-down box, then click *OK*.



- **During installation with AutoYaST:** In an AutoYaST profile, you can use the `fs_options` tag to set the `opt_bytes_per_inode` ratio value of 8192 for -i and the `opt_inode_density` value of 128 for -I:

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <initialize config:type="boolean">true</initialize>
    <partitions config:type="list">
      <partition>
        <filesystem config:type="symbol">ext3</filesystem>
        <format config:type="boolean">true</format>
```

```
        <fs_options>
          <opt_bytes_per_inode>
            <option_str>-i</option_str>
            <option_value>8192</option_value>
          </opt_bytes_per_inode>
          <opt_inode_density>
            <option_str>-I</option_str>
            <option_value>128</option_value>
          </opt_inode_density>
        </fs_options>
        <mount>/</mount>
        <partition_id config:type="integer">131</partition_id>
        <partition_type>primary</partition_type>
        <size>25G</size>
      </partition>
    </partitions>
  </drive>
<partitioning>
```

For information, see https://www.suse.com/support/kb/doc.php?id=7009075 (*SLES11 ext3 partitions can only store 50% of the files that can be stored on SLES10* [Technical Information Document 7009075]).

1.6 Ext4

In 2006, Ext4 started as a fork from Ext3. It eliminates some storage limitations of Ext3 by supporting volumes with a size of up to 1 exbibyte, files with a size of up to 16 tebibytes and an unlimited number of subdirectories. It also introduces several performance enhancements such as delayed block allocation and a much faster file system checking routine. Ext4 is also more reliable by supporting journal checksums and by providing time stamps measured in nanoseconds. Ext4 is fully backward compatible to Ext2 and Ext3—both file systems can be mounted as Ext4.

1.7 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 usage, better disk access performance, faster crash recovery, and reliability through data journaling.

Important: Support of ReiserFS in SUSE Linux Enterprise Server 12

Existing ReiserFS partitions are supported for the lifetime of SUSE Linux Enterprise Server 12 specifically for migration purposes. Support for creating new ReiserFS file systems has been removed starting with SUSE Linux Enterprise Server 12.

1.8 Other Supported File Systems

Table 1.1, "File System Types in Linux" 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
`cramfs`	Compressed ROM file system: A compressed read-only file system for ROMs.
`hpfs`	High Performance File System: The IBM OS/2 standard file system. Only supported in read-only mode.
`iso9660`	Standard file system on CD-ROMs.
`minix`	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.
`msdos`	`fat`, the file system originally used by DOS, is today used by various operating systems.
`nfs`	Network File System: Here, data can be stored on any machine in a network and access might be granted via a network.
`ntfs`	Windows NT file system; read-only.
`smbfs`	Server Message Block is used by products such as Windows to enable file access over a network.

File System Type	Description
`sysv`	Used on SCO Unix, Xenix, and Coherent (commercial Unix systems for PCs).
`ufs`	Used by BSD, SunOS, and NextStep. Only supported in read-only mode.
`umsdos`	Unix on MS-DOS: Applied on top of a standard `fat` file system, achieves Unix functionality (permissions, links, long file names) by creating special files.
`vfat`	Virtual FAT: Extension of the `fat` file system (supports long file names).

1.9 Large File Support in Linux

Originally, Linux supported a maximum file size of 2 GiB (2^{31} bytes). Unless a file system comes with large file support, the maximum file size on a 32-bit system is 2 GiB.

Currently, all our standard file systems have LFS (large file support), which gives a maximum file size of 2^{63} bytes in theory. *Table 1.2, "Maximum Sizes of Files and File Systems (On-Disk Format, 4 KiB Block Size)"* offers an overview of the current on-disk format limitations of Linux files and file systems. The numbers in the table assume that the file systems are using 4 KiB block size, which is a common standard. When using different block sizes, the results are different. The maximum file sizes in *Table 1.2, "Maximum Sizes of Files and File Systems (On-Disk Format, 4 KiB Block Size)"* can be larger than the file system's actual size when using sparse blocks.

Note: Binary Multiples

In this document: 1024 Bytes = 1 KiB; 1024 KiB = 1 MiB; 1024 MiB = 1 GiB; 1024 GiB = 1 TiB; 1024 TiB = 1 PiB; 1024 PiB = 1 EiB (see also *NIST: Prefixes for Binary Multiples* (http://physics.nist.gov/cuu/Units/binary.html).

TABLE 1.2: MAXIMUM SIZES OF FILES AND FILE SYSTEMS (ON-DISK FORMAT, 4 KIB BLOCK SIZE)

File System (4 KiB Block Size)	Maximum File System Size	Maximum File Size
Btrfs	16 EiB	16 EiB
Ext3	16 TiB	2 TiB
Ext4	1 EiB	16 TiB
OCFS2 (a cluster-aware file system available in the High Availability Extension)	16 TiB	1 EiB
ReiserFS v3.6	16 TiB	1 EiB
XFS	8 EiB	8 EiB
NFSv2 (client side)	8 EiB	2 GiB
NFSv3/NFSv4 (client side)	8 EiB	8 EiB

Important: Limitations

Table 1.2, "Maximum Sizes of Files and File Systems (On-Disk Format, 4 KiB Block Size)" describes the limitations regarding the on-disk format. The 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 TiB (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.

1.10 Linux Kernel Storage Limitations

Table 1.3, "Storage Limitations" summarizes the kernel limits for storage associated with SUSE Linux Enterprise Server.

TABLE 1.3: STORAGE LIMITATIONS

Storage Feature	Limitation
Maximum number of LUNs supported	16384 LUNs per target.
Maximum number of paths per single LUN	No limit by default. Each path is treated as a normal LUN. The actual limit is given by the number of LUNs per target and the number of targets per HBA (16777215 for a Fibre Channel HBA).
Maximum number of HBAs	Unlimited. The actual limit is determined by the amount of PCI slots of the system.
Maximum number of paths with device-mapper-multipath (in total) per operating system	Approximately 1024. The actual number depends on the length of the device number strings for each multipath device. It is a compile-time variable within multipath-tools, which can be raised if this limit poses a problem.
Maximum size per block device	Up to 8 EiB.

1.11 Troubleshooting File Systems

This section describes some known issues and possible solutions for file systems.

1.11.1 Btrfs Error: No space is left on device

The root (`/`) partition using the Btrfs file system stops accepting data. You receive the error "`No space left on device`".

See the following sections for information about possible causes and prevention of this issue.

1.11.1.1 Disk Space Consumed by Snapper Snapshots

If Snapper is running for the Btrfs file system, the "`No space left on device`" problem is typically caused by having too much data stored as snapshots on your system.

You can remove some snapshots from Snapper, however, the snapshots are not deleted immediately and might not free up as much space as you need.

To delete files from Snapper:

1. Open a terminal.
2. At the command prompt, enter **`btrfs filesystem show`**, for example:

   ```
   tux > sudo btrfs filesystem show
   Label: none uuid: 40123456-cb2c-4678-8b3d-d014d1c78c78
    Total devices 1 FS bytes used 20.00GB
    devid 1 size 20.00GB used 20.00GB path /dev/sda3
   ```

3. Enter

   ```
   sudo btrfs fi balance start MOUNTPOINT -dusage=5
   ```

   This command attempts to relocate data in empty or near-empty data chunks, allowing the space to be reclaimed and reassigned to metadata. This can take a while (many hours for 1 TB) although the system is otherwise usable during this time.

4. List the snapshots in Snapper. Enter

   ```
   sudo snapper -c root list
   ```

5. Delete one or more snapshots from Snapper. Enter

   ```
   sudo snapper -c root delete SNAPSHOT_NUMBER(S)
   ```

   Ensure that you delete the oldest snapshots first. The older a snapshot is, the more disk space it occupies.

To help prevent this problem, you can change the Snapper cleanup algorithms. See *Book "Administration Guide", Chapter 7 "System Recovery and Snapshot Management with Snapper", Section 7.6.1.2 "Cleanup Algorithms"* for details. The configuration values controlling snapshot cleanup are `EMPTY_*`, `NUMBER_*`, and `TIMELINE_*`.

If you use Snapper with Btrfs on the file system disk, it is advisable to reserve twice the amount of disk space than the standard storage proposal. The YaST Partitioner automatically proposes twice the standard disk space in the Btrfs storage proposal for the root file system.

1.11.1.2 Disk Space Consumed by Log, Crash, and Cache Files

If the system disk is filling up with data, you can try deleting files from `/var/log`, `/var/crash`, `/var/lib/systemd/coredump` and `/var/cache`.

The Btrfs `root` file system subvolumes `/var/log`, `/var/crash` and `/var/cache` can use all of the available disk space during normal operation, and cause a system malfunction. To help avoid this situation, SUSE Linux Enterprise Server offers Btrfs quota support for subvolumes. See *Section 1.2.5, "Btrfs Quota Support for Subvolumes"* for details.

On test and development machines, especially if you have frequent crashes of applications, you may also want to have a look at `/var/lib/systemd/coredump` where the coredumps are stored.

1.11.2 Freeing Unused File System Blocks

On solid-state drives (SSDs) and thinly provisioned volumes it is useful to trim blocks not in use by the file system. SUSE Linux Enterprise Server fully supports `unmap` or `trim` operations on all file systems supporting these methods.

The recommended way to trim a supported file system (except Btrfs) on SUSE Linux Enterprise Server is to run `/sbin/wiper.sh`. Make sure to read `/usr/share/doc/packages/hdparm/README.wiper` before running this script. For most desktop and server systems the sufficient trimming frequency is once a week. Mounting a file system with `-o discard` comes with a performance penalty and may negatively affect the lifetime of SSDs and is not recommended.

Warning: Do Not Use `wiper.sh` on Btrfs

The `wiper.sh` script trims read-write mounted ext4 or XFS file systems and read-only mounted/unmounted ext2, ext3, ext4, or XFS file systems. Do *not* use `wiper.sh` on the Btrfs file system as it may damage your data. Instead, use `/usr/share/btrfsmaintenance/btrfs-trim.sh` which is part of the `btrfsmaintenance` package.

1.11.3 Btrfs: Balancing Data across Devices

The **`btrfs balance`** command is part of the `btrfs-progs` package. It balances block groups on Btrfs file systems in the following example situations:

- Assume you have 1 TB drive with 600 GB used by data and you add another 1 TB drive. The balancing will theoretically result in having 300 GB used space on each drive.
- You have a lot of near-empty chunks on a device. Their space will not be available until the balancing has cleared those chunks.
- You need to compact half-empty block group based on the percentage of their usage. The following command will balance block groups whose usage is 5 % or less:

```
tux > sudo btrfs balance start -dusage=5 /
```

Tip

The `/etc/cron.weekly/btrfs-balance` script takes care of cleaning up unused block groups on weekly basis.

- You need to clear out non-full portions of block devices and spread data more evenly.
- You need to migrate data between different RAID types. For example, to convert data on a set of disks from RAID1 to RAID5, run the following command:

```
tux > sudo btrfs balance start -dprofiles=raid1,convert=raid5 /
```

Tip

To fine-tune the default behavior of balancing data on Btrfs file systems—for example, how frequently or which mount points to balance— inspect and customize `/etc/sysconfig/btrfsmaintenance`. The relevant options start with `BTRFS_BALANCE_`.

For details about the **`btrfs balance`** command usage, see its manual pages (**`man 8 btrfs-balance`**).

1.11.4 No Defragmentation on SSDs

Linux file systems contain mechanisms to avoid data fragmentation and usually it is not necessary to defragment. However, there are use cases, where data fragmentation cannot be avoided and where defragmenting the hard disk significantly improves the performance.

This only applies to conventional hard disks. On solid state disks (SSDs) which use flash memory to store data, the firmware provides an algorithm that determines to which chips the data is written. Data is usually spread all over the device. Therefore defragmenting an SSD does not have the desired effect and will reduce the lifespan of an SSD by writing unnecessary data.

For the reasons mentioned above, SUSE explicitly recommends *not* to defragment SSDs. Some vendors also warn about defragmenting solid state disks. This includes, but it is not limited to the following:

- HPE 3PAR StoreServ All-Flash
- HPE 3PAR StoreServ Converged Flash

1.12 Additional Information

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

- The Btrfs Wiki on Kernel.org: https://btrfs.wiki.kernel.org/
- E2fsprogs: Ext2/3/4 File System Utilities: http://e2fsprogs.sourceforge.net/
- Introducing Ext3: http://www.ibm.com/developerworks/linux/library/l-fs7/

A comprehensive multi-part tutorial about Linux file systems can be found at IBM developerWorks in the *Advanced File System Implementor's Guide* (https://www.ibm.com/developerworks/linux/library/l-fs/).

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).

2 Resizing File Systems

Resizing file systems—not to be confused with resizing partitions or volumes—can be used to make space available on physical volumes or to use additional space available on a physical volume.

2.1 Use Cases

It is strongly recommended to use the YaST Partitioner to resize partitions or logical volumes. When doing so, the file system will automatically be adjusted to the new size of the partition or volume. However, there are some cases where you need to resize the file system manually, because they are not supported by YaST:

- After having resized a virtual disk of a VM Guest.
- After having resized a volume from a network-attached storage.
- After having manually resized partitions (for example by using `fdisk` or `parted`) or logical volumes (for example by using `lvresize`).
- When wanting to shrink Btrfs file systems (as of SUSE Linux Enterprise Server 12, YaST only supports growing Btrfs file systems).

2.2 Guidelines for Resizing

Resizing any file system involves some risks that can potentially result in losing data.

Warning: Back Up your Data

To avoid data loss, ensure that you back up your data before you begin any resizing task.

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

2.2.1 File Systems that Support Resizing

The file system must support resizing to take advantage of increases in available space for the volume. In SUSE Linux Enterprise Server, file system resizing utilities are available for file systems Ext2, Ext3, Ext4, and ReiserFS. The utilities support increasing and decreasing the size as follows:

TABLE 2.1: FILE SYSTEM SUPPORT FOR RESIZING

File System	Utility	Increase Size (Grow)	Decrease Size (Shrink)
Btrfs	`btrfs filesystem resize`	Online	Online
XFS	`xfs_growfs`	Online	Not supported
Ext2	`resize2fs`	Online or offline	Offline only
Ext3	`resize2fs`	Online or offline	Offline only
Ext4	`resize2fs`	Online or offline	Offline only
ReiserFS	`resize_reiserfs`	Online or offline	Offline only

2.2.2 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. Ensure that you 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, ensure that 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.

2.2.3 Decreasing the Size of a File System

When decreasing the size of the file system on a device, ensure that 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, ensure that you decrease the size of the file system before you attempt to decrease the size of the device or logical volume.

Important: XFS

Decreasing the size of a file system formatted with XFS is not possible, since such a feature is not supported by XFS.

2.3 Changing the Size of a Btrfs File System

The size of a Btrfs file system can be changed by using the **`btrfs filesystem resize`** command when the file system is mounted. Increasing and decreasing the size are both supported while the file system is mounted.

1. Open a terminal.
2. Make sure the file system you want to change is mounted.
3. Change the size of the file system using the **`btrfs filesystem resize`** command with one of the following methods:
   - To extend the file system size to the maximum available size of the device, enter

     ```
     sudo btrfs filesystem resize max /mnt
     ```

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

     ```
     sudo btrfs filesystem resize SIZE /mnt
     ```

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 or decrease to the current size by prefixing the value with a plus (+) or a minus (-) sign, respectively:

```
sudo btrfs filesystem resize +SIZE /mnt
sudo btrfs filesystem resize -SIZE /mnt
```

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 prints sizes in human-readable format, such as 1K, 234M, or 2G.

2.4 Changing the Size of an XFS File System

The size of an XFS file system can be increased by using the **xfs_growfs** command when the file system is mounted. Reducing the size of an XFS file system is not possible.

1. Open a terminal.
2. Make sure the file system you want to change is mounted.
3. Increase the size of the file system using the **xfs_growfs** command. The following example expands the size of the file system to the maximum value available. See **man 8 xfs_growfs** for more options.

```
sudo xfs_growfs -d /mnt
```

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 prints sizes in human-readable format, such as 1K, 234M, or 2G.

2.5 Changing the Size of an Ext2, Ext3, or Ext4 File System

The size of Ext2, Ext3, and Ext4 file systems can be increased by using the **`resize2fs`** command, regardless of whether the respective partition is mounted or not. To decrease the size of an Ext file system it needs to be unmounted.

1. Open a terminal.
2. If the file system size should be decreased, unmount it.
3. Change 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

     ```
     sudo resize2fs /dev/sda1
     ```

     If a size parameter is not specified, the size defaults to the size of the partition.

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

     ```
     sudo 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 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.
4. If the file system is not mounted, mount it now.
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 prints sizes in human-readable format, such as 1K, 234M, or 2G.

2.6 Changing the Size of a Reiser File System

A ReiserFS file system can be increased in size while mounted or unmounted. To decrease its size it needs to be unmounted.

1. Open a terminal console.
2. If you want to decrease the size of the file system, unmount it in case it is mounted.
3. Change 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

     ```
     sudo 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

     ```
     sudo 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 or decrease to the current size by prefixing the value with a plus (`+`) or minus (`-`) sign, respectively:

     ```
     sudo resize_reiserfs -s +SIZE /dev/sda2
     sudo resize_reiserfs -s -SIZE /dev/sda2
     ```

   Wait until the resizing is completed before continuing.
4. If the file system is not mounted, mount it now.
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 prints sizes in human-readable format, such as 1K, 234M, or 2G.

3 Using UUIDs to Mount Devices

This section describes the use of UUIDs (Universally Unique Identifiers) instead of device names (such as `/dev/sda1`) to identify file system devices. Starting with SUSE Linux Enterprise Server 12, UUIDs are used by default in the boot loader file and the `/etc/fstab` file.

3.1 Persistent Device Names with udev

Starting with Linux kernel 2.6, **`udev`** provides a user space 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: UUID Generators

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

3.2 Understanding UUIDs

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

Using non-persistent "traditional" device names such as `/dev/sda1` may render the system unbootable when adding storage. For example, if root (`/`) is assigned to `/dev/sda1`, it might be reassigned to `/dev/sdg1` after a SAN has been attached or additional hard disks have been applied to the system. In this case the boot loader configuration and the `/etc/fstab` file need to be adjusted, otherwise the system will no longer boot.

One way to avoid this problem is to use the UUID in the boot loader and `/etc/fstab` files for the boot device. This is the default in SUSE Linux Enterprise since version 12. The UUID is a property of the file system and can change if you reformat the drive. Other alternatives to using UUIDs of device names would be to identify devices by ID or label.

You can also 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.

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

```
tux > ls -og /dev/disk/by-uuid/
lrwxrwxrwx 1 10 Dec  5 07:48 e014e482-1c2d-4d09-84ec-61b3aefde77a -> ../../sda1
```

3.3 Additional Information

For more information about using `udev` for managing devices, see *Book "Administration Guide", Chapter 22 "Dynamic Kernel Device Management with* `udev`*"*.

For more information about `udev` commands, see **`man 7 udev`**.

3.4 Mounting network storage devices

Some types of storage devices require network to be configured and available before **`systemd.mount`** starts to mount the devices. To postpone mounting of these types of devices, add the `_netdev` option to the `/etc/fstab` file for each particular network storage device. An example follows:

```
mars.example.org:/nfsexport  /shared   nfs  defaults,_netdev    0  0
```

4 Multi-tier Caching for Block Device Operations

A multi-tier cache is a replicated/distributed cache that consists of at least two tiers: one is represented by slower but cheaper rotational block devices (hard disks), while the other is more expensive but performs faster data operations (for example SSD flash disks).

SUSE Linux Enterprise Server implements two different solutions for caching between flash and rotational devices: `bcache` and `lvmcache`.

4.1 General Terminology

This section explains several terms often used when describing cache related features:

Migration

Movement of the primary copy of a logical block from one device to the other.

Promotion

Migration from the slow device to the fast device.

Demotion

Migration from the fast device to the slow device.

Origin device

The big and slower block device. It always contains a copy of the logical block, which may be out of date or kept in synchronization with the copy on the cache device (depending on policy).

Cache device

The small and faster block device.

Metadata device

A small device that records which blocks are in the cache, which are dirty, and extra hints for use by the policy object. This information could be put on the cache device as well, but having it separate allows the volume manager to configure it differently, for example as a mirror for extra robustness. The metadata device may only be used by a single cache device.

Dirty block

If some process writes to a block of data which is placed in the cache, the cached block is marked as *dirty* because it was overwritten in the cache and needs to be written back to the original device.

Cache miss

A request for I/O operations is pointed to the cached device's cache first. If it cannot find the requested values, it looks in the device itself, which is slow. This is called a *cache miss*.

Cache hit

When a requested value is found in the cached device's cache, it is served fast. This is called a *cache hit*.

Cold cache

Cache that holds no values (is empty) and causes *cache misses*. As the cached block device operations progress, it gets filled with data and becomes *warm*.

Warm cache

Cache that already holds some values and is likely to result in *cache hits*.

4.2 Caching Modes

Following are the basic caching modes that multi-tier caches use: *write-back*, *write-through*, *write-around* and *pass-through*.

write-back

Data written to a block that is cached go to the cache only, and the block is marked dirty. This is the default caching mode.

write-through

Writing to a cached block will not complete until it has hit both the origin and cache devices. Clean blocks remain clean with *write-through* cache.

write-around

A similar technique to write-through cache, but write I/O is written directly to a permanent storage, bypassing the cache. This can prevent the cache being flooded with write I/O that will not subsequently be re-read, but the disadvantage is that a read request for recently written data will create a 'cache miss' and needs to be read from slower bulk storage and experience higher latency.

pass-through

To enable the *pass-through* mode, the cache needs to be clean. Reading is served from the origin device bypassing the cache. Writing is forwarded to the origin device and 'invalidates' the cache block. *Pass-through* allows a cache device activation without having to care about data coherency, which is maintained. The cache will gradually become cold as writing takes place. If you can verify the coherency of the cache later, or establish it by using the `invalidate_cblocks` message, you can switch the cache device to *write-through* or *write-back* mode while it is still warm. Otherwise, you can discard the cache contents before switching to the desired caching mode.

4.3 bcache

`bcache` is a Linux kernel block layer cache. It allows one or more fast disk drives (such as SSDs) to act as a cache for one or more slower hard disks. `bcache` supports write-through and write-back, and is independent of the file system used. By default it caches random reads and writes only, which SSDs excel at. It is suitable for desktops, servers, and high end storage arrays as well.

4.3.1 Main Features

- A single cache device can be used to cache an arbitrary number of backing devices. Backing devices can be attached and detached at runtime, while mounted and in use.
- Recovers from unclean shutdowns—writes are not completed until the cache is consistent with regard to the backing device.
- Throttles traffic to the SSD if it becomes congested.
- Highly efficient write-back implementation. Dirty data is always written out in sorted order.
- Stable and reliable—in production use.

4.3.2 Setting Up a bcache Device

This section describes steps to set up and manage a `bcache` device.

1. Install the `bcache-tools` package:

```
sudo zypper in bcache-tools
```

2. Create a backing device (typically a mechanical drive). The backing device can be a whole device, a partition, or any other standard block device.

```
sudo make-bcache -B /dev/sdb
```

3. Create a cache device (typically an SSD disk).

```
sudo make-bcache -C /dev/sdc
```

   In this example, the default block and bucket sizes of 512 B and 128 KB are used. The block size should match the backing device's sector size which will usually be either 512 or 4k. The bucket size should match the erase block size of the caching device with the intention of reducing write amplification. For example, using a hard disk with 4k sectors and an SSD with an erase block size of 2 MB this command would look as follows:

```
sudo make-bcache --block 4k --bucket 2M -C /dev/sdc
```

Tip: Multi-Device Support

`make-bcache` can prepare and register multiple backing devices and a cache device at the same time. In this case you do not need to manually attach the cache device to the backing device afterward:

```
sudo make-bcache -B /dev/sda /dev/sdb -C /dev/sdc
```

4. `bcache` devices show up as

```
/dev/bcacheN
```

   and as

```
/dev/bcache/by-uuid/UUID
/dev/bcache/by-label/LABEL
```

   You can normally format and mount `bcache` devices as usual:

```
mkfs.ext4 /dev/bcache0
```

```
mount /dev/bcache0 /mnt
```

You can control `bcache` devices through `sysfs` at `/sys/block/bcache*N*/bcache`.

5. After both the cache and backing devices are registered, you need to attach the backing device to the related cache set to enable caching:

```
echo CACHE_SET_UUID > /sys/block/bcache0/bcache/attach
```

where *`CACHE_SET_UUID`* is found in `/sys/fs/bcache`.

6. By default `bcache` uses a pass-through caching mode. To change it to for example write-back, run

```
echo writeback > /sys/block/bcache0/bcache/cache_mode
```

4.3.3 bcache Configuration Using `sysfs`

`bcache` devices use the `sysfs` interface to store their runtime configuration values. This way you can change `bcache` backing and cache disks' behavior or see their usage statistics.

For the complete list of `bcache` `sysfs` parameters, see the contents of the `/usr/src/linux/Documentation/bcache.txt` file, mainly the `SYSFS - BACKING DEVICE`, `SYSFS - BACKING DEVICE STATS`, and `SYSFS - CACHE DEVICE` sections.

4.4 `lvmcache`

`lvmcache` is a caching mechanism consisting of logical volumes (LVs). It uses the `dm-cache` kernel driver and supports write-through (default) and write-back caching modes. `lvmcache` improves performance of a large and slow LV by dynamically migrating some of its data to a faster and smaller LV. For more information on LVM, see *Part II, "Logical Volumes (LVM)"*.

LVM refers to the small, fast LV as a *cache pool LV*. The large, slow LV is called the *origin LV*. Because of requirements from dm-cache, LVM further splits the cache pool LV into two devices: the *cache data LV* and *cache metadata LV*. The cache data LV is where copies of data blocks are kept from the origin LV to increase speed. The cache metadata LV holds the accounting information that specifies where data blocks are stored.

4.4.1 Configuring lvmcache

This section describes steps to create and configure LVM based caching.

1. *Create the origin LV.* Create a new LV or use an existing LV to become the origin LV:

```
lvcreate -n ORIGIN_LV -L 100G vg /dev/SLOW_DEV
```

2. *Create the cache data LV.* This LV will hold data blocks from the origin LV. The size of this LV is the size of the cache and will be reported as the size of the cache pool LV.

```
lvcreate -n CACHE_DATA_LV -L 10G vg /dev/FAST
```

3. *Create the cache metadata LV.* This LV will hold cache pool metadata. The size of this LV should be approximately 1000 times smaller than the cache data LV, with a minimum size of 8MB.

```
lvcreate -n CACHE_METADATA_LV -L 12M vg /dev/FAST
```

   List the volumes you have created so far:

```
lvs -a vg
LV                VG   Attr        LSize   Pool Origin
cache_data_lv     vg   -wi-a-----  10.00g
cache_metadata_lv vg   -wi-a-----  12.00m
origin_lv         vg   -wi-a----- 100.00g
```

4. *Create a cache pool LV.* Combine the data and metadata LVs into a cache pool LV. You can set the cache pool LV's behavior at the same time.
   CACHE_POOL_LV takes the name of *CACHE_DATA_LV*.
   CACHE_DATA_LV is renamed to *CACHE_DATA_LV*_cdata and becomes hidden.
   CACHE_META_LV is renamed to *CACHE_DATA_LV*_cmeta and becomes hidden.

```
lvconvert --type cache-pool \
 --poolmetadata vg/cache_metadata_lv vg/cache_data_lv
```

```
lvs -a vg
LV                    VG   Attr       LSize   Pool Origin
cache_data_lv         vg   Cwi---C---  10.00g
[cache_data_lv_cdata] vg   Cwi-------  10.00g
[cache_data_lv_cmeta] vg   ewi-------  12.00m
origin_lv             vg   -wi-a----- 100.00g
```

5. *Create a cache LV.* Create a cache LV by linking the cache pool LV to the origin LV.

The user accessible cache LV takes the name of the origin LV, while the origin LV becomes a hidden LV renamed to *ORIGIN_LV*_corig.
CacheLV takes the name of *ORIGIN_LV*.
ORIGIN_LV is renamed to *ORIGIN_LV*_corig and becomes hidden.

```
lvconvert --type cache --cachepool vg/cache_data_lv vg/origin_lv
```

```
lvs -a vg
LV              VG   Attr       LSize   Pool   Origin
cache_data_lv          vg   Cwi---C---  10.00g
[cache_data_lv_cdata]  vg   Cwi-ao----  10.00g
[cache_data_lv_cmeta]  vg   ewi-ao----  12.00m
origin_lv              vg   Cwi-a-C--- 100.00g cache_data_lv [origin_lv_corig]
[origin_lv_corig]      vg   -wi-ao---- 100.00g
```

4.4.2 Removing a Cache Pool

There are several ways to turn off the LV cache.

4.4.2.1 Detach a Cache Pool LV from a Cache LV

You can disconnect a cache pool LV from a cache LV, leaving an unused cache pool LV and an uncached origin LV. Data are written back from the cache pool to the origin LV when necessary.

```
lvconvert --splitcache vg/origin_lv
```

4.4.2.2 Removing a Cache Pool LV without Removing its Origin LV

This writes back data from the cache pool to the origin LV when necessary, then removes the cache pool LV, leaving the uncached origin LV.

```
lvremove vg/cache_data_lv
```

An alternative command that also disconnects the cache pool from the cache LV, and deletes the cache pool:

```
lvconvert --uncache vg/origin_lv
```

4.4.2.3 Removing Both the Origin LV and the Cache Pool LV

Removing a cache LV removes both the origin LV and the linked cache pool LV.

```
lvremove vg/origin_lv
```

4.4.2.4 For More Information

You can find more `lvmcache` related topics, such as supported cache modes, redundant sub-logical volumes, cache policy, or converting existing LVs to cache types, in the `lvmcache` manual page (**`man 7 lvmcache`**).

II Logical Volumes (LVM)

5 LVM Configuration

This chapter 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: Risks

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.

5.1 Understanding the Logical Volume Manager

LVM enables flexible distribution of hard disk space over several physical volumes (hard disks, partitions, LUNs). 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 storage 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 5.1, "Physical Partitioning versus LVM" 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.

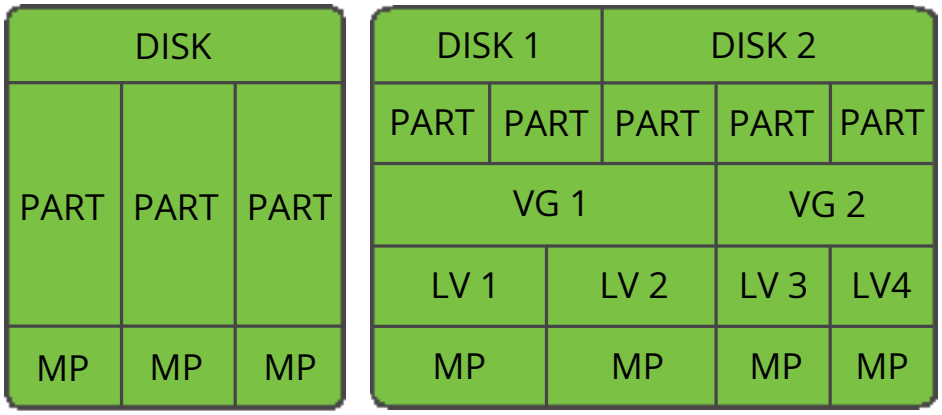


FIGURE 5.1: PHYSICAL PARTITIONING VERSUS LVM

In LVM, the physical disk partitions that are incorporated in a volume group are called physical volumes (PVs). Within the volume groups in *Figure 5.1, "Physical Partitioning versus LVM"*, four logical volumes (LV 1 through LV 4) have been defined, which can be used by the operating system via the associated mount points (MP). 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 hotpluggable 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 like RAID 0.
- The snapshot feature enables consistent backups (especially for servers) in the running system.

Note: LVM and RAID

Even though LVM also supports RAID of 0/1/4/5/6 levels, we recommend to use MD RAID (see *Chapter 7, Software RAID Configuration*). However, LVM works fine with RAID 0 and 1, as RAID 0 is similar to common logical volume management (individual logical

blocks are mapped onto blocks on the physical devices). LVM used on top of RAID 1 can keep track of mirror synchronization and is fully able to manage the synchronization process. With higher RAID levels you need a management daemon that monitors the states of attached disks and can inform administrators if there is a problem in the disk array. LVM includes such a daemon, but in exceptional situations like a device failure, the daemon does not working properly.

Warning: IBM IBM Z: LVM Root File System

If you configure the system with a root file system on LVM or a software RAID array, you must place `/boot` on a separate, non-LVM or non-RAID partition, otherwise the system will fail to boot. The recommended size for such a partition is 500 MB and the recommended file system is Ext4.

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. However, keep in mind that working with LVM is different from working with conventional partitions.

You can manage new or existing LVM storage objects by using the YaST Partitioner. Instructions and further information about configuring LVM are available in the official *LVM HOWTO* (https://tldp.org/HOWTO/LVM-HOWTO/).

5.2 Creating Volume Groups

An LVM volume group (VG) organizes the Linux LVM partitions into a logical pool of space. You can carve out logical volumes from the available space in the group. The Linux LVM partitions in a group can be on the same or different disks. You can add partitions or entire disks to expand the size of the group.

To use an entire disk, it must not contain any partitions. When using partitions, they must not be mounted. YaST will automatically change their partition type to `0x8E Linux LVM` when adding them to a VG.

1. Launch YaST and open the *Partitioner*.

2. In case you need to reconfigure your existing partitioning setup, proceed as follows. Refer to *Book "Deployment Guide", Chapter 13 "Advanced Disk Setup", Section 13.1 "Using the YaST Partitioner"* for details. Skip this step if you only want to use unused disks or partitions that already exist.

Warning: Physical Volumes on Unpartitioned Disks

You can use an unpartitioned disk as a physical volume (PV) if that disk is *not* the one where the operating system is installed and from which it boots.

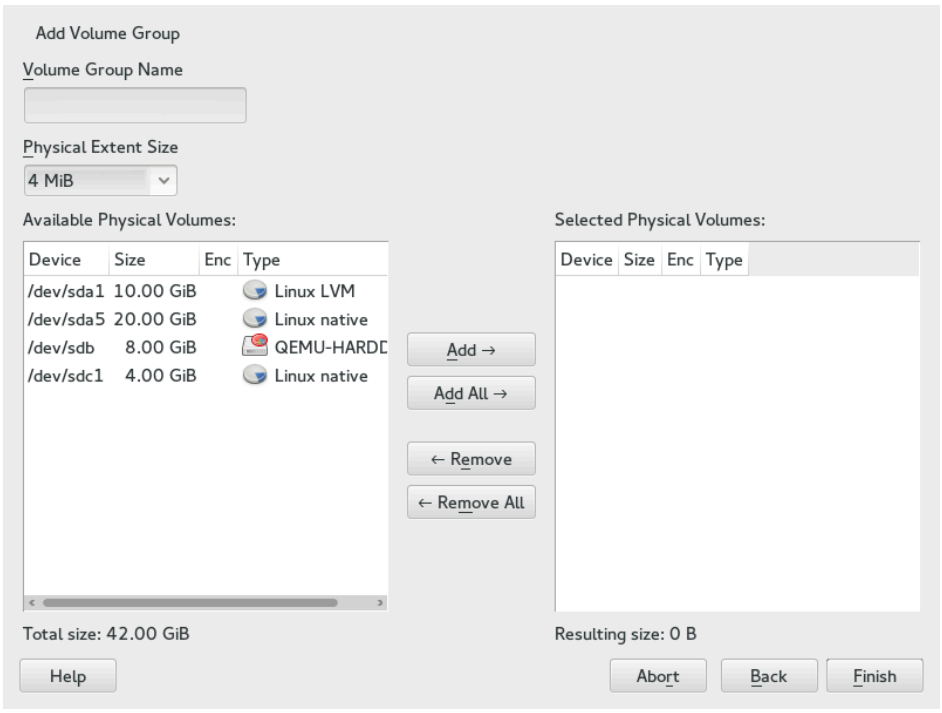
As unpartitioned disks appear as *unused* at the system level, they can easily be overwritten or wrongly accessed.

   a. To use an entire hard disk that already contains partitions, delete all partitions on that disk.

   b. To use a partition that is currently mounted, unmount it.

3. In the left panel, select *Volume Management*.
   A list of existing Volume Groups opens in the right panel.

4. At the lower left of the Volume Management page, click *Add* › *Volume Group*.



5. Define the volume group as follows:

   a. Specify the *Volume Group Name*.
      If you are creating a volume group at install time, the name `system` is suggested for a volume group that will contain the SUSE Linux Enterprise Server system files.

   b. Specify the *Physical Extent Size*.
      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. Values can be from 1 KB to 16 GB in powers of 2. This value is normally set to 4 MB.
      In LVM1, a 4 MB physical extent allowed a maximum LV size of 256 GB because it supports only up to 65534 extents per LV. LVM2, which is used on SUSE Linux Enterprise Server, does not restrict the number of physical extents. Having many extents has no impact on I/O performance to the logical volume, but it slows down the LVM tools.

      #### Important: Physical Extent Sizes

      Different physical extent sizes should not be mixed in a single VG. The extent should not be modified after the initial setup.

   c. In the *Available Physical Volumes* list, select the Linux LVM partitions that you want to make part of this volume group, then click *Add* to move them to the *Selected Physical Volumes* list.

   d. Click *Finish*.
      The new group appears in the *Volume Groups* list.

6. On the Volume Management page, click *Next*, verify that the new volume group is listed, then click *Finish*.

7. To check which physical devices are part of the volume group, open the YaST Partitioner at any time in the running system and click *Volume Management* › *Edit* › *Physical Devices*. Leave this screen with *Abort*.

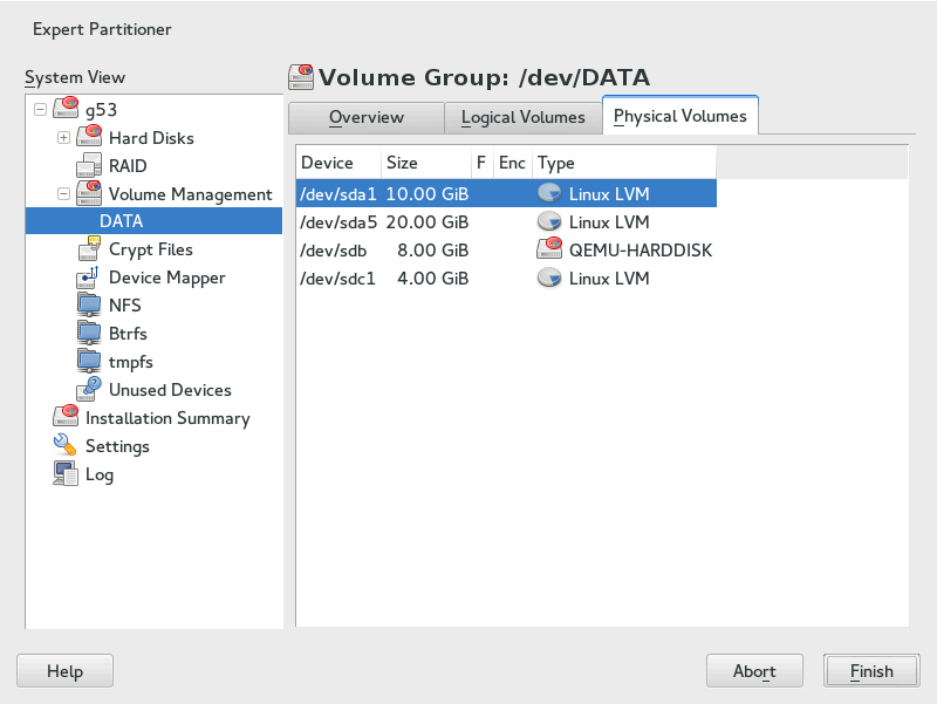


FIGURE 5.2: PHYSICAL VOLUMES IN THE VOLUME GROUP NAMED DATA

5.3 Creating Logical Volumes

A logical volume provides a pool of space similar to what a hard disk does. To make this space usable, you need to define logical volumes. A logical volume is similar to a regular partition—you can format and mount it.

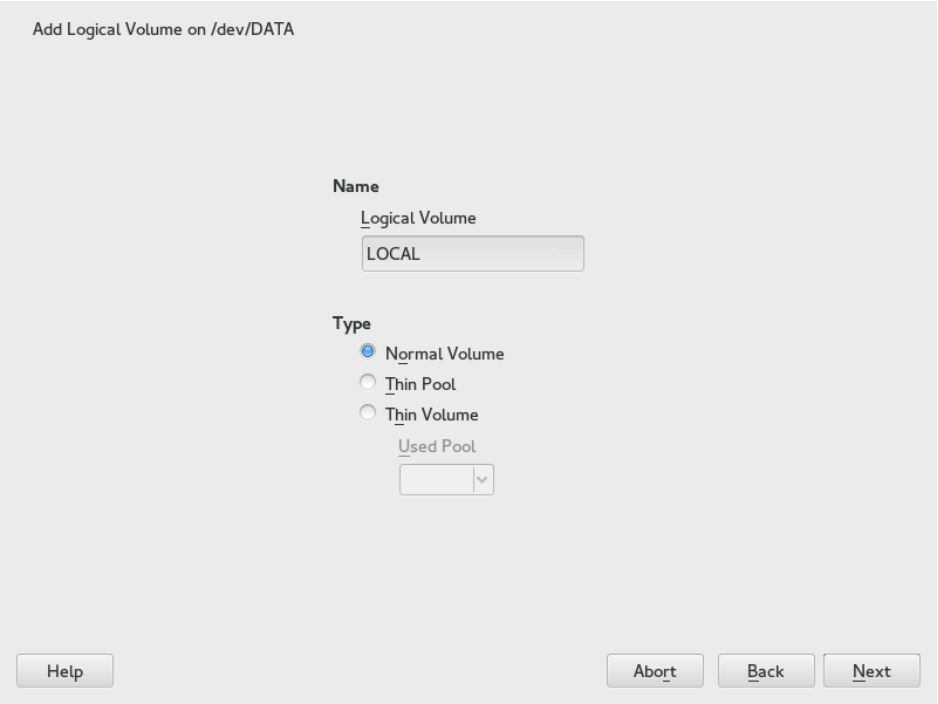
Use The YaST Partitioner to create logical volumes from an existing volume group. Assign at least one logical volume to each volume group. You can create new logical volumes as needed until all free space in the volume group has been exhausted. An LVM logical volume can optionally be thinly provisioned, allowing you to create logical volumes with sizes that overbook the available free space (see *Section 5.3.1, "Thinly Provisioned Logical Volumes"* for more information).

- **Normal volume:** (Default) The volume's space is allocated immediately.
- **Thin pool:** The logical volume is a pool of space that is reserved for use with thin volumes. The thin volumes can allocate their needed space from it on demand.

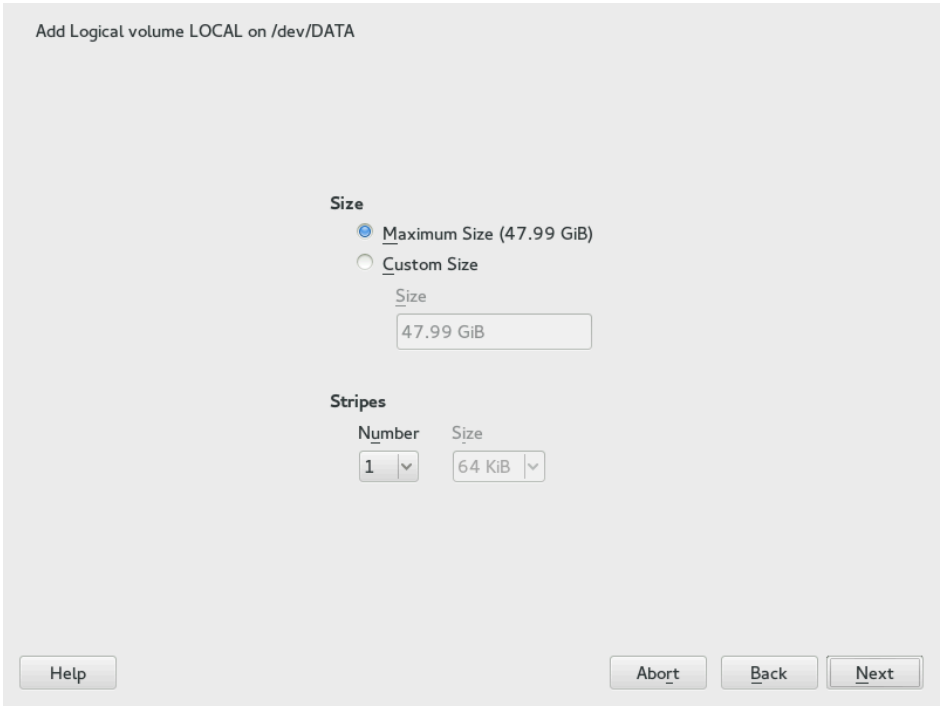
- **Thin volume:** The volume is created as a sparse volume. The volume allocates needed space on demand from a thin pool.
- **Mirrored volume:** The volume is created with a defined count of mirrors.

PROCEDURE 5.1: SETTING UP A LOGICAL VOLUME

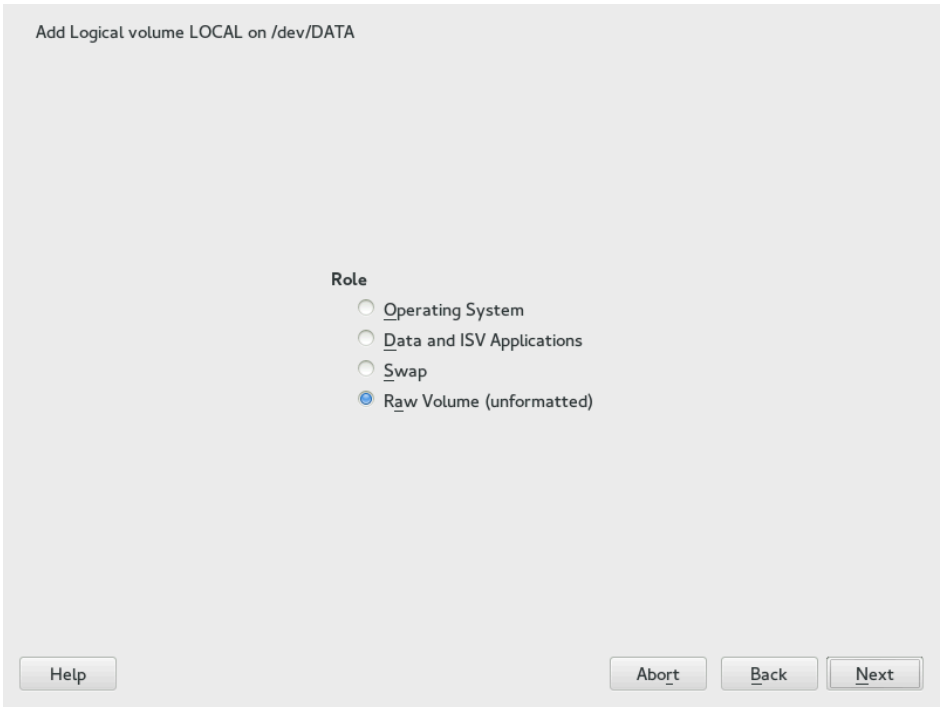
1. Launch YaST and open the *Partitioner*.
2. In the left panel, select *Volume Management*. A list of existing Volume Groups opens in the right panel.
3. Select the volume group in which you want to create the volume and choose *Add* › *Logical Volume*.
4. Provide a *Name* for the volume and choose *Normal Volume* (refer to *Section 5.3.1, "Thinly Provisioned Logical Volumes"* for setting up thinly provisioned volumes). Proceed with *Next*.



5. Specify the size of the volume and whether to use multiple stripes.
   Using a striped volume, the data will be distributed among several physical volumes. If these physical volumes reside on different hard disks, this generally results in a better reading and writing performance (like RAID 0). The maximum number of available stripes is equal to the number of physical volumes. The default (`1` is to not use multiple stripes.



6. Choose a *Role* for the volume. Your choice here only affects the default values for the upcoming dialog. They can be changed in the next step. If in doubt, choose *Raw Volume (Unformatted)*.



7. Under *Formatting Options*, select *Format Partition*, then select the *File system*. The content of the *Options* menu depends on the file system. Usually there is no need to change the defaults.
   Under *Mounting Options*, select *Mount partition*, then select the mount point. Click *Fstab Options* to add special mounting options for the volume.
8. Click *Finish*.
9. Click *Next*, verify that the changes are listed, then click *Finish*.

5.3.1 Thinly Provisioned Logical Volumes

An LVM logical volume can optionally be thinly provisioned. Thin provisioning allows you to create logical volumes with sizes that overbook the available free space. You create a thin pool that contains unused space reserved for use with an arbitrary number of thin volumes. A thin volume is created as a sparse volume and space is allocated from a thin pool as needed. The thin pool can be expanded dynamically when needed for cost-effective allocation of storage space. Thinly provisioned volumes also support snapshots which can be managed with Snapper—see *Book "Administration Guide", Chapter 7 "System Recovery and Snapshot Management with Snapper"* for more information.

To set up a thinly provisioned logical volume, proceed as described in *Procedure 5.1, "Setting Up a Logical Volume"*. When it comes to choosing the volume type, do not choose *Normal Volume*, but rather *Thin Volume* or *Thin Pool*.

Thin pool

The logical volume is a pool of space that is reserved for use with thin volumes. The thin volumes can allocate their needed space from it on demand.

Thin volume

The volume is created as a sparse volume. The volume allocates needed space on demand from a thin pool.

Important: Thinly Provisioned Volumes in a Cluster

To use thinly provisioned volumes in a cluster, the thin pool and the thin volumes that use it must be managed in a single cluster resource. This allows the thin volumes and thin pool to always be mounted exclusively on the same node.

5.3.2 Creating Mirrored Volumes

A logical volume can be created with several mirrors. LVM ensures that data written to an underlying physical volume is mirrored onto a different physical volume. Thus even though a physical volume crashes, you can still access the data on the logical volume. LVM also keeps a log file to manage the synchronization process. The log contains information about which volume regions are currently undergoing synchronization with mirrors. By default the log is stored on disk and if possible on a different disk than are the mirrors. But you may specify a different location for the log, for example volatile memory.

Currently there are two types of mirror implementation available: "normal" (non-raid) `mirror` logical volumes and `raid1` logical volumes.

After you create mirrored logical volumes, you can perform standard operations with mirrored logical volumes like activating, extending, and removing.

5.3.2.1 Setting Up Mirrored Non-raid Logical Volumes

To create a mirrored volume use the **`lvcreate`** command. The following example creates a 500 GB logical volume with two mirrors called *lv1* which uses a volume group *vg1*.

```
lvcreate -L 500G -m 2 -n lv1 vg1
```

Such a logical volume is a linear volume (without striping) that provides three copies of the file system. The `m` option specifies the count of mirrors. The `L` option specifies the size of the logical volumes.

The logical volume is divided into regions of the 512 KB default size. If you need a different size of regions, use the `-R` option followed by the desired region size in megabytes. Or you can configure the preferred region size by editing the `mirror_region_size` option in the `lvm.conf` file.

5.3.2.2 Setting Up `raid1` Logical Volumes

As LVM supports RAID you can implement mirroring by using RAID1. Such implementation provides the following advantages compared to the non-raid mirrors:

- LVM maintains a fully redundant bitmap area for each mirror image, which increases its fault handling capabilities.
- Mirror images can be temporarily split from the array and then merged back.

- The array can handle transient failures.
- The LVM RAID 1 implementation supports snapshots.

On the other hand, this type of mirroring implementation does not enable to create a logical volume in a clustered volume group.

To create a mirror volume by using RAID, issue the command

```
lvcreate --type raid1 -m 1 -L 1G -n lv1 vg1
```

where the options/parameters have the following meanings:

- `--type` - you need to specify `raid1`, otherwise the command uses the implicit segment type `mirror` and creates a non-raid mirror.
- `-m` - specifies the count of mirrors.
- `-L` - specifies the size of the logical volume.
- `-n` - by using this option you specify a name of the logical volume.
- `vg1` - is a name of the volume group used by the logical volume.

LVM creates a logical volume of one extent size for each data volume in the array. If you have two mirrored volumes, LVM creates another two volumes that stores metadata.

After you create a RAID logical volume, you can use the volume in the same way as a common logical volume. You can activate it, extend it, etc.

5.4 Automatically Activating Non-Root LVM Volume Groups

Activation behavior for non-root LVM volume groups is controlled in the `/etc/lvm/lvm.conf` file and by the `auto_activation_volume_list` parameter. By default, the parameter is empty and all volumes are activated. To activate only some volume groups, add the names in quotes and separate them with commas, for example:

```
auto_activation_volume_list = [ "vg1", "vg2/lvol1", "@tag1", "@*" ]
```

If you have defined a list in the `auto_activation_volume_list` parameter, the following will happen:

1. Each logical volume is first checked against this list.
2. If it does not match, the logical volume will not be activated.

By default, non-root LVM volume groups are automatically activated on system restart by Dracut. This parameter allows you to activate all volume groups on system restart, or to activate only specified non-root LVM volume groups.

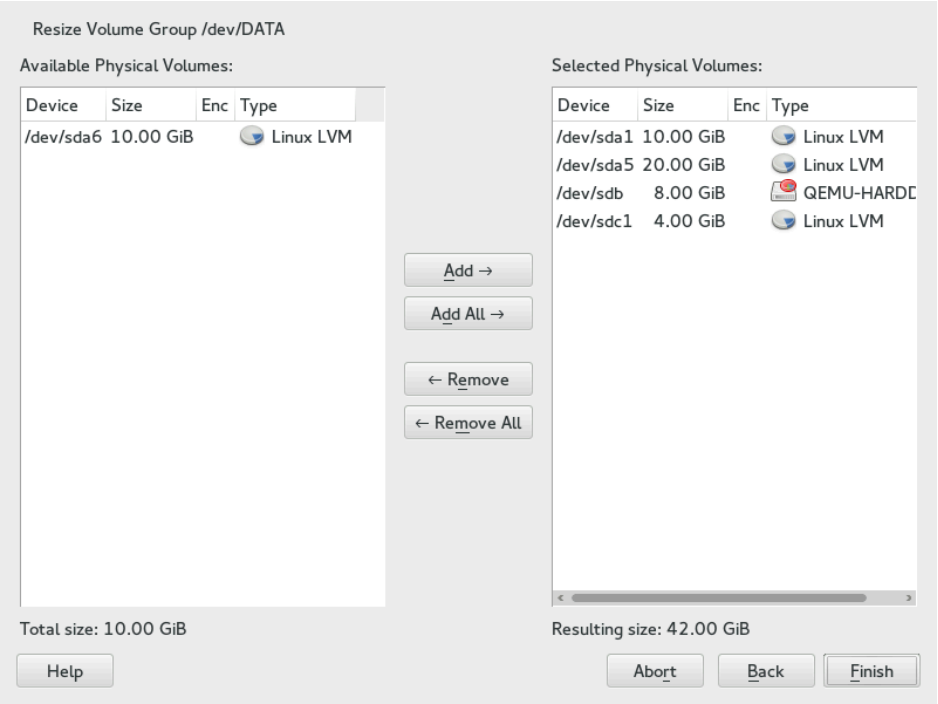
5.5 Resizing an Existing Volume Group

The space provided by a volume group can be expanded at any time in the running system without service interruption by adding more physical volumes. This will allow you to add logical volumes to the group or to expand the size of existing volumes as described in *Section 5.6, "Resizing a Logical Volume"*.

It is also possible to reduce the size of the volume group by removing physical volumes. YaST only allows to remove physical volumes that are currently unused. To find out which physical volumes are currently in use, run the following command. The partitions (physical volumes) listed in the `PE Ranges` column are the ones in use:

```
tux > sudo pvs -o vg_name,lv_name,pv_name,seg_pe_ranges
root's password:
  VG   LV    PV         PE Ranges
             /dev/sda1
  DATA DEVEL /dev/sda5  /dev/sda5:0-3839
  DATA       /dev/sda5
  DATA LOCAL /dev/sda6  /dev/sda6:0-2559
  DATA       /dev/sda7
  DATA       /dev/sdb1
  DATA       /dev/sdc1
```

1. Launch YaST and open the *Partitioner*.
2. In the left panel, select *Volume Management*. A list of existing Volume Groups opens in the right panel.
3. Select the volume group you want to change, then click *Resize*.



4. Do one of the following:

   - **Add:** Expand the size of the volume group by moving one or more physical volumes (LVM partitions) from the *Available Physical Volumes* list to the *Selected Physical Volumes* list.

   - **Remove:** Reduce the size of the volume group by moving one or more physical volumes (LVM partitions) from the *Selected Physical Volumes* list to the *Available Physical Volumes* list.

5. Click *Finish*.

6. Click *Next*, verify that the changes are listed, then click *Finish*.

5.6 Resizing a Logical Volume

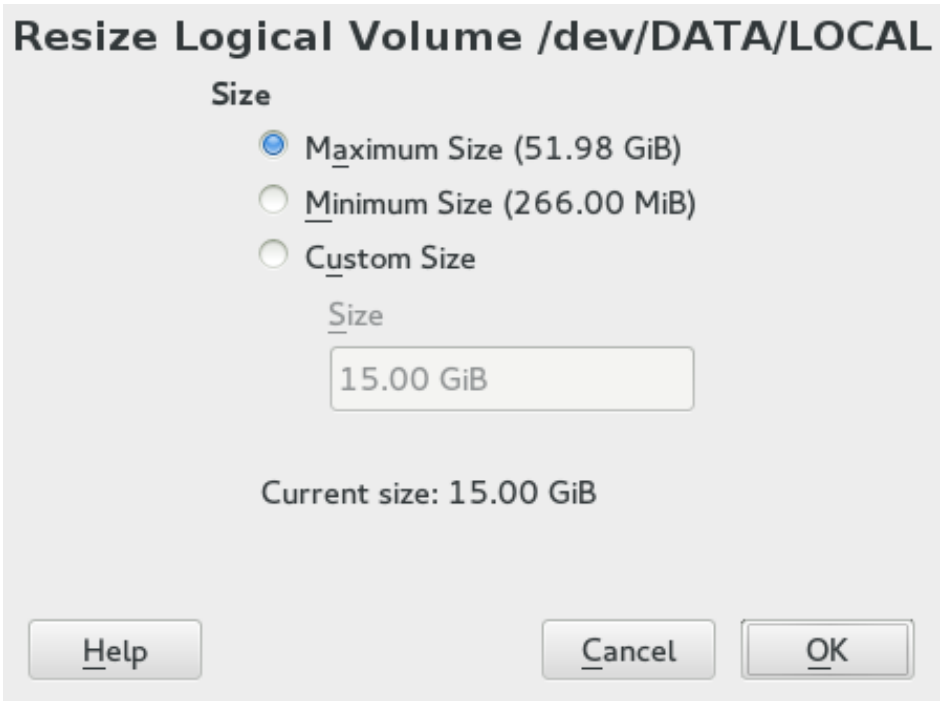
In case there is unused free space available in the volume group, you can enlarge a logical volume to provide more usable space. You may also reduce the size of a volume to free space in the volume group that can be used by other logical volumes.

Note: “Online” Resizing

When reducing the size of a volume, YaST automatically resizes its file system, too. Whether a volume that is currently mounted can be resized “online” (that is while being mounted), depends on its file system. Growing the file system online is supported by Btrfs, XFS, Ext3, and ReiserFS.

Shrinking the file system online is only supported by Btrfs. To shrink XFS, Ext2/3/4, and ReiserFS volumes, you need to unmount them. Shrinking volumes formatted with XFS is not possible, since XFS does not support file system shrinking.

1. Launch YaST and open the *Partitioner*.
2. In the left panel, select *Volume Management*. A list of existing Volume Groups opens in the right panel.
3. Select the logical volume you want to change, then click *Resize*.



4. Set the intended size by using one of the following options:

   - **Maximum Size.** Expand the size of the logical volume to use all space left in the volume group.
   - **Minimum Size.** Reduce the size of the logical volume to the size occupied by the data and the file system metadata.
   - **Custom Size.** Specify the new size for the volume. The value must be within the range of the minimum and maximum values listed above. Use K, M, G, T for Kilobytes, Megabytes, Gigabytes and Terabytes (for example `20G`).

5. Click *OK*.

6. Click *Next*, verify that the change is listed, then click *Finish*.

5.7 Deleting a Volume Group or a Logical Volume

Warning: Data Loss

Deleting a volume group destroys all of the data in each of its member partitions. Deleting a logical volume destroys all data stored on the volume.

1. Launch YaST and open the *Partitioner*.

2. In the left panel, select *Volume Management*. A list of existing volume groups opens in the right panel.

3. Select the volume group or the logical volume you want to remove and click *Delete*.

4. Depending on your choice warning dialogs are shown. Confirm them with *Yes*.

5. Click *Next*, verify that the deleted volume group is listed (deletion is indicated by a red colored font), then click *Finish*.

5.8 Using LVM Commands

For information about using LVM commands, see the man pages for the commands described in the following table. All commands need to be executed with `root` privileges. Either use **`sudo`** *`COMMAND`* (recommended) or execute them directly as `root`.

LVM COMMANDS

`pvcreate` *`DEVICE`*

Initializes a device (such as `/dev/sdb1`) for use by LVLM as a physical volume. If there is any file system on the specified device, a warning appears. Bear in mind that **`pvcreate`** checks for existing file systems only if **`blkid`** is installed (which is done by default). If **`blkid`** is not available, **`pvcreate`** will not produce any warning and you may lose your file system without any warning.

`pvdisplay` *`DEVICE`*

Displays information about the LVM physical volume, such as whether it is currently being used in a logical volume.

`vgcreate -c y` *`VG_NAME DEV1 [DEV2...]`*

Creates a clustered volume group with one or more specified devices.

`vgcreate --activationmode` *`ACTIVATION_MODE VG_NAME`*

Configures the mode of volume group activation. You can specify one of the following values:

- `complete` - only the logical volumes that are not affected by missing physical volumes can be activated, even though the particular logical volume can tolerate such a failure.
- `degraded` - is the default activation mode. If there is a sufficient level of redundancy to activate a logical volume, the logical volume can be activated even though some physical volumes are missing.
- `partial` - the LVM tries to activate the volume group even though some physical volumes are missing. If a non-redundant logical volume is missing important physical volumes, then the logical volume usually cannot be activated and is handled as an error target.

`vgchange -a [ey|n]` *`VG_NAME`*

Activates (`-a ey`) or deactivates (`-a n`) a volume group and its logical volumes for input/output.

When activating a volume in a cluster, ensure that you use the `ey` option. This option is used by default in the load script.

`vgremove` *`VG_NAME`*

Removes a volume group. Before using this command, remove the logical volumes, then deactivate the volume group.

`vgdisplay` *`VG_NAME`*

Displays information about a specified volume group.

To find the total physical extent of a volume group, enter

```
vgdisplay VG_NAME | grep "Total PE"
```

`lvcreate -L` *`SIZE`* **`-n`** *`LV_NAME VG_NAME`*

Creates a logical volume of the specified size.

`lvcreate -L` *`SIZE`* **`--thinpool`** *`POOL_NAME VG_NAME`*

Creates a thin pool named `myPool` of the specified size from the volume group *`VG_NAME`*. The following example creates a thin pool with a size of 5 GB from the volume group `LOCAL`:

```
lvcreate -L 5G --thinpool myPool LOCAL
```

`lvcreate -T` *`VG_NAME/POOL_NAME`* **`-V`** *`SIZE`* **`-n`** *`LV_NAME`*

Creates a thin logical volume within the pool *`POOL_NAME`*. The following example creates a 1GB thin volume named `myThin1` from the pool `myPool` on the volume group `LOCAL`:

```
lvcreate -T LOCAL/myPool -V 1G -n myThin1
```

`lvcreate -T` *`VG_NAME/POOL_NAME`* **`-V`** *`SIZE`* **`-L`** *`SIZE`* **`-n`** *`LV_NAME`*

It is also possible to combine thin pool and thin logical volume creation in one command:

```
lvcreate -T LOCAL/myPool -V 1G -L 5G -n myThin1
```

lvcreate --activationmode *ACTIVATION_MODE LV_NAME*

Configures the mode of logical volume activation. You can specify one of the following values:

- `complete` - the logical volume can be activated only if all its physical volumes are active.
- `degraded` - is the default activation mode. If there is a sufficient level of redundancy to activate a logical volume, the logical volume can be activated even though some physical volumes are missing.
- `partial` - the LVM tries to activate the volume even though some physical volumes are missing. In this case part of the logical volume may be unavailable and it might cause data loss. This option is typically not used, but might be useful when restoring data.

You can specify the activation mode also in `/etc/lvm/lvm.conf` by specifying one of the above described values of the `activation_mode` configuration option.

lvcreate -s [-L *SIZE*] **-n** *SNAP_VOLUME SOURCE_VOLUME_PATH VG_NAME*

Creates a snapshot volume for the specified logical volume. If the size option (`-L` or `--size`) is not included, the snapshot is created as a thin snapshot.

lvremove /dev/*VG_NAME/LV_NAME*

Removes a logical volume.

Before using this command, close the logical volume by unmounting it with the **umount** command.

lvremove *SNAP_VOLUME_PATH*

Removes a snapshot volume.

lvconvert --merge *SNAP_VOLUME_PATH*

Reverts the logical volume to the version of the snapshot.

vgextend *VG_NAME DEVICE*

Adds the specified device (physical volume) to an existing volume group.

vgreduce *VG_NAME DEVICE*

Removes a specified physical volume from an existing volume group.

Ensure that the physical volume is not currently being used by a logical volume. If it is, you must move the data to another physical volume by using the **pvmove** command.

lvextend -L *SIZE* **/dev/***VG_NAME/LV_NAME*

Extends the size of a specified logical volume. Afterward, you must also expand the file system to take advantage of the newly available space. See *Chapter 2, Resizing File Systems* for details.

lvreduce -L *SIZE* **/dev/***VG_NAME/LV_NAME*

Reduces the size of a specified logical volume.

Ensure that you reduce the size of the file system first before shrinking the volume, otherwise you risk losing data. See *Chapter 2, Resizing File Systems* for details.

lvrename /dev/*VG_NAME/LV_NAME* **/dev/***VG_NAME/NEW_LV_NAME*

Renames an existing LVM logical volume. It does not change the volume group name.

Tip: Bypassing udev on Volume Creation

In case you want to manage LV device nodes and symbolic links by using LVM instead of by using udev rules, you can achieve this by disabling notifications from udev with one of the following methods:

- Configure `activation/udev_rules = 0` and `activation/udev_sync = 0` in `/etc/lvm/lvm.conf`.

  Note that specifying `--nodevsync` with the **lvcreate** command has the same effect as `activation/udev_sync = 0`; setting `activation/udev_rules = 0` is still required.

- Setting the environment variable `DM_DISABLE_UDEV`:

  ```
  export DM_DISABLE_UDEV=1
  ```

  This will also disable notifications from udev. In addition, all udev related settings from `/etc/lvm/lvm.conf` will be ignored.

5.8.1 Resizing a Logical Volume with Commands

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

The recommended way to grow or shrink a logical volume is to use the YaST Partitioner. When using YaST, the size of the file system in the volume will automatically be adjusted, too.

LVs can be extended or shrunk manually while they are being used, but this may not be true for a file system 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 file system afterward. For information about resizing file systems, see *Chapter 2, Resizing File Systems*.

Ensure that you use the right sequence when manually resizing an LV:

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

To extend the size of a logical volume:

1. Open a terminal.
2. If the logical volume contains an Ext2 or Ext4 file system, which do not support online growing, dismount it. In case it contains file systems that are hosted for a virtual machine (such as a Xen VM), shut down the VM first.
3. At the terminal prompt, enter the following command to grow the size of the logical volume:

   ```
   sudo lvextend -L +SIZE /dev/VG_NAME/LV_NAME
   ```

   For `SIZE`, specify the amount of space you want to add to the logical volume, such as 10 GB. Replace `/dev/VG_NAME/LV_NAME` with the Linux path to the logical volume, such as `/dev/LOCAL/DATA`. For example:

   ```
   tux > sudo lvextend -L +10GB /dev/vg1/v1
   ```
4. Adjust the size of the file system. See *Chapter 2, Resizing File Systems* for details.
5. In case you have dismounted the file system, mount it again.

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

```
sudo lvextend –L +10G /dev/LOCAL/DATA
sudo btrfs filesystem resize +10G /dev/LOCAL/DATA
```

To shrink the size of a logical volume:

1. Open a terminal.

2. If the logical volume does not contain a Btrfs file system, dismount it. In case it contains file systems that are hosted for a virtual machine (such as a Xen VM), shut down the VM first. Note that volumes with the XFS file system cannot be reduced in size.
3. Adjust the size of the file system. See *Chapter 2, Resizing File Systems* for details.
4. At the terminal prompt, enter the following command to shrink the size of the logical volume to the size of the file system:

   ```
   sudo lvreduce /dev/VG_NAME/LV_NAME
   ```
5. In case you have unmounted the file system, mount it again.

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

```
sudo btrfs filesystem resize -size 5G /dev/LOCAL/DATA
sudo lvreduce /dev/LOCAL/DATA
```

Tip: Resizing the Volume and the File System with a Single Command

Starting with SUSE Linux Enterprise Server 12 SP1, **`lvextend`**, **`lvresize`**, and **`lvreduce`** support the option `--resizefs` which will not only change the size of the volume, but will also resize the file system. Therefore the examples for **`lvextend`** and **`lvreduce`** shown above can alternatively be run as follows:

```
sudo lvextend --resizefs –L +10G /dev/LOCAL/DATA
sudo lvreduce  --resizefs -L -5G /dev/LOCAL/DATA
```

Note that the `--resizefs` is supported for the following file systems: ext2/3/4, reiserfs, Btrfs, XFS. Resizing Btrfs with this option is currently only available on SUSE Linux Enterprise Server, since it is not yet accepted upstream.

5.8.2 Dynamic Aggregation of LVM Metadata via `lvmetad`

Most LVM commands require an accurate view of the LVM metadata stored on the disk devices in the system. With the current LVM design, if this information is not available, LVM must scan all the physical disk devices in the system. This requires a significant amount of I/O operations in systems that have many disks. In case a disk fails to respond, LVM commands might run into a timeout while waiting for the disk.

Dynamic aggregation of LVM metadata via `lvmetad` provides a solution for this problem. The purpose of the `lvmetad` daemon is to eliminate the need for this scanning by dynamically aggregating metadata information each time the status of a device changes. These events are signaled to `lvmetad` by udev rules. If the daemon is not running, LVM performs a scan as it normally would do.

This feature is enabled by default. In case it is disabled on your system, proceed as follows to enable it:

1. Open a terminal.
2. Stop the `lvmetad` daemon:

```
sudo systemctl stop lvm2-lvmetad
```

3. Edit `/etc/lvm/lvm.conf` and set `use_lvmetad` to `1`:

```
use_lvmetad = 1
```

4. Restart the `lvmetad` daemon:

```
sudo systemctl start lvm2-lvmetad
```

5.8.3 Using LVM Cache Volumes

LVM supports the use of fast block devices (such as an SSD device) as write-back or write-through caches for large slower block devices. The cache logical volume type uses a small and fast LV to improve the performance of a large and slow LV.

To set up LVM caching, you need to create two logical volumes on the caching device. A large one is used for the caching itself, a smaller volume is used to store the caching metadata. These two volumes need to be part of the same volume group as the original volume. When these volumes are created, they need to be converted into a cache pool which needs to be attached to the original volume:

PROCEDURE 5.2: SETTING UP A CACHED LOGICAL VOLUME

1. Create the original volume (on a slow device) if not already existing.
2. Add the physical volume (from a fast device) to the same volume group the original volume is part of and create the cache data volume on the physical volume.

3. Create the cache metadata volume. The size should be 1/1000 of the size of the cache data volume, with a minimum size of 8 MB.
4. Combine the cache data volume and metadata volume into a cache pool volume:

```
lvconvert --type cache-pool --poolmetadata VOLUME_GROUP/
METADATA_VOLUME VOLUME_GROUP/CACHING_VOLUME
```

5. Attach the cache pool to the original volume:

```
lvconvert --type cache --cachepool VOLUME_GROUP/CACHING_VOLUME VOLUME_GROUP/
ORIGINAL_VOLUME
```

For more information on LVM caching, see the lvmcache(7) man page.

5.9 Tagging LVM2 Storage Objects

A tag is an unordered keyword or term assigned to the metadata of a storage object. Tagging allows you to classify collections of LVM storage objects in ways that you find useful by attaching an unordered list of tags to their metadata.

5.9.1 Using LVM2 Tags

After you tag the LVM2 storage objects, you can use the tags in commands to accomplish the following tasks:

- Select LVM objects for processing according to the presence or absence of specific tags.
- Use tags in the configuration file to control which volume groups and logical volumes are activated on a server.
- Override settings in a global configuration file by specifying tags in the command.

A tag can be used in place of any command line LVM object reference that accepts:

- a list of objects
- a single object as long as the tag expands to a single object

Replacing the object name with a tag is not supported everywhere yet. After the arguments are expanded, duplicate arguments in a list are resolved by removing the duplicate arguments, and retaining the first instance of each argument.

Wherever there might be ambiguity of argument type, you must prefix a tag with the commercial at sign (@) character, such as `@mytag`. Elsewhere, using the "@" prefix is optional.

5.9.2 Requirements for Creating LVM2 Tags

Consider the following requirements when using tags with LVM:

Supported Characters

An LVM tag word can contain the ASCII uppercase characters A to Z, lowercase characters a to z, numbers 0 to 9, underscore (_), plus (+), hyphen (-), and period (.). The word cannot begin with a hyphen. The maximum length is 128 characters.

Supported Storage Objects

You can tag LVM2 physical volumes, volume groups, logical volumes, and logical volume segments. PV tags are stored in its volume group's metadata. Deleting a volume group also deletes the tags in the orphaned physical volume. Snapshots cannot be tagged, but their origin can be tagged.

LVM1 objects cannot be tagged because the disk format does not support it.

5.9.3 Command Line Tag Syntax

`--addtag` *`TAG_INFO`*

Add a tag to (or *tag*) an LVM2 storage object. Example:

```
sudo vgchange --addtag @db1 vg1
```

`--deltag` *`TAG_INFO`*

Remove a tag from (or *untag*) an LVM2 storage object. Example:

```
sudo vgchange --deltag @db1 vg1
```

`--tag` *`TAG_INFO`*

Specify the tag to use to narrow the list of volume groups or logical volumes to be activated or deactivated.

Enter the following to activate the volume if it has a tag that matches the tag provided (example):

```
sudo lvchange -ay --tag @db1 vg1/vol2
```

5.9.4 Configuration File Syntax

The following sections show example configurations for certain use cases.

5.9.4.1 Enabling Host Name Tags in the `lvm.conf` File

Add the following code to the `/etc/lvm/lvm.conf` file to enable host tags that are defined separately on host in a `/etc/lvm/lvm_<HOSTNAME>.conf` file.

```
tags {
   # Enable hostname tags
   hosttags = 1
}
```

You place the activation code in the `/etc/lvm/lvm_<HOSTNAME>.conf` file on the host. See *Section 5.9.4.3, "Defining Activation"*.

5.9.4.2 Defining Tags for Host Names in the lvm.conf File

```
tags {

   tag1 { }
      # Tag does not require a match to be set.

   tag2 {
      # If no exact match, tag is not set.
      host_list = [ "hostname1", "hostname2" ]
   }
}
```

5.9.4.3 Defining Activation

You can modify the `/etc/lvm/lvm.conf` file to activate LVM logical volumes based on tags.

In a text editor, add the following code to the file:

```
  activation {
      volume_list = [ "vg1/lvol0", "@database" ]
  }
```

Replace `@database` with your tag. Use `"@*"` to match the tag against any tag set on the host.

The activation command matches against *VGNAME*, *VGNAME/LVNAME*, or @*TAG* set in the metadata of volume groups and logical volumes. A volume group or logical volume is activated only if a metadata tag matches. The default if there is no match, is not to activate.

If `volume_list` is not present and tags are defined on the host, then it activates the volume group or logical volumes only if a host tag matches a metadata tag.

If `volume_list` is defined, but empty, and no tags are defined on the host, then it does not activate.

If volume_list is undefined, it imposes no limits on LV activation (all are allowed).

5.9.4.4 Defining Activation in Multiple Host Name Configuration Files

You can use the activation code in a host's configuration file (`/etc/lvm/lvm_<HOST_TAG>.conf`) when host tags are enabled in the `lvm.conf` file. For example, a server has two configuration files in the `/etc/lvm/` directory:

`lvm.conf`
`lvm_<HOST_TAG>.conf`

At start-up, load the `/etc/lvm/lvm.conf` file, and process any tag settings in the file. If any host tags were defined, it loads the related `/etc/lvm/lvm_<HOST_TAG>.conf` file. When it searches for a specific configuration file entry, it searches the host tag file first, then the `lvm.conf` file, and stops at the first match. Within the `lvm_<HOST_TAG>.conf` file, use the reverse order that tags were set in. This allows the file for the last tag set to be searched first. New tags set in the host tag file will trigger additional configuration file loads.

5.9.5 Using Tags for a Simple Activation Control in a Cluster

You can set up a simple host name activation control by enabling the `hostname_tags` option in the `/etc/lvm/lvm.conf` file. Use the same file on every machine in a cluster so that it is a global setting.

1. In a text editor, add the following code to the `/etc/lvm/lvm.conf` file:

```
tags {
   hostname_tags = 1
}
```

2. Replicate the file to all hosts in the cluster.

3. From any machine in the cluster, add `db1` to the list of machines that activate `vg1/lvol2`:

```
sudo lvchange --addtag @db1 vg1/lvol2
```

4. On the `db1` server, enter the following to activate it:

```
sudo lvchange -ay vg1/vol2
```

5.9.6 Using Tags to Activate On Preferred Hosts in a Cluster

The examples in this section demonstrate two methods to accomplish the following:

- Activate volume group `vg1` only on the database hosts `db1` and `db2`.
- Activate volume group `vg2` only on the file server host `fs1`.
- Activate nothing initially on the file server backup host `fsb1`, but be prepared for it to take over from the file server host `fs1`.

5.9.6.1 Option 1: Centralized Admin and Static Configuration Replicated Between Hosts

In the following solution, the single configuration file is replicated among multiple hosts.

1. Add the `@database` tag to the metadata of volume group `vg1`. In a terminal, enter

```
sudo vgchange --addtag @database vg1
```

2. Add the `@fileserver` tag to the metadata of volume group `vg2`. In a terminal, enter

```
sudo vgchange --addtag @fileserver vg2
```

3. In a text editor, modify the `/etc/lvm/lvm.conf` file with the following code to define the `@database`, `@fileserver`, `@fileserverbackup` tags.

```
tags {
   database {
      host_list = [ "db1", "db2" ]
   }
   fileserver {
      host_list = [ "fs1" ]
   }
```

```
   fileserverbackup {
      host_list = [ "fsb1" ]
   }
}

activation {
   # Activate only if host has a tag that matches a metadata tag
   volume_list = [ "@*" ]
}
```

4. Replicate the modified `/etc/lvm/lvm.conf` file to the four hosts: `db1`, `db2`, `fs1`, and `fsb1`.

5. If the file server host goes down, `vg2` can be brought up on `fsb1` by entering the following commands in a terminal on any node:

```
sudo vgchange --addtag @fileserverbackup vg2
sudo vgchange -ay vg2
```

5.9.6.2 Option 2: Localized Admin and Configuration

In the following solution, each host holds locally the information about which classes of volume to activate.

1. Add the `@database` tag to the metadata of volume group `vg1`. In a terminal, enter

```
sudo vgchange --addtag @database vg1
```

2. Add the `@fileserver` tag to the metadata of volume group `vg2`. In a terminal, enter

```
sudo vgchange --addtag @fileserver vg2
```

3. Enable host tags in the `/etc/lvm/lvm.conf` file:

   a. In a text editor, modify the `/etc/lvm/lvm.conf` file with the following code to enable host tag configuration files.

```
tags {
   hosttags = 1
}
```

   b. Replicate the modified `/etc/lvm/lvm.conf` file to the four hosts: `db1`, `db2`, `fs1`, and `fsb1`.

4. On host `db1`, create an activation configuration file for the database host `db1`. In a text editor, create `/etc/lvm/lvm_db1.conf` file and add the following code:

```
activation {
   volume_list = [ "@database" ]
}
```

5. On host `db2`, create an activation configuration file for the database host `db2`. In a text editor, create `/etc/lvm/lvm_db2.conf` file and add the following code:

```
activation {
   volume_list = [ "@database" ]
}
```

6. On host fs1, create an activation configuration file for the file server host `fs1`. In a text editor, create `/etc/lvm/lvm_fs1.conf` file and add the following code:

```
activation {
   volume_list = [ "@fileserver" ]
}
```

7. If the file server host `fs1` goes down, to bring up a spare file server host fsb1 as a file server:

   a. On host `fsb1`, create an activation configuration file for the host `fsb1`. In a text editor, create `/etc/lvm/lvm_fsb1.conf` file and add the following code:

   ```
   activation {
      volume_list = [ "@fileserver" ]
   }
   ```

   b. In a terminal, enter one of the following commands:

   ```
   sudo vgchange -ay vg2
   sudo vgchange -ay @fileserver
   ```

6 LVM Volume Snapshots

A Logical Volume Manager (LVM) logical volume 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.

6.1 Understanding Volume Snapshots

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

Important: Mounting Volumes with Snapshots

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

LVM volume 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 back up 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 source logical 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 source block. As more blocks change from their original value on the source logical volume, the snapshot size grows.

When you are sizing the snapshot, consider how much data is expected to change on the source logical 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 source logical 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 source logical volume will change at least one time

before you delete the snapshot, then the snapshot volume should be at least as large as the source logical 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.

In LVM2, snapshots are read/write by default. When you write data directly to the snapshot, that block is marked in the exception table as used, and never gets copied from the source logical volume. You can mount the snapshot volume, and test application changes by writing data directly to the snapshot volume. You can easily discard the changes by dismounting the snapshot, removing the snapshot, and then remounting the source logical volume.

In a virtual guest environment, you can use the snapshot function for LVM logical volumes you create on the server's disks, as you would on a physical server.

In a virtual host environment, you can use the snapshot function to back up the virtual machine's storage back-end, or to test changes to a virtual machine image, such as for patches or upgrades, without modifying the source logical volume. The virtual machine must be using an LVM logical volume as its storage back-end, as opposed to using a virtual disk file. You can mount the LVM logical volume and use it to store the virtual machine image as a file-backed disk, or you can assign the LVM logical volume as a physical disk to write to it as a block device.

Beginning in SLES 11 SP3, an LVM logical volume snapshot can be thinly provisioned. Thin provisioning is assumed if you create a snapshot without a specified size. The snapshot is created as a thin volume that uses space as needed from a thin pool. A thin snapshot volume has the same characteristics as any other thin volume. You can independently activate the volume, extend the volume, rename the volume, remove the volume, and even snapshot the volume.

Important: Thinly Provisioned Volumes in a Cluster

To use thinly provisioned snapshots in a cluster, the source logical volume and its snapshots must be managed in a single cluster resource. This allows the volume and its snapshots to always be mounted exclusively on the same node.

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 source logical volume. When the snapshot is full, it is disabled, which prevents you from remounting the source logical volume.

If you create multiple snapshots for a source logical volume, remove the snapshots in a last created, first deleted order.

6.2 Creating Linux Snapshots with LVM

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

Open a terminal and enter

```
sudo lvcreate -s [-L <size>] -n SNAP_VOLUME SOURCE_VOLUME_PATH
```

If no size is specified, the snapshot is created as a thin snapshot.

For example:

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

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

6.3 Monitoring a Snapshot

Open a terminal and enter

```
sudo lvdisplay SNAP_VOLUME
```

For example:

```
tux > sudo 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
```

6.4 Deleting Linux Snapshots

Open a terminal and enter

```
sudo lvremove SNAP_VOLUME_PATH
```

For example:

```
sudo lvremove /dev/lvmvg/linux01-snap
```

6.5 Using Snapshots for Virtual Machines on a Virtual Host

Using an LVM logical volume for a virtual machine's back-end storage allows flexibility in administering the underlying device, such as making it easier to move storage objects, create snapshots, and back up data. You can mount the LVM logical volume and use it to store the virtual machine image as a file-backed disk, or you can assign the LVM logical volume as a physical disk to write to it as a block device. You can create a virtual disk image on the LVM logical volume, then snapshot the LVM.

You can leverage the read/write capability of the snapshot to create different instances of a virtual machine, where the changes are made to the snapshot for a particular virtual machine instance. You can create a virtual disk image on an LVM logical volume, snapshot the source logical volume, and modify the snapshot for a particular virtual machine instance. You can create another snapshot of the source logical volume, and modify it for a different virtual machine instance. The majority of the data for the different virtual machine instances resides with the image on the source logical volume.

You can also leverage the read/write capability of the snapshot to preserve the virtual disk image while testing patches or upgrades in the guest environment. You create a snapshot of the LVM volume that contains the image, and then run the virtual machine on the snapshot location. The source logical volume is unchanged, and all changes for that machine are written to the snapshot. To return to the source logical volume of the virtual machine image, you power off the virtual machine, then remove the snapshot from the source logical volume. To start over, you re-create the snapshot, mount the snapshot, and restart the virtual machine on the snapshot image.

The following procedure uses a file-backed virtual disk image and the Xen hypervisor. You can adapt the procedure in this section for other hypervisors that run on the SUSE Linux Enterprise platform, such as KVM. To run a file-backed virtual machine image from the snapshot volume:

1. Ensure that the source logical volume that contains the file-backed virtual machine image is mounted, such as at mount point `/var/lib/xen/images/<IMAGE_NAME>`.
2. Create a snapshot of the LVM logical volume with enough space to store the differences that you expect.

   ```
   sudo lvcreate -s -L 20G -n myvm-snap /dev/lvmvg/myvm
   ```

   If no size is specified, the snapshot is created as a thin snapshot.
3. Create a mount point where you will mount the snapshot volume.

   ```
   sudo mkdir -p /mnt/xen/vm/myvm-snap
   ```
4. Mount the snapshot volume at the mount point you created.

   ```
   sudo mount -t auto /dev/lvmvg/myvm-snap /mnt/xen/vm/myvm-snap
   ```
5. In a text editor, copy the configuration file for the source virtual machine, modify the paths to point to the file-backed image file on the mounted snapshot volume, and save the file such as `/etc/xen/myvm-snap.cfg`.
6. Start the virtual machine using the mounted snapshot volume of the virtual machine.

   ```
   sudo xm create -c /etc/xen/myvm-snap.cfg
   ```
7. (Optional) Remove the snapshot, and use the unchanged virtual machine image on the source logical volume.

   ```
   sudo unmount /mnt/xenvms/myvm-snap
   sudo lvremove -f /dev/lvmvg/mylvm-snap
   ```
8. (Optional) Repeat this process as desired.

6.6 Merging a Snapshot with the Source Logical Volume to Revert Changes or Roll Back to a Previous State

Snapshots can be useful if you need to roll back or restore data on a volume to a previous state. For example, you might need to revert data changes that resulted from an administrator error or a failed or undesirable package installation or upgrade.

You can use the `lvconvert --merge` command to revert the changes made to an LVM logical volume. The merging begins as follows:

- If both the source logical volume and snapshot volume are not open, the merge begins immediately.
- If the source logical volume or snapshot volume are open, the merge starts the first time either the source logical volume or snapshot volume are activated and both are closed.
- If the source logical volume cannot be closed, such as the root file system, the merge is deferred until the next time the server reboots and the source logical volume is activated.
- If the source logical volume contains a virtual machine image, you must shut down the virtual machine, deactivate the source logical volume and snapshot volume (by dismounting them in that order), and then issue the merge command. Because the source logical volume is automatically remounted and the snapshot volume is deleted when the merge is complete, you should not restart the virtual machine until after the merge is complete. After the merge is complete, you use the resulting logical volume for the virtual machine.

After a merge begins, the merge continues automatically after server restarts until it is complete. A new snapshot cannot be created for the source logical volume while a merge is in progress.

While the merge is in progress, reads or writes to the source logical volume are transparently redirected to the snapshot that is being merged. This allows users to immediately view and access the data as it was when the snapshot was created. They do not need to wait for the merge to complete.

When the merge is complete, the source logical volume contains the same data as it did when the snapshot was taken, plus any data changes made after the merge began. The resulting logical volume has the source logical volume's name, minor number, and UUID. The source logical volume is automatically remounted, and the snapshot volume is removed.

1. Open a terminal and enter

```
sudo lvconvert --merge  [-b] [-i SECONDS] [SNAP_VOLUME_PATH[...snapN]|@VOLUME_TAG]
```

You can specify one or multiple snapshots on the command line. You can alternatively tag multiple source logical volumes with the same volume tag then specify `@<VOLUME_TAG>` on the command line. The snapshots for the tagged volumes are merged to their respective source logical volumes. For information about tagging logical volumes, see *Section 5.9, "Tagging LVM2 Storage Objects"*.
The options include:

-b,
--background

Run the daemon in the background. This allows multiple specified snapshots to be merged concurrently in parallel.

-i,
--interval <*SECONDS*>

Report progress as a percentage at regular intervals. Specify the interval in seconds.

For more information about this command, see the **`lvconvert(8)`** man page.
For example:

```
sudo lvconvert --merge /dev/lvmvg/linux01-snap
```

This command merges `/dev/lvmvg/linux01-snap` into its source logical volume.

```
sudo lvconvert --merge @mytag
```

If `lvol1`, `lvol2`, and `lvol3` are all tagged with `mytag`, each snapshot volume is merged serially with its respective source logical volume; that is: `lvol1`, then `lvol2`, then `lvol3`. If the `--background` option is specified, the snapshots for the respective tagged logical volume are merged concurrently in parallel.

2. (Optional) If both the source logical volume and snapshot volume are open and they can be closed, you can manually deactivate and activate the source logical volume to get the merge to start immediately.

```
sudo umount ORIGINAL_VOLUME
sudo lvchange -an ORIGINAL_VOLUME
sudo lvchange -ay ORIGINAL_VOLUME
sudo mount ORIGINAL_VOLUME MOUNT_POINT
```

For example:

```
sudo umount /dev/lvmvg/lvol01
sudo lvchange -an /dev/lvmvg/lvol01
sudo lvchange -ay /dev/lvmvg/lvol01
sudo mount /dev/lvmvg/lvol01 /mnt/lvol01
```

3. (Optional) If both the source logical volume and snapshot volume are open and the source logical volume cannot be closed, such as the `root` file system, you can restart the server and mount the source logical volume to get the merge to start immediately after the restart.

III Software RAID

7 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: RAID on Cluster File Systems

Software RAID underneath clustered file systems needs to be set up using a cluster multi-device (Cluster MD). Refer to the High Availability documentation at https://documentation.suse.com/sle-ha/html/SLE-HA-all/cha-ha-cluster-md.html.

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*.

7.1 Understanding RAID Levels

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

7.1.1 RAID 0

This level improves the performance of your data access by spreading out blocks of each file across multiple disks. 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.

7.1.2 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 percent 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.

7.1.3 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.

7.1.4 RAID 4

Level 4 provides block-level striping 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.

7.1.5 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 when possible to avoid the risk of losing data.

7.1.6 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 disks 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 four 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. A RAID 6 configuration needs a considerable amount of CPU time and memory for write operations.

TABLE 7.1: COMPARISON OF RAID 5 AND RAID 6

Feature	RAID 5	RAID 6
Number of devices	N + 1, minimum of 3	N + 2, minimum of 4
Parity	Distributed, single	Distributed, dual
Performance	Medium impact on write and rebuild	More impact on sequential write than RAID 5
Fault-tolerance	Failure of one component device	Failure of two component devices

7.1.7 Nested and Complex RAID Levels

Other RAID levels have been developed, such as RAIDn, RAID 10, RAID 0 + 1, RAID 30, and RAID 50. Some are proprietary implementations created by hardware vendors. Examples for creating RAID 10 configurations can be found in *Chapter 9, Creating Software RAID 10 Devices*.

7.2 Soft RAID Configuration with YaST

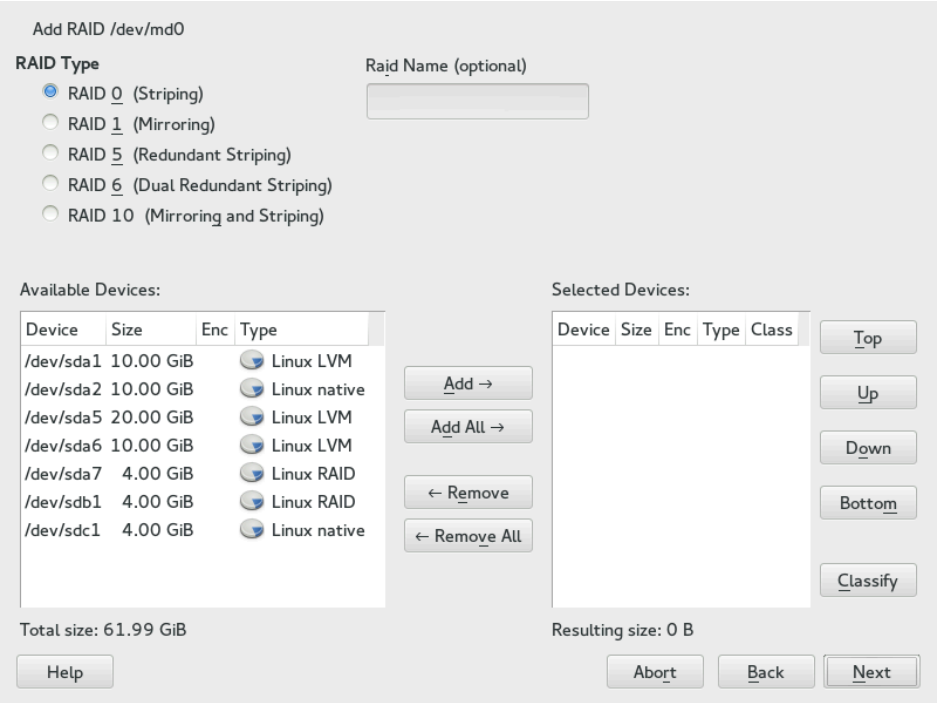
The YaST soft RAID configuration can be reached from the YaST Expert Partitioner. This partitioning tool also enables you to edit and delete existing partitions and create new ones that should be used with soft RAID. These instructions apply on setting up RAID levels 0, 1, 5, and 6. Setting up RAID 10 configurations is explained in *Chapter 9, Creating Software RAID 10 Devices*.

1. Launch YaST and open the *Partitioner*.

2. If necessary, create partitions that should be used with your RAID configuration. Do not format them and set the partition type to *0xFD Linux RAID*. When using existing partitions it is not necessary to change their partition type—YaST will automatically do so. Refer to *Book "Deployment Guide", Chapter 13 "Advanced Disk Setup", Section 13.1 "Using the YaST Partitioner"* for details.
   It is strongly recommended to use partitions 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.
   For RAID 0 at least two partitions are needed. RAID 1 requires exactly two partitions, while at least three partitions are required for RAID 5. A RAID 6 setup requires at least four partitions. 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.

3. In the left panel, select *RAID*.
   A list of existing RAID configurations opens in the right panel.

4. At the lower left of the RAID page, click *Add RAID*.



5. Select a *RAID Type* and *Add* an appropriate number of partitions from the *Available Devices* dialog.

You can optionally assign a *RAID Name* to your RAID. It will make it available as `/dev/md/NAME`. See *Section 7.2.1, "RAID Names"* for more information.

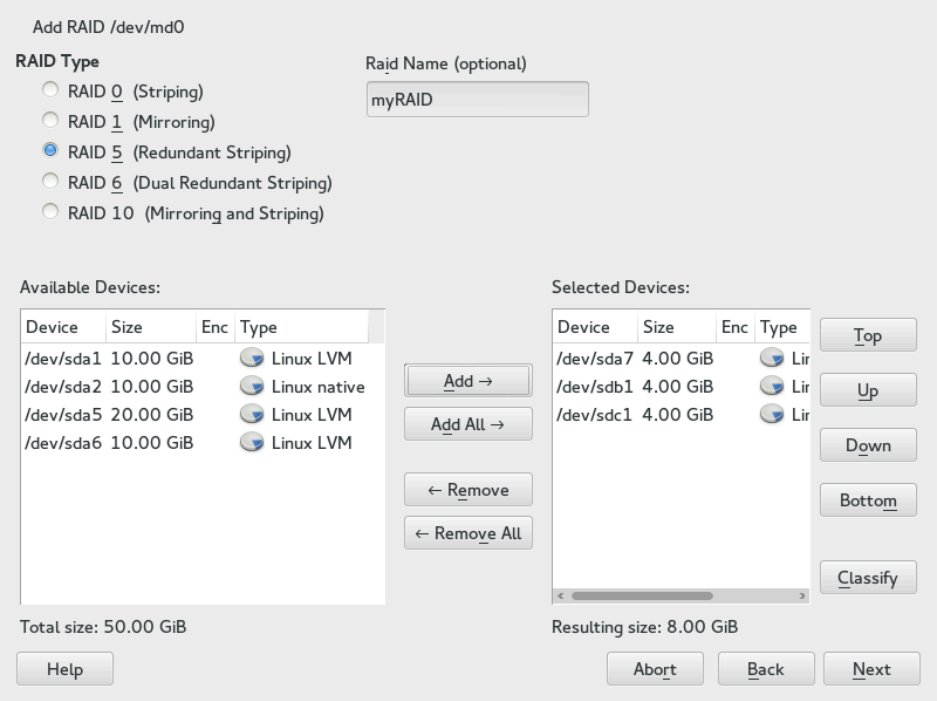


FIGURE 7.1: EXAMPLE RAID 5 CONFIGURATION

Proceed with *Next*.

6. Select the *Chunk Size* and, if applicable, the *Parity Algorithm*. The optimal chunk size depends on the type of data and the type of RAID. See https://raid.wiki.kernel.org/index.php/RAID_setup#Chunk_sizes for more information. More information on parity algorithms can be found with `man 8 mdadm` when searching for the `--layout` option. If unsure, stick with the defaults.

7. Choose a *Role* for the volume. Your choice here only affects the default values for the upcoming dialog. They can be changed in the next step. If in doubt, choose *Raw Volume (Unformatted)*.

8. Under *Formatting Options*, select *Format Partition*, then select the *File system*. The content of the *Options* menu depends on the file system. Usually there is no need to change the defaults.

   Under *Mounting Options*, select *Mount partition*, then select the mount point. Click *Fstab Options* to add special mounting options for the volume.

9. Click *Finish*.

10. Click *Next*, verify that the changes are listed, then click *Finish*.

7.2.1 RAID Names

By default, software RAID devices have numeric names following the pattern `mdN`, where `N` is a number. As such they can be accessed as, for example, `/dev/md127` and are listed as `md127` in `/proc/mdstat` and `/proc/partitions`. Working with these names can be clumsy. SUSE Linux Enterprise Server offers two ways to work around this problem:

Providing a Named Link to the Device

You can optionally specify a name for the RAID device when creating it with YaST or on the command line with **`mdadm --create '/dev/md/`** *`NAME`*`'`. The device name will still be `mdN`, but a link `/dev/md/`*`NAME`* will be created:

```
tux > ls -og /dev/md
total 0
lrwxrwxrwx 1 8 Dec  9 15:11 myRAID -> ../md127
```

The device will still be listed as `md127` under `/proc`.

Providing a Named Device

In case a named link to the device is not sufficient for your setup, add the line CREATE names=yes to `/etc/mdadm.conf` by running the following command:

```
tux > echo "CREATE names=yes" | sudo tee -a  /etc/mdadm.conf
```

It will cause names like `myRAID` to be used as a “real” device name. The device will not only be accessible at `/dev/myRAID`, but also be listed as `myRAID` under `/proc`. Note that this will only apply to RAIDs configured after the change to the configuration file. Active RAIDS will continue to use the `mdN` names until they get stopped and re-assembled.

Warning: Incompatible Tools

Not all tools may support named RAID devices. In case a tool expects a RAID device to be named `mdN`, it will fail to identify the devices.

7.3 Troubleshooting Software RAIDs

Check the `/proc/mdstat` file to find out whether a RAID partition has been damaged. If a disk fails, 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 (for all RAID levels except for RAID 0).

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

7.3.1 Recovery after Failing Disk is Back Again

There are several reasons a disk included in a RAID array may fail. Here is a list of the most common ones:

- Problems with the disk media.
- Disk drive controller failure.
- Broken connection to the disk.

In the case of the disk media or controller failure, the device needs to be replaced or repaired. If a hot-spare was not configured within the RAID, then manual intervention is required.

In the last case, the failed device can be automatically re-added by the **`mdadm`** command after the connection is repaired (which might be automatic).

Because **`md`** / **`mdadm`** cannot reliably determine what caused the disk failure, it assumes a serious disk error and treats any failed device as faulty until it is explicitly told that the device is reliable.

Under some circumstances—such as storage devices with the internal RAID array— the connection problems are very often the cause of the device failure. In such case, you can tell **`mdadm`** that it is safe to automatically `--re-add` the device after it appears. You can do this by adding the following line to `/etc/mdadm.conf`:

```
POLICY action=re-add
```

Note that the device will be automatically re-added after re-appearing only if the `udev` rules cause **`mdadm -I`** *`DISK_DEVICE_NAME`* to be run on any device that spontaneously appears (default behavior), and if write-intent bitmaps are configured (they are by default).

If you want this policy to only apply to some devices and not to the others, then the `path=` option can be added to the `POLICY` line in `/etc/mdadm.conf` to restrict the non-default action to only selected devices. Wild cards can be used to identify groups of devices. See **`man 5 mdadm.conf`** for more information.

7.4 For More Information

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

- *The Linux RAID wiki*: https://raid.wiki.kernel.org/
- *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* at http://marc.info/?l=linux-raid.

8 Configuring Software RAID for the Root Partition

In SUSE Linux Enterprise Server, 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. The `/boot` partition cannot be stored on a RAID partition unless it is RAID 1.

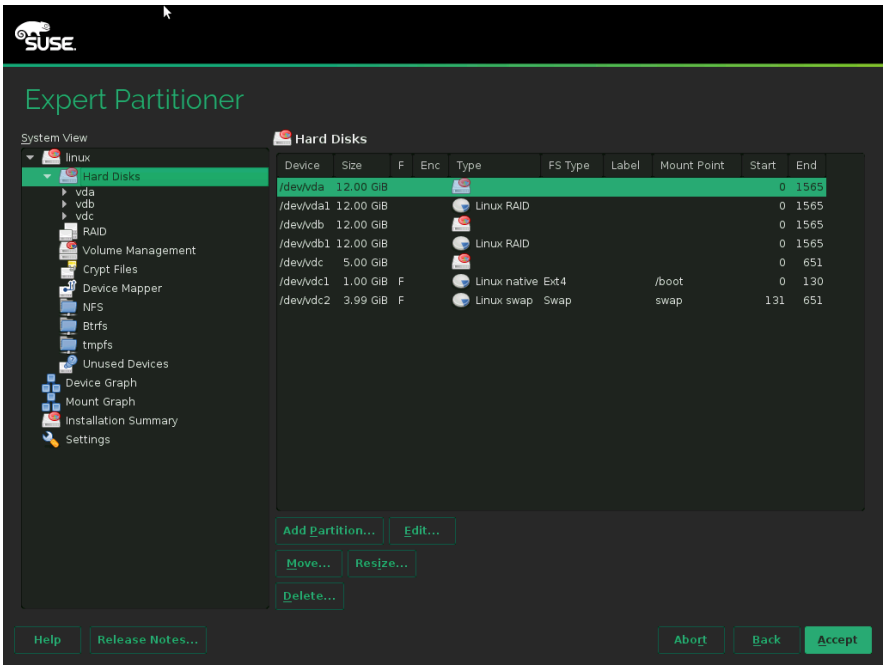
8.1 Prerequisites for Using a Software RAID Device for the Root Partition

Ensure that your configuration meets the following requirements:

- You need two hard disks to create the RAID 1 mirror device. The hard disks should be similarly sized. The RAID assumes the size of the smaller 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.
- A separate partition for `/boot` is not required if you install the boot loader in the MBR. If installing the boot loader in the MBR is not an option, `/boot` needs to reside on a separate partition.
- For UEFI machines, you need to set up a dedicated `/boot/efi` partition. It needs to be VFAT-formatted, and may reside on the RAID 1 device to prevent booting problems in case the physical disk with `/boot/efi` fails.
- 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, you need to 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 need to enable the multipath support before you create the RAID device.

8.2 Setting Up the System with a Software RAID Device for the Root (/) Partition

1. Start the installation with YaST and proceed as described in *Book "Deployment Guide", Chapter 6 "Installation with YaST"* until you reach the *Suggested Partitioning* step.
2. Click *Expert Partitioner* to open the custom partitioning tool.
3. (Optional) If there are iSCSI target devices that you want to use, you need to enable the iSCSI Initiator software by choosing *Configure* › *Configure iSCSI* from the lower right section of the screen. Refer to *Chapter 14, Mass Storage over IP Networks: iSCSI* for further details.
4. (Optional) If there are multiple I/O paths to the devices that you want to use you need to enable multipath support by choosing *Configure* › *Configure Multipath* › *Yes* from the lower right section of the screen.
5. (Optional) In case you have neither configured iSCSI or Multipath, the default proposal settings are shown. Click *Rescan Devices* to delete them.
6. Set up the *0xFD Linux RAID* format for each of the devices you want to use for the software RAID. You should use RAID for `/`, `/boot/efi`, or swap partitions.
    a. In the left panel, select *Hard Disks* and select the device you want to use, then click *Add Partition*.
    b. Under *New Partition Type*, select *Primary Partition*, then click *Next*.
    c. Under *New Partition Size*, specify the size to use, then click *Next*.
    d. Under *Role*, choose *Raw Volume (unformatted)*.
    e. Select *Do not format* and set the *File SystemID* to *0xFD Linux RAID*.
    f. Click *Finish* and repeat these instructions for the second partition.

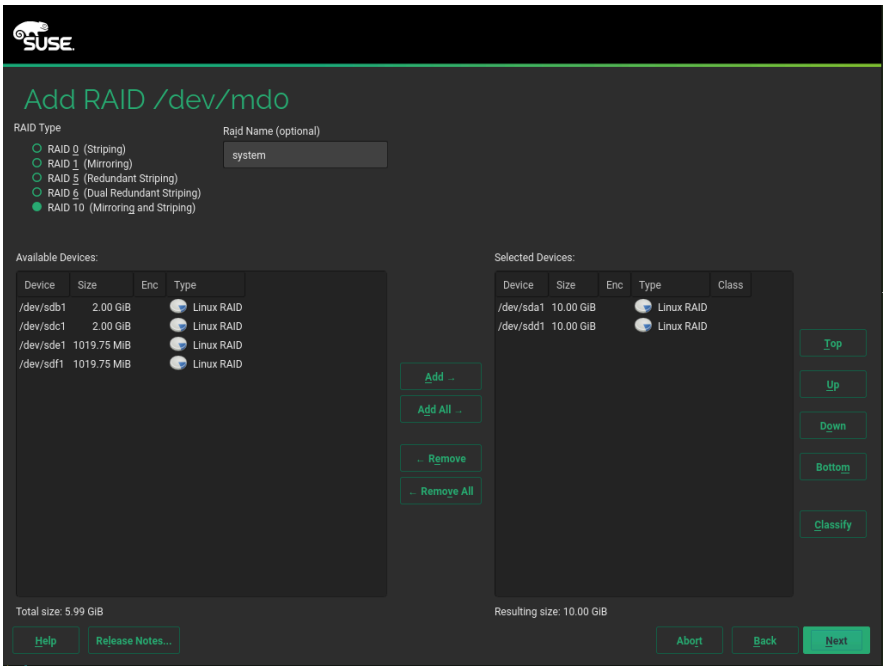


7. Create the RAID device for the `/` partition.

   a. In the left panel, select *RAID* and then *Add RAID*.

   b. Set the desired *RAID Type* for the `/` partition and the *RAID name* to `system`.

   c. Select the two RAID devices you prepared in the previous step from the *Available Devices* section and *Add* them.



Proceed with *Next*.

d. Under *RAID Options*, select the chunk size from the drop-down box. Sticking with the default is a safe choice.

e. Under *Role*, select *Operating System* and proceed with *Finish*.

f. Select the *File System* and set the mount point to `/`. Leave the dialog with `Finish`.

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

9. Optionally for UEFI machines, use similar steps to create the `/boot/efi` mounted partition. Remember that only RAID 1 is supported for `/boot/efi`, and the partition needs to be formatted with the FAT file system.

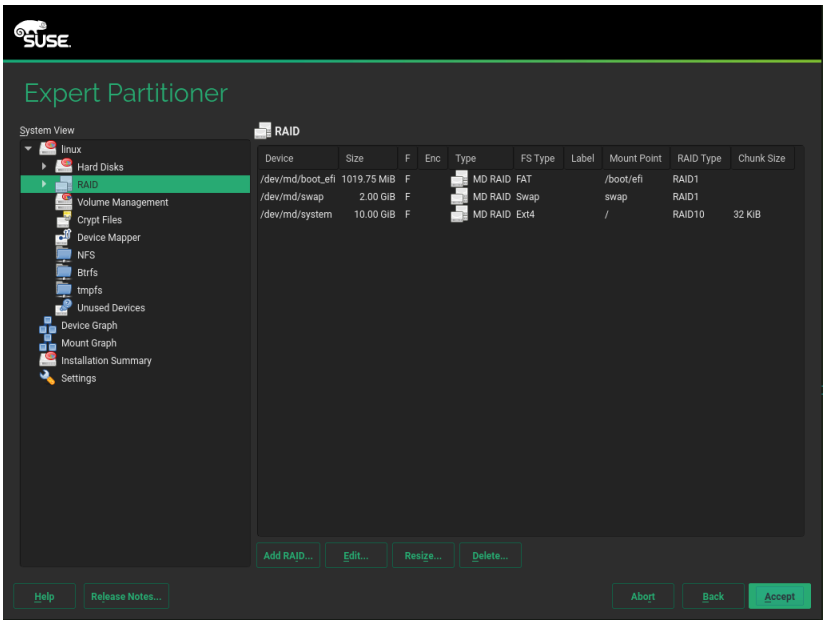


FIGURE 8.1: */*, */BOOT/EFI*, AND SWAP ON RAIDS

10. Click *Accept* to leave the partitioner.
    The new proposal appears on the *Suggested Partitioning* page.

11. Continue with the installation. For UEFI machines with a separate `/boot/efi` partition, click *Booting* on the *Installation Settings* screen and set *GRUB2 for EFI* as the *Boot Loader*. Check that the *Enable Secure Boot Support* option is activated.
    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.

9 Creating Software RAID 10 Devices

This section describes how to set up nested and complex RAID 10 devices. A RAID 10 device consists of nested RAID 1 (mirroring) and RAID 0 (striping) arrays. Nested RAIDs can either be set up as striped mirrors (RAID 1 + 0) or as mirrored stripes (RAID 0 + 1). A complex RAID 10 setup also combines mirrors and stripes and additional data security by supporting a higher data redundancy level.

9.1 Creating Nested RAID 10 Devices with `mdadm`

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. Setting up nested RAID levels is not supported by YaST, but can be done by using the `mdadm` command line tool.

Based on the order of nesting, two different nested RAIDs can be set up. 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 9.1: NESTED RAID LEVELS

RAID Level	Description	Performance and Fault Tolerance
10 (1 + 0)	RAID 0 (stripe) built with RAID 1 (mirror) arrays	RAID 1 + 0 provides high levels of I/O performance, data redundancy, and disk fault tolerance. Because each member device in the RAID 0 is mirrored individually, multiple disk failures can be tolerated and data remains available as long as the disks that fail are in different mirrors.

RAID Level	Description	Performance and Fault Tolerance
		You can optionally configure a spare for each underlying mirrored array, or configure a spare to serve a spare group that serves all mirrors.
10 (0+1)	RAID 1 (mirror) built with RAID 0 (stripe) arrays	RAID 0+1 provides high levels of I/O performance and data redundancy, but slightly less fault tolerance than a 1+0. If multiple disks fail on one side of the mirror, then the other mirror is available. However, if disks are lost concurrently on both sides of the mirror, all data is lost. This solution offers less disk fault tolerance than a 1+0 solution, but if you need to perform maintenance or maintain the mirror on a different site, you can take an entire side of the mirror offline and still have a fully functional storage device. Also, if you lose the connection between the two sites, either site operates independently of the other. That is not true if you stripe the mirrored segments, because the mirrors are managed at a lower level. If a device fails, the mirror on that side fails because RAID 1 is not fault-tolerant. Create a new RAID 0 to replace the failed side, then resynchronize the mirrors.

9.1.1 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: Multipathing

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 17, Managing multipath I/O for devices*.

The procedure in this section uses the device names shown in the following table. Ensure that you modify the device names with the names of your own devices.

TABLE 9.2: 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/sde1`	`/dev/md1`	

1. Open a terminal.
2. If necessary, create four 0xFD Linux RAID partitions of equal size using a disk partitioner such as parted.
3. Create two software RAID 1 devices, using two different devices for each device. At the command prompt, enter these two commands:

```
sudo mdadm --create /dev/md0 --run --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
sudo mdadm --create /dev/md1 --run --level=1 --raid-devices=2 /dev/sdd1 /dev/sde1
```

4. Create the nested RAID 1 + 0 device. At the command prompt, enter the following command using the software RAID 1 devices you created in the previous step:

```
sudo mdadm --create /dev/md2 --run --level=0 --chunk=64 \
--raid-devices=2 /dev/md0 /dev/md1
```

   The default chunk size is 64 KB.

5. Create a file system on the RAID 1 + 0 device `/dev/md2`, for example an XFS file system:

```
sudo mkfs.xfs /dev/md2
```

Modify the command to use a different file system.

6. Edit the `/etc/mdadm.conf` file or create it, if it does not exist (for example by running `sudo vi /etc/mdadm.conf`). Add the following lines (if the file already exists, the first line probably already exists).

```
DEVICE containers partitions
ARRAY /dev/md0 UUID=UUID
ARRAY /dev/md1 UUID=UUID
ARRAY /dev/md2 UUID=UUID
```

   The UUID of each device can be retrieved with the following command:

```
sudo mdadm -D /dev/DEVICE | grep UUID
```

7. Edit the `/etc/fstab` file to add an entry for the RAID 1+0 device `/dev/md2`. The following example shows an entry for a RAID device with the XFS file system and `/data` as a mount point.

```
/dev/md2 /data xfs defaults 1 2
```

8. Mount the RAID device:

```
sudo mount /data
```

9.1.2 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: Multipathing

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 17, Managing multipath I/O for devices*.

In this configuration, spare devices cannot be specified for the underlying RAID 0 devices because RAID 0 cannot tolerate a device loss. If a device fails on one side of the mirror, you must create a replacement RAID 0 device, than add it into the mirror.

The procedure in this section uses the device names shown in the following table. Ensure that you modify the device names with the names of your own devices.

TABLE 9.3: 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.
2. If necessary, create four 0xFD Linux RAID partitions of equal size using a disk partitioner such as parted.
3. Create two software RAID 0 devices, using two different devices for each RAID 0 device. At the command prompt, enter these two commands:

   ```
   sudo mdadm --create /dev/md0 --run --level=0 --chunk=64 \
   --raid-devices=2 /dev/sdb1 /dev/sdc1
   sudo mdadm --create /dev/md1 --run --level=0 --chunk=64 \
   --raid-devices=2 /dev/sdd1 /dev/sde1
   ```

   The default chunk size is 64 KB.
4. Create the nested RAID 0 + 1 device. At the command prompt, enter the following command using the software RAID 0 devices you created in the previous step:

   ```
   sudo mdadm --create /dev/md2 --run --level=1 --raid-devices=2 /dev/md0 /dev/md1
   ```
5. Create a file system on the RAID 1 + 0 device `/dev/md2`, for example an XFS file system:

   ```
   sudo mkfs.xfs /dev/md2
   ```

   Modify the command to use a different file system.
6. Edit the `/etc/mdadm.conf` file or create it, if it does not exist (for example by running **`sudo vi /etc/mdadm.conf`**). Add the following lines (if the file exists, the first line probably already exists, too).

   ```
   DEVICE containers partitions
   ```

```
ARRAY /dev/md0 UUID=UUID
ARRAY /dev/md1 UUID=UUID
ARRAY /dev/md2 UUID=UUID
```

The UUID of each device can be retrieved with the following command:

```
sudo mdadm -D /dev/DEVICE | grep UUID
```

7. Edit the `/etc/fstab` file to add an entry for the RAID 1+0 device `/dev/md2`. The following example shows an entry for a RAID device with the XFS file system and `/data` as a mount point.

```
/dev/md2 /data xfs defaults 1 2
```

8. Mount the RAID device:

```
sudo mount /data
```

9.2 Creating a Complex RAID 10

YaST (and **`mdadm`** with the `--level=10` option) creates a single complex software RAID 10 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.

The complex RAID 10 is similar in purpose to a nested RAID 10 (1+0), but differs in the following ways:

TABLE 9.4: COMPLEX RAID 10 COMPARED TO NESTED RAID 10

Feature	Complex RAID 10	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	Complex RAID 10	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.

9.2.1 Number of Devices and Replicas in the Complex RAID 10

When configuring a complex RAID 10 array, you must specify the number of replicas of each data block that are required. The default number of replicas is two, but the value can be two to the number of devices in the array.

You must use at least as many component devices as the number of replicas you specify. However, the number of component devices in a RAID 10 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 two replicas for an array created with five 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.

9.2.2 Layout

The complex RAID 10 setup supports three different layouts which define how the data blocks are arranged on the disks. The available layouts are near (default), far and offset. They have different performance characteristics, so it is important to choose the right layout for your workload.

9.2.2.1 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 RAID 10. 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 complex RAID 10 yields read and write performance similar to RAID 0 over half the number of drives.

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

```
sda1 sdb1 sdc1 sde1
  0    0    1    1
  2    2    3    3
  4    4    5    5
  6    6    7    7
  8    8    9    9
```

Near layout with an odd number of disks and two replicas:

```
sda1 sdb1 sdc1 sde1 sdf1
  0    0    1    1    2
  2    3    3    4    4
  5    5    6    6    7
  7    8    8    9    9
  10   10   11   11   12
```

9.2.2.2 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 complex RAID 10 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 RAID 10 for writing is similar to other mirrored RAID types, like RAID 1 and RAID 10 using near layout, as the elevator of the file system schedules the writes in a more optimal way than raw writing. Using RAID 10 in the far layout is 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
  4    5    6    7
  . . .
  3    0    1    2
  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
```

9.2.2.3 Offset Layout

The offset layout duplicates stripes so that the multiple copies of a given chunk are laid out on consecutive drives and at consecutive offsets. Effectively, each stripe is duplicated and the copies are offset by one device. This should give similar read characteristics to a far layout if a suitably large chunk size is used, but without as much seeking for writes.

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

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

```
 11    8    9   10
```

Offset layout with an odd number of disks and two replicas:

```
sda1 sdb1 sdc1 sde1 sdf1
  0    1    2    3    4
  4    0    1    2    3
  5    6    7    8    9
  9    5    6    7    8
 10   11   12   13   14
 14   10   11   12   13
```

9.2.2.4 Specifying the number of Replicas and the Layout with YaST and mdadm

The number of replicas and the layout is specified as *Parity Algorithm* in YaST or with the `--layout` parameter for mdadm. The following values are accepted:

`n`*`N`*

Specify `n` for near layout and replace *`N`* with the number of replicas. `n2` is the default that is used when not configuring layout and the number of replicas.

`f`*`N`*

Specify `f` for far layout and replace *`N`* with the number of replicas.

`o`*`N`*

Specify `o` for offset layout and replace *`N`* with the number of replicas.

Note: Number of Replicas

YaST automatically offers a selection of all possible values for the *Parity Algorithm* parameter.

9.2.3 Creating a Complex RAID 10 with the YaST Partitioner

1. Launch YaST and open the Partitioner.

2. If necessary, create partitions that should be used with your RAID configuration. Do not format them and set the partition type to *0xFD Linux RAID*. When using existing partitions it is not necessary to change their partition type—YaST will automatically do so. Refer to *Book "Deployment Guide", Chapter 13 "Advanced Disk Setup", Section 13.1 "Using the YaST Partitioner"* for details.

   For RAID 10 at least four partitions are needed. It is strongly recommended to use partitions stored on different hard disks to decrease the risk of losing data if one is defective. 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.

3. In the left panel, select *RAID*.

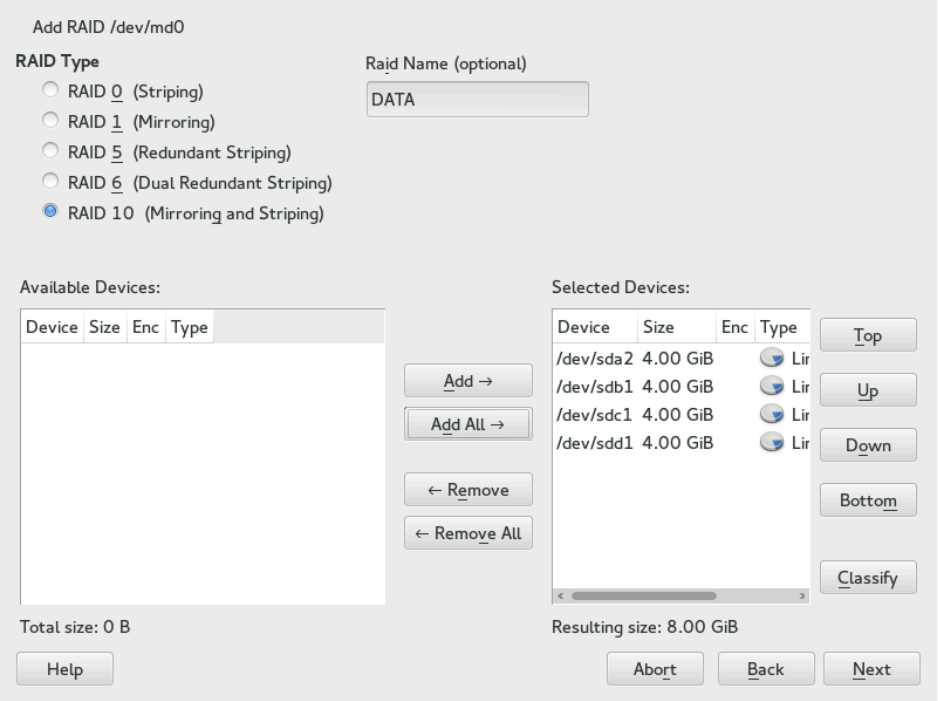
   A list of existing RAID configurations opens in the right panel.

4. At the lower left of the RAID page, click *Add RAID*.

5. Under *RAID Type*, select *RAID 10 (Mirroring and Striping)*.

   You can optionally assign a *RAID Name* to your RAID. It will make it available as `/dev/md/NAME`. See *Section 7.2.1, "RAID Names"* for more information.

6. In the *Available Devices* list, select the desired partitions, then click *Add* to move them to the *Selected Devices* list.



7. (Optional) Click *Classify* to specify the preferred order of the disks in the RAID array.

For RAID types such as RAID 10, where the order of added disks matters, you can specify the order in which the devices will be used. This will ensure that one half of the array resides on one disk subsystem and the other half of the array resides on a different disk subsystem. For example, if one disk subsystem fails, the system keeps running from the second disk subsystem.

a. Select each disk in turn and click one of the *Class X* buttons, where X is the letter you want to assign to the disk. Available classes are A, B, C, D and E but for many cases fewer classes are needed (only A and B, for example). Assign all available RAID disks this way.
   You can press the `Ctrl` or `Shift` key to select multiple devices. You can also right-click a selected device and choose the appropriate class from the context menu.

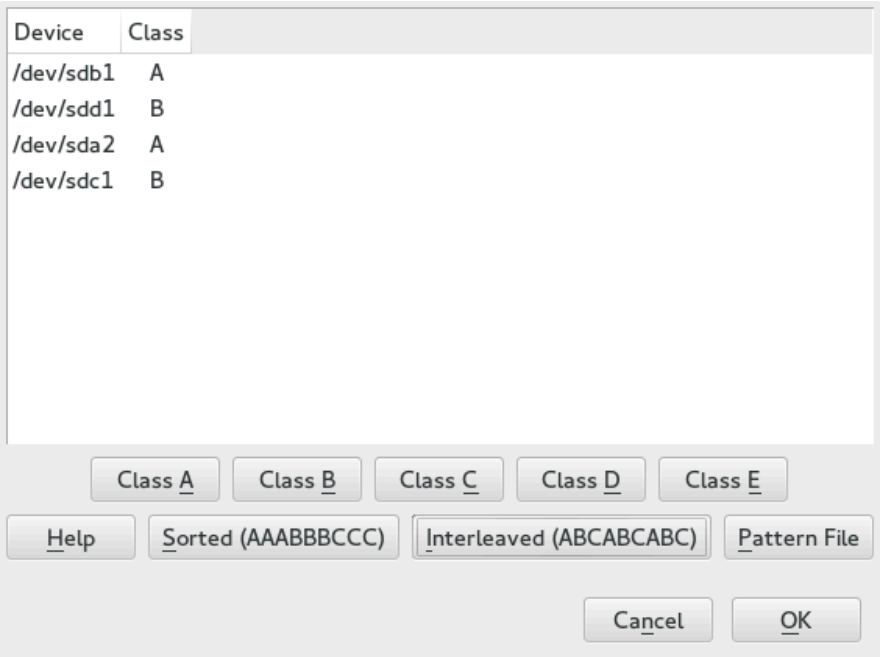
b. Specify the order of the devices by selecting one of the sorting options:

   Sorted: Sorts all devices of class A before all devices of class B and so on. For example: `AABBCC`.

   Interleaved: Sorts devices by the first device of class A, then first device of class B, then all the following classes with assigned devices. Then the second device of class A, the second device of class B, and so on follows. All devices without a class are sorted to the end of the devices list. For example: `ABCABC`.

   Pattern File: Select an existing file that contains multiple lines, where each is a regular expression and a class name (`"sda.* A"`). All devices that match the regular expression are assigned to the specified class for that line. The regular expression is matched against the kernel name (`/dev/sda1`), the udev path name (`/dev/disk/by-path/pci-0000:00:1f.2-scsi-0:0:0:0-part1`) and then the udev ID (`dev/disk/by-id/ata-ST3500418AS_9VMN8X8L-part1`). The first match made determines the class if a device's name matches more than one regular expression.

c. At the bottom of the dialog, click *OK* to confirm the order.



8. Click *Next*.
9. Under *RAID Options*, specify the *Chunk Size* and *Parity Algorithm*, then click *Next*.
   For a RAID 10, the parity options are n (near), f (far), and o (offset). The number indicates the number of replicas of each data block that are required. Two is the default. For information, see *Section 9.2.2, "Layout"*.
10. Add a file system and mount options to the RAID device, then click *Finish*.
11. Click *Next*.
12. Verify the changes to be made, then click *Finish* to create the RAID.

9.2.4 Creating a Complex RAID 10 with mdadm

The procedure in this section uses the device names shown in the following table. Ensure that you modify the device names with the names of your own devices.

TABLE 9.5: SCENARIO FOR CREATING A RAID 10 USING MDADM

Raw Devices	RAID 10
`/dev/sdf1`	`/dev/md3`

Raw Devices	RAID 10

`/dev/sdg1`
`/dev/sdh1`
`/dev/sdi1`	

1. Open a terminal.
2. If necessary, create at least four 0xFD Linux RAID partitions of equal size using a disk partitioner such as parted.
3. Create a RAID 10 by entering the following command.

   ```
   mdadm --create /dev/md3 --run --level=10 --chunk=32 --raid-devices=4 \
   /dev/sdf1 /dev/sdg1 /dev/sdh1 /dev/sdi1
   ```

   Make sure to adjust the value for `--raid-devices` and the list of partitions according to your setup.
   The command creates an array with near layout and two replicas. To change any of the two values, use the `--layout` as described in *Section 9.2.2.4, "Specifying the number of Replicas and the Layout with YaST and mdadm"*.
4. Create a file system on the RAID 10 device `/dev/md3`, for example an XFS file system:

   ```
   sudo mkfs.xfs /dev/md3
   ```

   Modify the command to use a different file system.
5. Edit the `/etc/mdadm.conf` file or create it, if it does not exist (for example by running **`sudo vi /etc/mdadm.conf`**). Add the following lines (if the file exists, the first line probably already exists, too) .

   ```
   DEVICE containers partitions
   ARRAY /dev/md3 UUID=UUID
   ```

   The UUID of the device can be retrieved with the following command:

   ```
   sudo mdadm -D /dev/md3 | grep UUID
   ```
6. Edit the `/etc/fstab` file to add an entry for the RAID 10 device `/dev/md3`. The following example shows an entry for a RAID device with the XFS file system and `/data` as a mount point.

```
/dev/md3 /data xfs defaults 1 2
```

7. Mount the RAID device:

```
sudo mount /data
```

10 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. To 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 the device you copied the data from 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
   ```

11 Resizing Software RAID Arrays with mdadm

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.

Resizing an existing software RAID device involves increasing or decreasing the space contributed by each component partition. The file system that resides on the RAID must also be able to be resized to take advantage of the changes in available space on the device. In SUSE Linux Enterprise Server, file system resizing utilities are available for file systems Btrfs, Ext2, Ext3, Ext4, ReiserFS, and XFS (increase size only). Refer to *Chapter 2, Resizing File Systems* for more information.

The `mdadm` 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.

Warning: Back Up your Data Before Resizing

Resizing any partition or file system involves some risks that can potentially result in losing data. To avoid data loss, ensure that you back up your data before you begin any resizing task.

Resizing the RAID involves the following tasks. The order in which these tasks are performed depends on whether you are increasing or decreasing its size.

TABLE 11.1: TASKS INVOLVED IN RESIZING A RAID

Tasks	Description	Order If Increasing Size	Order If Decreasing Size
Resize each of the component partitions.	Increase or decrease the active size of each component partition. You remove only one component partition at a time, modify its size, then return it to the RAID.	1	2

Tasks	Description	Order If Increasing Size	Order If Decreasing Size
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.	3	1

The procedures in the following sections use the device names shown in the following table. Ensure that you modify the names to use the names of your own devices.

TABLE 11.2: SCENARIO FOR INCREASING THE SIZE OF COMPONENT PARTITIONS

RAID Device	Component Partitions

`/dev/md0`	`/dev/sda1`
	`/dev/sdb1`
	`/dev/sdc1`

11.1 Increasing the Size of a Software RAID

Increasing the size of a software RAID involves the following tasks in the given order: increase the size of all partitions the RAID consists of, increase the size of the RAID itself and, finally, increase the size of the file system.

Warning: Potential Data Loss

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 ensure that you have a backup of your data available.

11.1.1 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. To increase the size of the component partitions for the RAID, proceed as follows:

1. Open a terminal.
2. Ensure 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

   ```
   sudo mdadm /dev/md0 --fail /dev/sda1 --remove /dev/sda1
   ```

   To succeed, both the fail and remove actions must be specified.
4. Increase the size of the partition that you removed in the previous step by doing one of the following:

   - Increase the size of the partition, using a disk partitioner such as the YaST Partitioner or the command line tool parted. 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

   ```
   sudo mdadm -a /dev/md0 /dev/sda1
   ```

   Wait until the RAID is synchronized and consistent before continuing with the next partition.

6. Repeat these steps for each of the remaining component devices in the array. Ensure that you 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 11.1.2, "Increasing the Size of the RAID Array"*.

Alternatively, if you are replacing the disks and can install the new disks temporarily alongside the existing array, you can hot-replace the partitions. This will keep them in service until a new partition has been rebuilt as a spare, so the array does not enter a degraded state and remains fault-tolerant during the process. The following steps replace steps 3–5 in the above procedure.

1. Mark a component partition for replacement. For example, to replace `/dev/sda1`, enter

```
tux > sudo mdadm /dev/md0 --replace /dev/sda1
```

2. Add a replacement partition to the RAID array. For example, to add `/dev/sdd1`, enter

```
tux > sudo mdadm -a /dev/md0 /dev/sdd1
```

3. Once the new partition has been added and has finished rebuilding, the partition marked for replacement will be automatically marked as faulty, and can be removed from the array. For example, to remove `/dev/sda1`, enter

```
tux > sudo mdadm /dev/md0 --remove /dev/sda1
```

11.1.2 Increasing the Size of the RAID Array

After you have resized each of the component partitions in the RAID (see *Section 11.1.1, "Increasing the Size of Component Partitions"*), 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. Ensure that you modify the name to use the name of your own device.

1. Open a terminal.
2. Ensure 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. Check the size of the array and the device size known to the array by entering

```
sudo mdadm -D /dev/md0 | grep -e "Array Size" -e "Dev Size"
```

4. Do one of the following:

   - Increase the size of the array to the maximum available size by entering

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

   - Increase the size of the array to the maximum available size by entering

     ```
     sudo mdadm --grow /dev/md0 -z max --assume-clean
     ```

     The array uses any space that has been added to the devices, but this space will not be synchronized. This is recommended for RAID 1 because the synchronization is not needed. It can be useful for other RAID levels if the space that was added to the member devices was pre-zeroed.

   - Increase the size of the array to a specified value by entering

     ```
     sudo mdadm --grow /dev/md0 -z SIZE
     ```

     Replace *SIZE* with an integer value in kilobytes (a kilobyte is 1024 bytes) for the desired size.

5. Recheck the size of your array and the device size known to the array by entering

```
sudo mdadm -D /dev/md0 | grep -e "Array Size" -e "Dev Size"
```

6. Do one of the following:

   - If your array was successfully resized, continue with *Section 11.1.3, "Increasing the Size of the File System"*.

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

11.1.3 Increasing the Size of the File System

After you increase the size of the array (see *Section 11.1.2, "Increasing the Size of the RAID Array"*), 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, ensure that 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.

Refer to *Chapter 2, Resizing File Systems* for detailed instructions.

11.2 Decreasing the Size of a Software RAID

Decreasing the Size of a Software RAID involves the following tasks in the given order: decrease the size of the file system, decrease the size of all partitions the RAID consists of, and finally decrease the size of the RAID itself.

Warning: Potential Data Loss

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 ensure that you have a backup of your data available.

Important: XFS

Decreasing the size of a file system formatted with XFS is not possible, since such a feature is not supported by XFS. As a consequence, the size of a RAID that uses the XFS file system cannot be decreased.

11.2.1 Decreasing the Size of the File System

When decreasing the size of the file system on a RAID device, ensure that 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.

Refer to *Chapter 2, Resizing File Systems* for detailed instructions.

11.2.2 Decreasing the Size of the RAID Array

After you have resized the file system (see *Section 11.2.1, "Decreasing the Size of the File System"*), the RAID array configuration continues to use the original array size until you force it to reduce the available space. Use the **`mdadm --grow`** mode to force the RAID to use a smaller segment size. To do this, you must use the -z option to specify the amount of space in kilobytes to use from each device in the RAID. This size must be a multiple of the chunk size, and it must leave about 128 KB of space for the RAID superblock to be written to the device.

The procedure in this section uses the device name `/dev/md0` for the RAID device. Ensure that you modify commands to use the name of your own device.

1. Open a terminal.
2. Check the size of the array and the device size known to the array by entering

   ```
   sudo mdadm -D /dev/md0 | grep -e "Array Size" -e "Dev Size"
   ```

3. Decrease the array's device size to a specified value by entering

   ```
   sudo mdadm --grow /dev/md0 -z SIZE
   ```

   Replace *`SIZE`* with an integer value in kilobytes for the desired size. (A kilobyte is 1024 bytes.)

   For example, the following command sets the segment size for each RAID device to about 40 GB where the chunk size is 64 KB. It includes 128 KB for the RAID superblock.

   ```
   sudo mdadm --grow /dev/md2 -z 41943168
   ```

4. Recheck the size of your array and the device size known to the array by entering

```
sudo 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, "Decreasing the Size of Component Partitions"*.
   - If your array was not resized as you expected, you must reboot, then try this procedure again.

11.2.3 Decreasing the Size of Component Partitions

After you decrease the segment size that is used on each device in the RAID (see *Section 11.2.2, "Decreasing the Size of the RAID Array"*), the remaining space in each component partition is not used by the RAID. You can leave partitions at their current size to allow for the RAID to grow at a future time, or you can reclaim this now unused space.

To reclaim the space, you decrease the component partitions one at a time. For each component partition, you remove it from the RAID, reduce its partition size, return the partition to the RAID, then wait until the RAID stabilizes. To allow for metadata, you should specify a slightly larger size than the size you specified for the RAID in *Section 11.2.2, "Decreasing the Size of the RAID Array"*.

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. To decrease the size of the component partitions for the RAID, proceed as follows:

1. Open a terminal.
2. Ensure 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

   ```
   sudo mdadm /dev/md0 --fail /dev/sda1 --remove /dev/sda1
   ```

To succeed, both the fail and remove actions must be specified.

4. Decrease the size of the partition that you removed in the previous step to a size that is slightly larger than the size you set for the segment size. The size should be a multiple of the chunk size and allow 128 KB for the RAID superblock. Use a disk partitioner such as the YaST partitioner or the command line tool parted to decrease the size of the partition.

5. Re-add the partition to the RAID array. For example, to add `/dev/sda1`, enter

   ```
   sudo mdadm -a /dev/md0 /dev/sda1
   ```

   Wait until the RAID is synchronized and consistent before continuing with the next partition.

6. Repeat these steps for each of the remaining component devices in the array. Ensure that you 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. (Optional) Expand the size of the RAID and file system to use the maximum amount of space in the now smaller component partitions and increase the size of the file system afterward. Refer to *Section 11.1.2, "Increasing the Size of the RAID Array"* for instructions.

12 Storage Enclosure LED Utilities for MD Software RAIDs

Storage enclosure LED Monitoring utility (**ledmon**) and LED Control (**ledctl**) utility are Linux user space applications that use a broad range of interfaces and protocols to control storage enclosure LEDs. The primary usage is to visualize the status of Linux MD software RAID devices created with the mdadm utility. The `ledmon` daemon monitors the status of the drive array and updates the status of the drive LEDs. The **ledctl** utility allows you to set LED patterns for specified devices.

These LED utilities use the SGPIO (Serial General Purpose Input/Output) specification (Small Form Factor (SFF) 8485) and the SCSI Enclosure Services (SES) 2 protocol to control LEDs. They implement the International Blinking Pattern Interpretation (IBPI) patterns of the SFF-8489 specification for SGPIO. The IBPI defines how the SGPIO standards are interpreted as states for drives and slots on a backplane and how the backplane should visualize the states with LEDs.

Some storage enclosures do not adhere strictly to the SFF-8489 specification. An enclosure processor might accept an IBPI pattern but not blink the LEDs according to the SFF-8489 specification, or the processor might support only a limited number of the IBPI patterns.

LED management (AHCI) and SAF-TE protocols are not supported by the **ledmon** and **ledctl** utilities.

The `ledmon` and **ledctl** applications have been verified to work with Intel storage controllers such as the Intel AHCI controller and Intel SAS controller. They also support PCIe-SSD (solid-state drive) enclosure LEDs to control the storage enclosure status (OK, Fail, Rebuilding) LEDs of PCIe-SSD devices that are part of an MD software RAID volume. The applications might also work with the IBPI-compliant storage controllers of other vendors (especially SAS/SCSI controllers); however, other vendors' controllers have not been tested.

`ledmon` and **ledctl** are part of the `ledmon` package, which is not installed by default. Run **sudo zypper in ledmon** to install it.

12.1 The Storage Enclosure LED Monitor Service

The `ledmon` application is a daemon process that constantly monitors the state of MD software RAID devices or the state of block devices in a storage enclosure or drive bay. Only a single instance of the daemon should be running at a time. The `ledmon` daemon is part of Intel Enclosure LED Utilities.

The state is visualized on LEDs associated with each slot in a storage array enclosure or a drive bay. The application monitors all software RAID devices and visualizes their state. It does not provide a way to monitor only selected software RAID volumes.

The `ledmon` daemon supports two types of LED systems: A two-LED system (Activity LED and Status LED) and a three-LED system (Activity LED, Locate LED, and Fail LED). This tool has the highest priority when accessing the LEDs.

To start `ledmon`, enter

```
sudo ledmon [options]
```

where [options] is one or more of the following:

OPTIONS FOR `ledmon`

`-c` *PATH*,
`--confg=`*PATH*

The configuration is read from `~/.ledctl` or from `/etc/ledcfg.conf` if existing. Use this option to specify an alternative configuration file.

Currently this option has no effect, since support for configuration files has not been implemented yet. See **`man 5 ledctl.conf`** for details.

-l *PATH*,
--log= *PATH*

Sets a path to local log file. If this user-defined file is specified, the global log file `/var/log/ledmon.log` is not used.

`-t` *SECONDS*,
`--interval=`*SECONDS*

Sets the time interval between scans of `sysfs`. The value is given in seconds. The minimum is 5 seconds. The maximum is not specified.

`--quiet`, `--error`, `--warning`, `--info`, `--debug`, `--all`

Specifies the verbosity level. The level options are specified in the order of no information to the most information. Use the `--quiet` option for no logging. Use the `--all` option to log everything. If you specify more than one verbose option, the last option in the command applies.

`-h`,
`--help`

Prints the command information to the console, then exits.

`-v`,
`--version`

Displays version of **`ledmon`** and information about the license, then exits.

Note: Known Issues

The **`ledmon`** daemon does not recognize the PFA (Predicted Failure Analysis) state from the SFF-8489 specification. Thus, the PFA pattern is not visualized.

12.2 The Storage Enclosure LED Control Application

The Enclosure LED Application (**`ledctl`**) is a user space application that controls LEDs associated with each slot in a storage enclosure or a drive bay. The **`ledctl`** application is a part of Intel Enclosure LED Utilities.

When you issue the command, the LEDs of the specified devices are set to a specified pattern and all other LEDs are turned off. This application needs to be run with `root` privileges. Because the `ledmon` application has the highest priority when accessing LEDs, some patterns set by **`ledctl`** might have no effect if the `ledmon` daemon is running (except for the Locate pattern).

The **`ledctl`** application supports two types of LED systems: A two-LED system (Activity LED and Status LED) and a three-LED system (Activity LED, Fail LED, and Locate LED).

To start **`ledctl`**, enter

```
sudo [options] PATTERN_NAME=list_of_devices
```

where [options] is one or more of the following:

`-c PATH`,
`--confg=PATH`
: Sets a path to local configuration file. If this option is specified, the global configuration file and user configuration file have no effect.

-l *PATH*,
--log= *PATH*
: Sets a path to local log file. If this user-defined file is specified, the global log file `/var/log/ledmon.log` is not used.

`--quiet`
: Turns off all messages sent to `stdout` or `stderr` out. The messages are still logged to local file and the `syslog` facility.

`-h`,
`--help`
: Prints the command information to the console, then exits.

`-v`,
`--version`
: Displays version of **`ledctl`** and information about the license, then exits.

12.2.1 Pattern Names

The **`ledctl`** application accepts the following names for *pattern_name* argument, according to the SFF-8489 specification.

`locate`
: Turns on the Locate LED associated with the specified devices or empty slots. This state is used to identify a slot or drive.

`locate_off`
: Turns off the Locate LED associated with the specified devices or empty slots.

`normal`
: Turns off the Status LED, Failure LED, and Locate LED associated with the specified devices.

`off`
: Turns off only the Status LED and Failure LED associated with the specified devices.

`ica`,
`degraded`
: Visualizes the `In a Critical Array` pattern.

`rebuild`,
`rebuild_p`
: Visualizes the `Rebuild` pattern. This supports both of the rebuild states for compatibility and legacy reasons.

`ifa`,
`failed_array`
: Visualizes the `In a Failed Array` pattern.

`hotspare`
: Visualizes the `Hotspare` pattern.

`pfa`
: Visualizes the `Predicted Failure Analysis` pattern.

`failure`,
`disk_failed`
: Visualizes the `Failure` pattern.

ses_abort
: SES-2 R/R ABORT

`ses_rebuild`
: SES-2 REBUILD/REMAP

`ses_ifa`
: SES-2 IN FAILED ARRAY

`ses_ica`
: SES-2 IN CRITICAL ARRAY

`ses_cons_check`
: SES-2 CONS CHECK

`ses_hotspare`
: SES-2 HOTSPARE

`ses_rsvd_dev`
: SES-2 RSVD DEVICE

`ses_ok`
: SES-2 OK

`ses_ident`
: SES-2 IDENT

`ses_rm`
: SES-2 REMOVE

`ses_insert`
: SES-2 INSERT

`ses_missing`
: SES-2 MISSING

`ses_dnr`
: SES-2 DO NOT REMOVE

`ses_active`
: SES-2 ACTIVE

`ses_enable_bb`
: SES-2 ENABLE BYP B

`ses_enable_ba`
: SES-2 ENABLE BYP A

`ses_devoff`
: SES-2 DEVICE OFF

`ses_fault`
: SES-2 FAULT

When a non-SES-2 pattern is sent to a device in an enclosure, the pattern is automatically translated to the SCSI Enclosure Services (SES) 2 pattern as shown above.

TABLE 12.1: TRANSLATION BETWEEN NON-SES-2 PATTERNS AND SES-2 PATTERNS

Non-SES-2 Pattern	SES-2 Pattern
locate	ses_ident
locate_off	ses_ident
normal	ses_ok

Non-SES-2 Pattern	SES-2 Pattern
off	ses_ok
ica	ses_ica
degraded	ses_ica
rebuild	ses_rebuild
rebuild_p	ses_rebuild
ifa	ses_ifa
failed_array	ses_ifa
hotspare	ses_hotspare
pfa	ses_rsvd_dev
failure	ses_fault
disk_failed	ses_fault

12.2.2 List of Devices

When you issue the **`ledctl`** command, the LEDs of the specified devices are set to the specified pattern and all other LEDs are turned off. The list of devices can be provided in one of two formats:

- A list of devices separated by a comma and no spaces
- A list in curly braces with devices separated by a space

If you specify multiple patterns in the same command, the device list for each pattern can use the same or different format. For examples that show the two list formats, see *Section 12.2.3, "Examples"*.

A device is a path to file in the `/dev` directory or in the `/sys/block` directory. The path can identify a block device, an MD software RAID device, or a container device. For a software RAID device or a container device, the reported LED state is set for all of the associated block devices.

The LEDs of devices listed in list_of_devices are set to the given pattern pattern_name and all other LEDs are turned off.

12.2.3 Examples

To locate a single block device:

```
sudo ledctl locate=/dev/sda
```

To turn off the Locate LED for a single block device:

```
sudo ledctl locate_off=/dev/sda
```

To locate disks of an MD software RAID device and to set a rebuild pattern for two of its block devices at the same time:

```
sudo ledctl locate=/dev/md127 rebuild={ /sys/block/sd[a-b] }
```

To turn off the Status LED and Failure LED for the specified devices:

```
sudo ledctl off={ /dev/sda /dev/sdb }
```

To locate three block devices run one of the following commends (both are equivalent):

```
sudo ledctl locate=/dev/sda,/dev/sdb,/dev/sdc
sudo ledctl locate={ /dev/sda /dev/sdb /dev/sdc }
```

12.3 Additional Information

See the following resources for details about the LED patterns and monitoring tools:

- LEDMON open source project on GitHub.com (https://github.com/intel/ledmon.git)
- SGPIO specification SFF-8485 (ftp://ftp.seagate.com/sff/SFF-8485.PDF)
- IBPI specification SFF-8489 (ftp://ftp.seagate.com/sff/SFF-8489.PDF)

IV Network Storage

13 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 simplifies 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: Security Considerations

iSNS should be used only in secure internal networks.

13.1 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 into which you then group or organize iSCSI targets and initiators. 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.

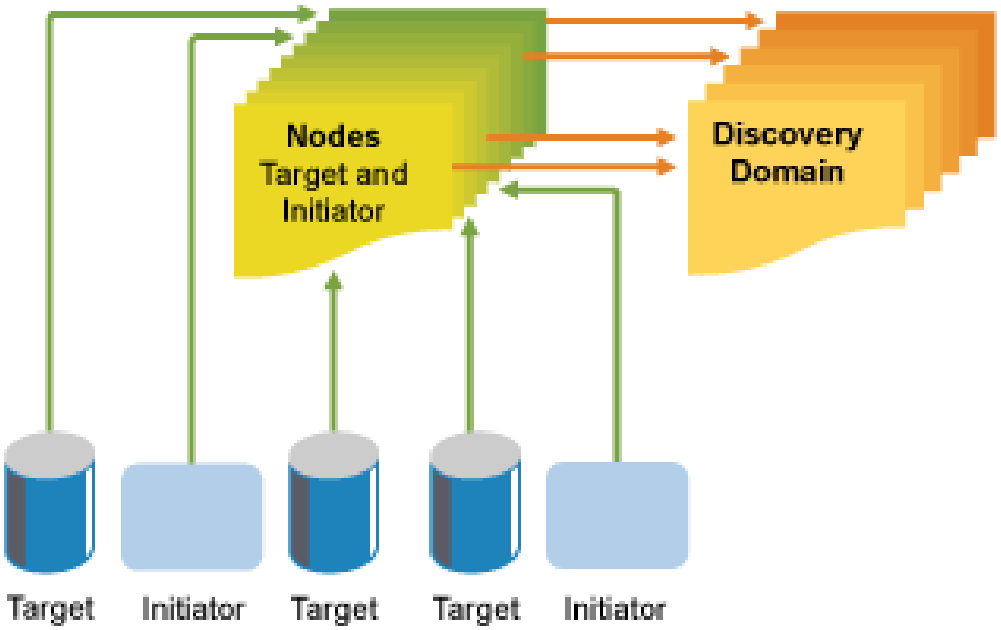


FIGURE 13.1: ISNS DISCOVERY DOMAINS

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.

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.

13.2 Installing iSNS Server for Linux

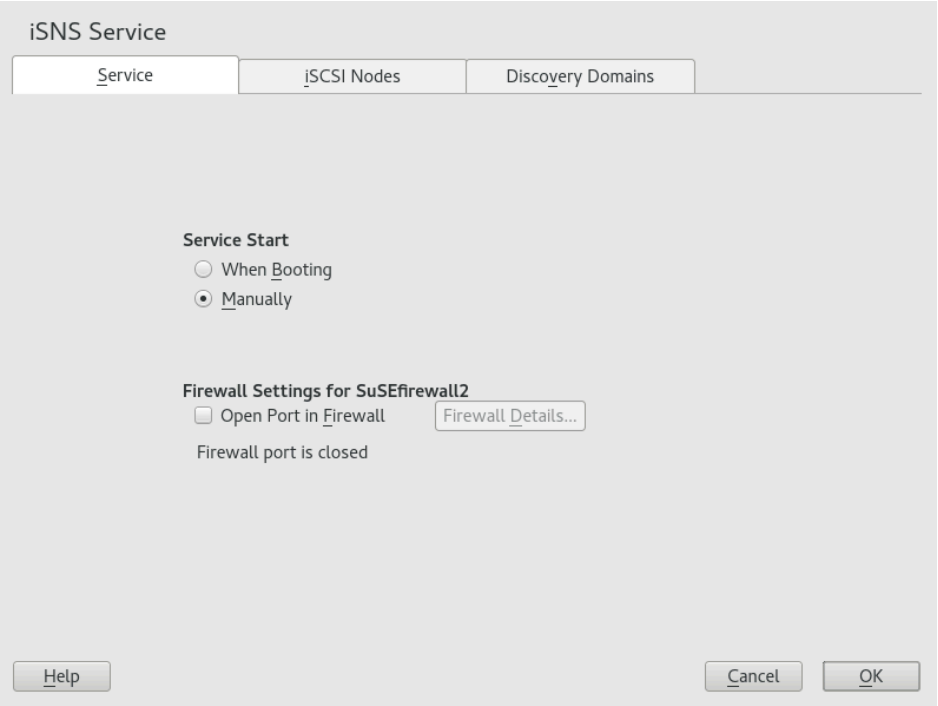
iSNS Server for Linux is included with SUSE Linux Enterprise Server, but is not installed or configured by default. You need to install the package `open-isns` and configure the iSNS service.

Note: iSNS and iSCSI on the Same Server

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. In case `open-isns` is not installed yet, you are prompted to install it now. Confirm by clicking *Install*.
3. The iSNS Service configuration dialog opens automatically to the *Service* tab.



4. In *Service Start*, select one of the following:

   - **When Booting:** The iSNS service starts automatically on server start-up.
   - **Manually (Default):** The iSNS service must be started manually by entering `sudo systemctl start isnsd` at the server console of the server where you install it.

5. 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*.

6. Click *OK* to apply the configuration settings and complete the installation.

7. Continue with *Section 13.3, "Configuring iSNS Discovery Domains"*.

13.3 Configuring iSNS Discovery Domains

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

Important: The iSNS Service Must be Active

The iSNS service must be installed and running to configure iSNS discovery domains. For information, see *Section 13.4, "Starting the iSNS Service"*.

13.3.1 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:

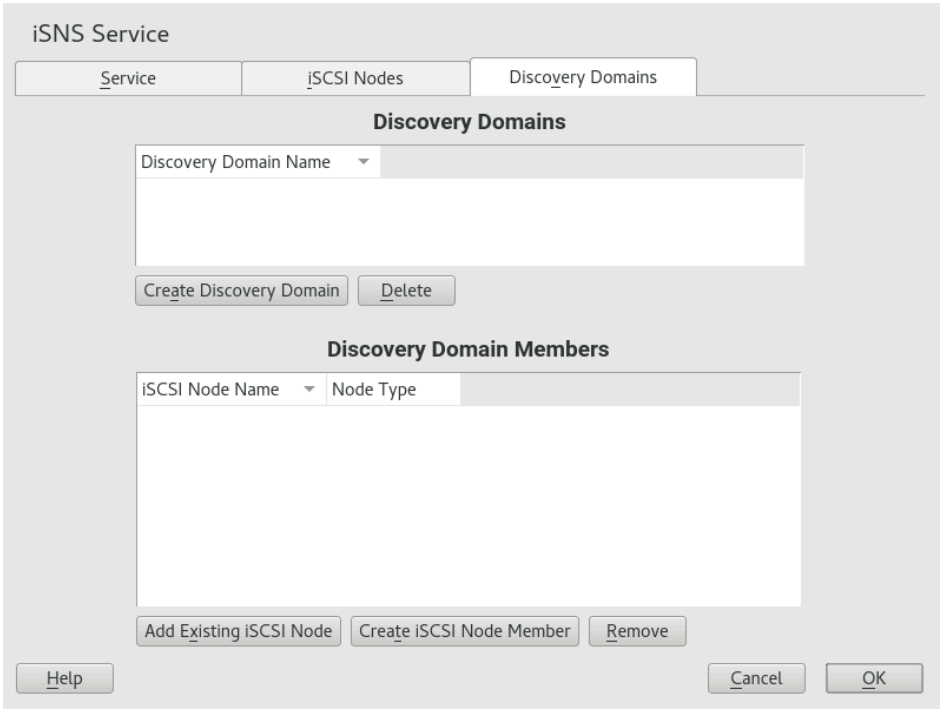
1. Start YaST and under *Network Services*, select *iSNS Server*.

2. Click the *Discovery Domains* tab.

   The *Discovery Domains* area lists all existing discovery domains. You can *Create 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. *Create iSCSI Node Member* 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.

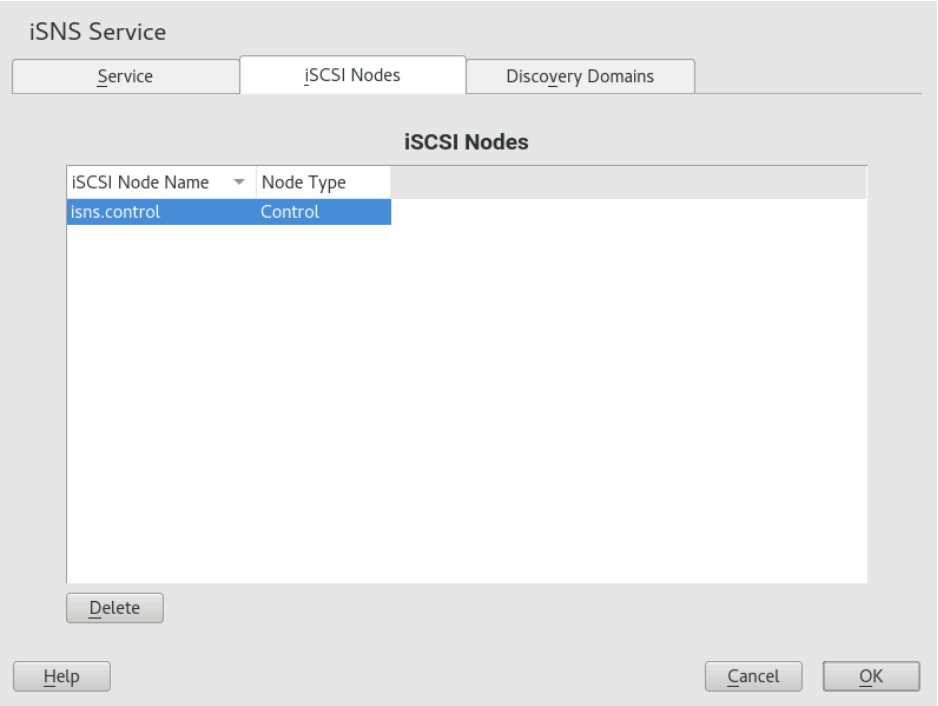
4. Specify the name of the discovery domain you are creating, then click *OK*.

5. Continue with *Section 13.3.2, "Adding iSCSI Nodes to a Discovery Domain"*.

13.3.2 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 ensure 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

   ```
   sudo systemctl restart iscsid.socket
   sudo systemctl restart iscsi
   ```

   to restart an initiator or

   ```
   sudo systemctl restart target-isns
   ```

   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 and select the desired discovery domain.

5. Click *Add existing iSCSI Node*, select the node you want to add to the domain, then click *Add Node*.

6. Repeat the previous step for as many nodes as you want to add to the discovery domain, then click *Done* when you are finished adding nodes.
   Note that an iSCSI node can belong to more than one discovery domain.

13.4 Starting the iSNS Service

iSNS must be started at the server where you install it. If you have not configured it to be started at boot time (see *Section 13.2, "Installing iSNS Server for Linux"* for details), enter the following command at a terminal:

```
sudo systemctl start isnsd
```

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

13.5 For More Information

For information, see:

- iSNS server and client for Linux project (https://github.com/open-iscsi/open-isns)
- iSNS client for the Linux LIO iSCSI target (https://github.com/open-iscsi/target-isns)
- iSCSI tools for Linux (https://www.open-iscsi.com)

General information about iSNS is available in *RFC 4171: Internet Storage Name Service* at https://datatracker.ietf.org/doc/html/rfc4171.

14 Mass Storage over IP Networks: iSCSI

One of the primary tasks of a computer center, or any site that supports servers, is to provide adequate disk capacity. 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 iSCSI LIO target software for connecting Linux servers to central storage systems.

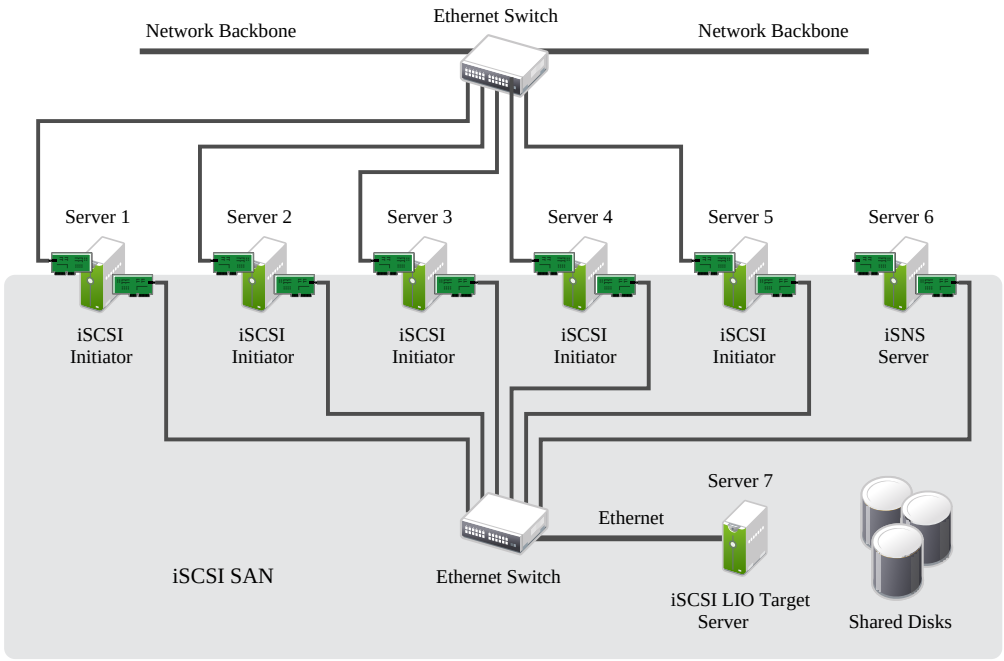


FIGURE 14.1: ISCSI SAN WITH AN ISNS SERVER

Note: LIO

LIO (http://linux-iscsi.org) is the standard open source multiprotocol SCSI target for Linux. LIO replaced the STGT (SCSI Target) framework as the standard unified storage target in Linux with Linux kernel version 2.6.38 and later. In SUSE Linux Enterprise Server 12 the iSCSI LIO Target Server replaces the iSCSI Target Server from previous versions.

iSCSI is a storage networking protocol that simplifies 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.

The iSCSI LIO target server and iSCSI 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 LIO 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 LIO target server*, or simply 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 and a lot of memory to improve data caching. For more information about RAID, also see *Chapter 7, Software RAID Configuration*.

14.1 Installing the iSCSI LIO Target Server and iSCSI Initiator

While the iSCSI initiator is installed by default (packages `open-iscsi` and `yast2-iscsi-client`), the iSCSI LIO target packages need to be installed manually.

Important: Initiator and Target may not Run on the Same Server

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

To install the iSCSI LIO Target Server, run the following command in a terminal:

```
sudo zypper in yast2-iscsi-lio-server
```

In case you need to install the iSCSI initiator or any of its dependencies, run the command **`sudo zypper in yast2-iscsi-client`**.

Alternatively, use the YaST Software Management module for installation.

Any packages required in addition to the ones mentioned above will either be automatically pulled in by the installer, or be installed when you first run the respective YaST module.

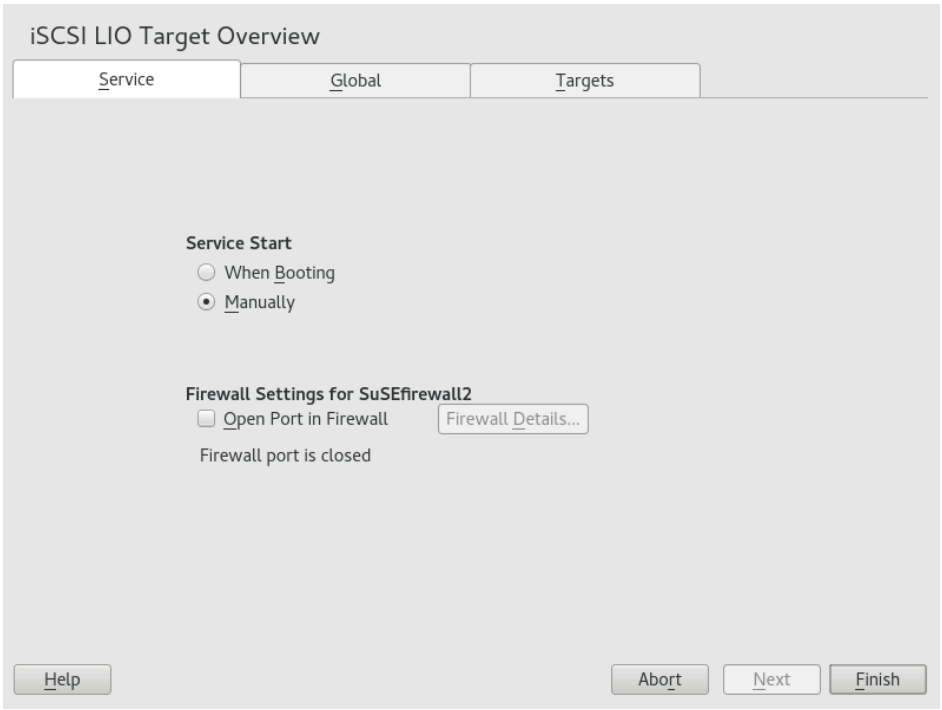
14.2 Setting Up an iSCSI LIO Target Server

This section describes how to use YaST to configure an iSCSI LIO Target Server and set up iSCSI LIO target devices. You can use any iSCSI initiator software to access the target devices.

14.2.1 iSCSI LIO Target Service Start-up and Firewall Settings

The iSCSI LIO Target service is by default configured to be started manually. You can configure the service to start automatically at boot time. If you use a firewall on the server and you want the iSCSI LIO targets to be available to other computers, you must open a port in the firewall for each adapter that you want to use for target access. TCP port 3260 is the port number for the iSCSI protocol, as defined by IANA (Internet Assigned Numbers Authority).

1. Start YaST and launch *Network Services* › *iSCSI LIO Target*.
2. Switch to the *Service* tab.



3. Under *Service Start*, specify how you want the iSCSI LIO target service to be started:

   - **When Booting:** The service starts automatically on server restart.
   - **Manually:** (Default) You must start the service manually after a server restart by running `sudo systemctl start target`. The target devices are not available until you start the service.

4. If you use a firewall on the server and you want the iSCSI LIO targets to be available to other computers, open port 3260 in the firewall for each adapter interface that you want to use for target access. If the port is closed for all of the network interfaces, the iSCSI LIO targets are not available to other computers.
   If you do not use a firewall on the server, the firewall settings are disabled. In this case skip the following steps and leave the configuration dialog with *Finish* or switch to another tab to continue with the configuration.

   a. On the *Services* tab, select the *Open Port in Firewall* check box to enable the firewall settings.

   b. Click *Firewall Details* to view or configure the network interfaces to use. All available network interfaces are listed, and all are selected by default. Deselect all interfaces on which the port should *not* be opened. Save your settings with *OK*.

5. Click *Finish* to save and apply the iSCSI LIO Target service settings.

14.2.2 Configuring Authentication for Discovery of iSCSI LIO Targets and Initiators

The iSCSI LIO Target Server software supports the PPP-CHAP (Point-to-Point Protocol Challenge Handshake Authentication Protocol), a three-way authentication method defined in the *Internet Engineering Task Force (IETF) RFC 1994* (https://datatracker.ietf.org/doc/html/rfc1994). The server uses this authentication method for the discovery of iSCSI LIO targets and initiators, not for accessing files on the targets. If you do not want to restrict the access to the discovery, use *No Authentication*. The *No Discovery Authentication* option is enabled by default. Without requiring authentication all iSCSI LIO targets on this server can be discovered by any iSCSI initiator on the same network.

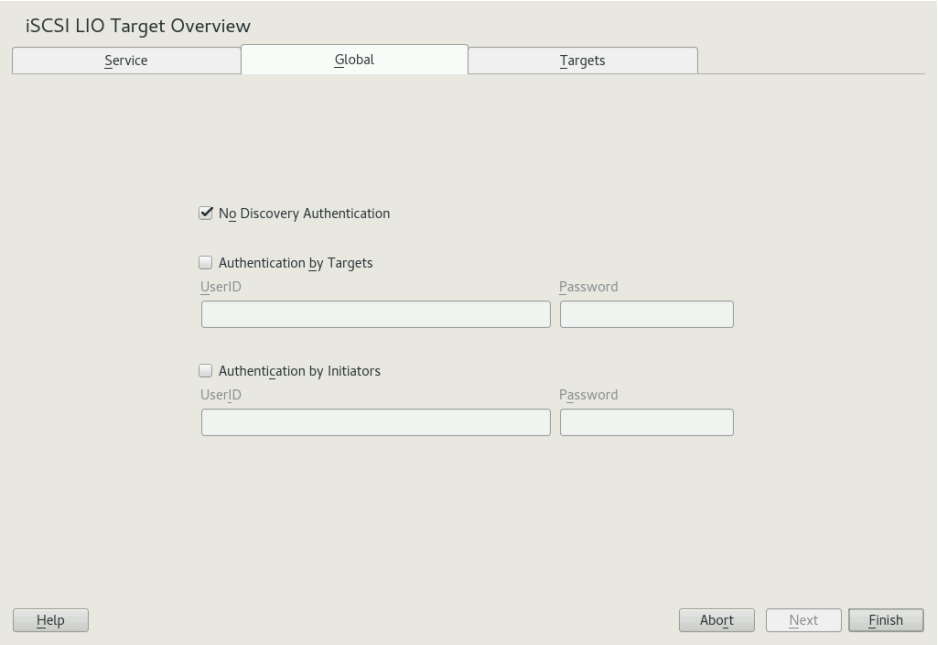
If authentication is needed for a more secure configuration, you can use incoming authentication, outgoing authentication, or both. *Authentication by Initiators* requires an iSCSI initiator to prove that it has the permissions to run a discovery on the iSCSI LIO target. The initiator must provide the incoming user name and password. *Authentication by Targets* requires the iSCSI LIO target to prove to the initiator that it is the expected target. The iSCSI LIO target must provide the outgoing user name and password to the iSCSI initiator. The password needs to be different for incoming and outgoing discovery. If authentication for discovery is enabled, its settings apply to all iSCSI LIO target groups.

Important: Security

We recommend that you use authentication for target and initiator discovery in production environments for security reasons.

To configure authentication preferences for iSCSI LIO targets:

1. Start YaST and launch *Network Services* › *iSCSI LIO Target*.
2. Switch to the *Global* tab.



3. By default, authentication is disabled (*No Discovery Authentication*). To enable Authentication, select *Authentication by Initiators*, *Outgoing Authentication* or both.

4. Provide credentials for the selected authentication method(s). The user name and password pair must be different for incoming and outgoing discovery.

5. Click *Finish* to save and apply the settings.

14.2.3 Preparing the Storage Space

Before you configure LUNs for your iSCSI Target servers, you must prepare the storage you want to use. You can use the entire unformatted block device as a single LUN, or you can subdivide a device into unformatted partitions and use each partition as a separate LUN. The iSCSI target configuration exports the LUNs to iSCSI initiators.

You can use the Partitioner in YaST or the command line to set up the partitions. Refer to *Book "Deployment Guide", Chapter 13 "Advanced Disk Setup", Section 13.1 "Using the YaST Partitioner"* for details. iSCSI LIO targets can use unformatted partitions with Linux, Linux LVM, or Linux RAID file system IDs.

Important: Do Not Mount iSCSI Target Devices

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 mount the partitions on the target server.

14.2.3.1 Partitioning Devices in a Virtual Environment

You can use a virtual machine guest server as an iSCSI LIO Target Server. This section describes how to assign partitions to a Xen virtual machine. You can also use other virtual environments that are supported by SUSE Linux Enterprise Server.

In a Xen virtual environment, you must assign the storage space you want to use for the iSCSI LIO target 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 machine, 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-backed 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.

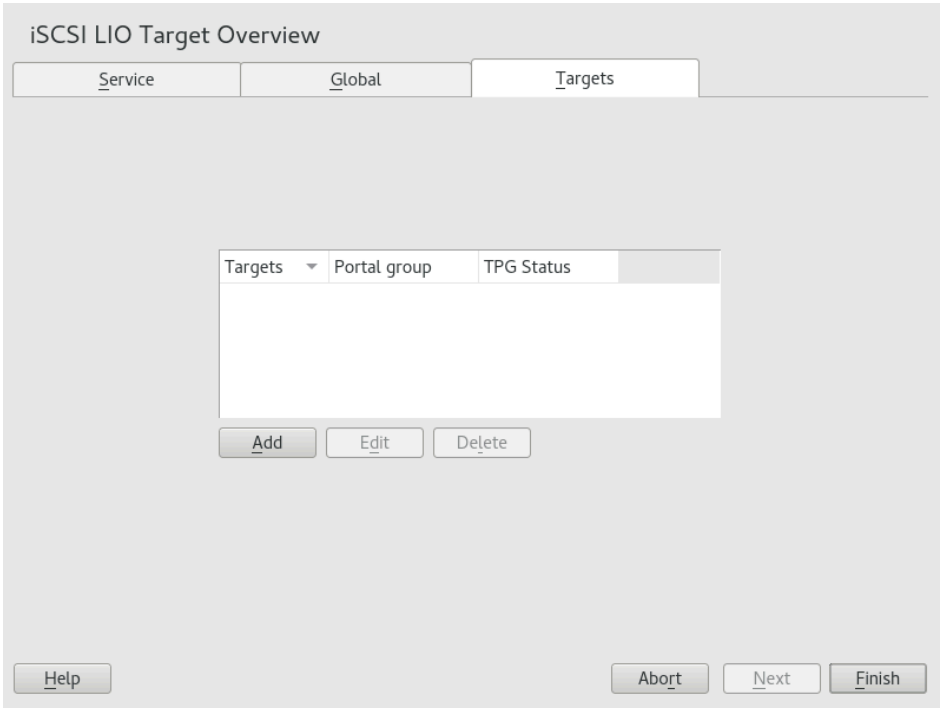
14.2.4 Setting Up an iSCSI LIO Target Group

You can use YaST to configure iSCSI LIO target devices. YaST uses APIs provided by the **lio-utils** software. iSCSI LIO targets can use unformatted partitions with Linux, Linux LVM, or Linux RAID file system IDs.

Important: Partitions

Before you begin, create the unformatted partitions that you want to use as iSCSI LIO targets as described in *Section 14.2.3, "Preparing the Storage Space"*.

1. Start YaST and launch *Network Services* › *iSCSI LIO Target*.
2. Switch to the *Targets* tab.



3. Click *Add*, then define a new iSCSI LIO target group and devices:
   The iSCSI LIO Target software automatically completes the *Target*, *Identifier*, *Portal Group*, *IP Address*, and *Port Number* fields. *Use Authentication* is selected by default.

   a. If you have multiple network interfaces, use the IP address drop-down box to select the IP address of the network interface to use for this target group. To make the server accessible under all addresses, choose *Bind All IP Addresses*.

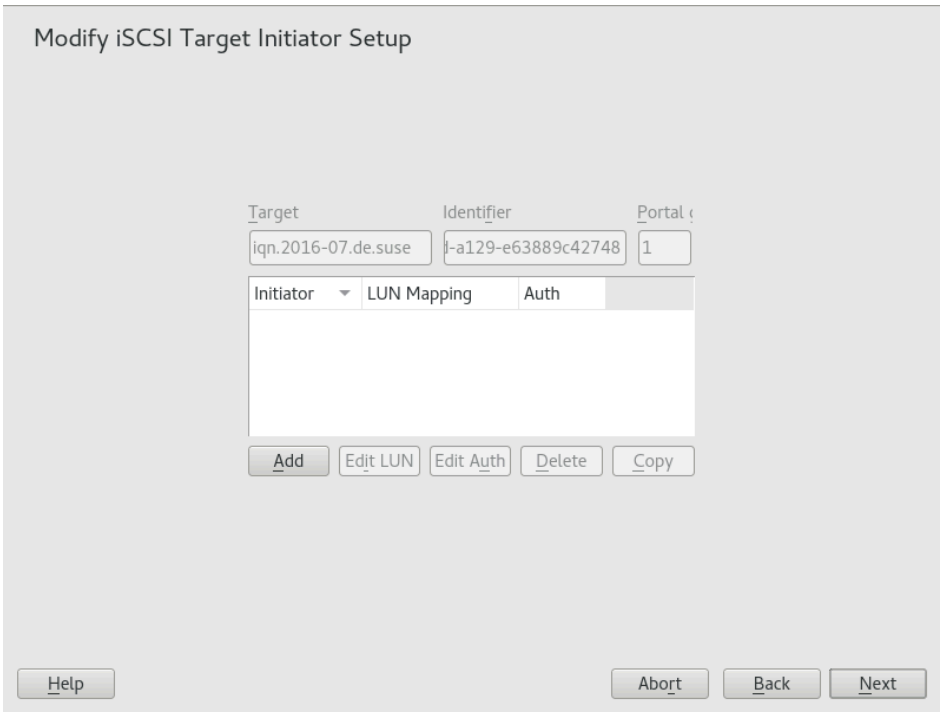
   b. Deselect *Use Authentication* if you do not want to require initiator authentication for this target group (not recommended).

   c. Click *Add*. Enter the path of the device or partition or *Browse* to add it. Optionally specify a name, then click *OK*. The LUN number is automatically generated, beginning with 0. A name is automatically generated if you leave the field empty.

   d. (Optional) Repeat the previous steps to add more targets to this target group.

   e. After all desired targets have been added to the group, click *Next*.

4. On the *Modify iSCSI Target Initiator Setup* page, configure information for the initiators that are permitted to access LUNs in the target group:



After you specify at least one initiator for the target group, the *Edit LUN*, *Edit Auth*, *Delete*, and *Copy* buttons are enabled. You can use *Add* or *Copy* to add more initiators for the target group:

MODIFY ISCSI TARGET: OPTIONS

- **Add:** Add a new initiator entry for the selected iSCSI LIO target group.
- **Edit LUN:** Configure which LUNs in the iSCSI LIO target group to map to a selected initiator. You can map each of the allocated targets to a preferred initiator.
- **Edit Auth:** Configure the preferred authentication method for a selected initiator. You can specify no authentication, or you can configure incoming authentication, outgoing authentication, or both.
- **Delete:** Remove a selected initiator entry from the list of initiators allocated to the target group.
- **Copy:** Add a new initiator entry with the same LUN mappings and authentication settings as a selected initiator entry. This allows you to easily allocate the same shared LUNs, in turn, to each node in a cluster.

a. Click *Add*, specify the initiator name, select or deselect the *Import LUNs from TPG* check box, then click *OK* to save the settings.

b. Select an initiator entry, click *Edit LUN*, modify the LUN mappings to specify which LUNs in the iSCSI LIO target group to allocate to the selected initiator, then click *OK* to save the changes.
   If the iSCSI LIO target group consists of multiple LUNs, you can allocate one or multiple LUNs to the selected initiator. By default, each of the available LUNs in the group are assigned to an initiator LUN.
   To modify the LUN allocation, perform one or more of the following actions:

   - **Add:** Click *Add* to create a new *Initiator LUN* entry, then use the *Change* drop-down box to map a target LUN to it.
   - **Delete:** Select the *Initiator LUN* entry, then click *Delete* to remove a target LUN mapping.
   - **Change:** Select the *Initiator LUN* entry, then use the *Change* drop-down box to select which Target LUN to map to it.

Typical allocation plans include the following:

- A single server is listed as an initiator. All of the LUNs in the target group are allocated to it.
  You can use this grouping strategy to logically group the iSCSI SAN storage for a given server.

- Multiple independent servers are listed as initiators. One or multiple target LUNs are allocated to each server. Each LUN is allocated to only one server.
  You can use this grouping strategy to logically group the iSCSI SAN storage for a given department or service category in the data center.

- Each node of a cluster is listed as an initiator. All of the shared target LUNs are allocated to each node. All nodes are attached to the devices, but for most file systems, the cluster software locks a device for access and mounts it on only one node at a time. Shared file systems (such as OCFS2) make it possible for multiple nodes to concurrently mount the same file structure and to open the same files with read and write access.
  You can use this grouping strategy to logically group the iSCSI SAN storage for a given server cluster.

c. Select an initiator entry, click *Edit Auth*, specify the authentication settings for the initiator, then click *OK* to save the settings.
   You can require *No Discovery Authentication*, or you can configure *Authentication by Initiators*, *Outgoing Authentication*, or both. You can specify only one user name and password pair for each initiator. The credentials can be different for incoming and outgoing authentication for an initiator. The credentials can be different for each initiator.

d. Repeat the previous steps for each iSCSI initiator that can access this target group.

e. After the initiator assignments are configured, click *Next*.

5. Click *Finish* to save and apply the settings.

14.2.5 Modifying an iSCSI LIO Target Group

You can modify an existing iSCSI LIO target group as follows:

- Add or remove target LUN devices from a target group
- Add or remove initiators for a target group
- Modify the initiator LUN-to-target LUN mappings for an initiator of a target group
- Modify the user name and password credentials for an initiator authentication (incoming, outgoing, or both)

To view or modify the settings for an iSCSI LIO target group:

1. Start YaST and launch *Network Services* › *iSCSI LIO Target*.
2. Switch to the *Targets* tab.
3. Select the iSCSI LIO target group to be modified, then click *Edit*.
4. On the Modify iSCSI Target LUN Setup page, add LUNs to the target group, edit the LUN assignments, or remove target LUNs from the group. After all desired changes have been made to the group, click *Next*.
   For option information, see *Modify iSCSI Target: Options*.
5. On the Modify iSCSI Target Initiator Setup page, configure information for the initiators that are permitted to access LUNs in the target group. After all desired changes have been made to the group, click *Next*.
6. Click *Finish* to save and apply the settings.

14.2.6 Deleting an iSCSI LIO Target Group

Deleting an iSCSI LIO target group removes the definition of the group, and the related setup for initiators, including LUN mappings and authentication credentials. It does not destroy the data on the partitions. To give initiators access again, you can allocate the target LUNs to a different or new target group, and configure the initiator access for them.

1. Start YaST and launch *Network Services* › *iSCSI LIO Target*.

2. Switch to the *Targets* tab.
3. Select the iSCSI LIO target group to be deleted, then click *Delete*.
4. When you are prompted, click *Continue* to confirm the deletion, or click *Cancel* to cancel it.
5. Click *Finish* to save and apply the settings.

14.3 Configuring iSCSI Initiator

The iSCSI initiator can be used to connect to any iSCSI target. This is not restricted to the iSCSI target solution explained in *Section 14.2, "Setting Up an iSCSI LIO Target Server"*. 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.

14.3.1 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.

Connected Targets:

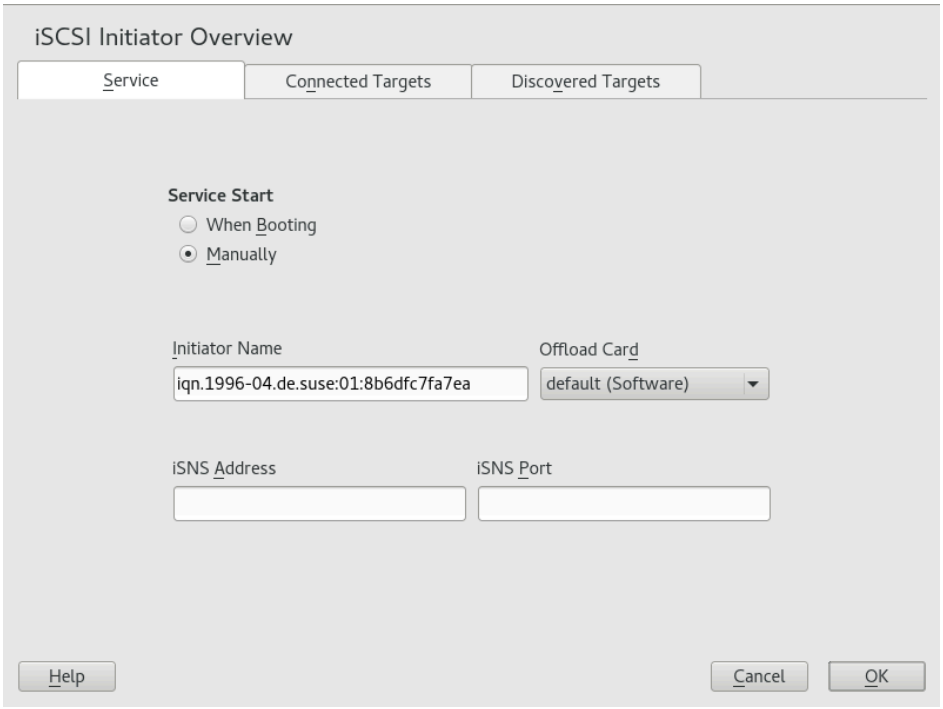
The *Connected Targets* tab gives an overview of the currently connected iSCSI targets. Like the *Discovered Targets* tab, it also gives the option to add new targets to the system.

Discovered Targets:

The *Discovered Targets* tab provides the possibility of manually discovering iSCSI targets in the network.

14.3.1.1 Configuring the iSCSI Initiator

1. Start YaST and launch *Network Services* › *iSCSI Initiator*.
2. Switch to the *Services* tab.



3. Under *Service Start*, specify how you want the iSCSI initiator service to be started:

   - **When Booting:** The service starts automatically on server restart.
   - **Manually:** (Default) You must start the service manually after a server restart by running `sudo systemctl start iscsi iscsid`.

4. Specify or verify the *Initiator Name*.
   Specify a well-formed iSCSI 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:a5dfcea717a
   ```

   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 14.3.1.2, "Discovering iSCSI Targets by Using iSNS"*.
   - **Discovered Targets:** To discover iSCSI target devices manually, continue with *Section 14.3.1.3, "Discovering iSCSI Targets Manually"*.

14.3.1.2 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 13, iSNS for Linux*.

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. Click *OK* to save and apply your changes.

14.3.1.3 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 Discovery Authentication*, then specify the credentials for *Authentication by Initiator* or *Authentication by Targets*.
5. Click *Next* to start the discovery and connect to the iSCSI target server.

6. If credentials are required, after a successful discovery, use *Connect* to activate the target. You are prompted for authentication credentials to use the selected iSCSI target.

7. Click *Next* to finish the configuration.
   The target now appears in *Connected Targets* and the virtual iSCSI device is now available.

8. Click *OK* to save and apply your changes.

9. You can find the local device path for the iSCSI target device by using the `lsscsi` command.

14.3.1.4 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. This option is ignored on platforms that cannot boot from iSCSI, such as IBM IBM Z. Therefore it should not be used on these platforms; use *Automatic* instead.

4. Click *OK* to save and apply your changes.

14.3.2 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 `/etc/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 now. 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 following command:

```
sudo iscsiadm -m discovery --type=st --portal=TARGET_IP
```

The output should look like the following:

```
10.44.171.99:3260,1 iqn.2006-02.com.example.iserv:systems
```

To discover the available targets on an `iSNS` server, use the following command:

```
sudo iscsiadm --mode discovery --type isns --portal TARGET_IP
```

For each target defined on the iSCSI target, one line appears. For more information about the stored data, see *Section 14.3.3, "The iSCSI Initiator Databases"*.

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

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

The newly generated devices show up in the output of **`lsscsi`** and can now be mounted.

14.3.3 The iSCSI Initiator Databases

All information that was discovered by the iSCSI initiator is stored in two database files that reside in `/etc/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`** with one of these parameters gives an overview of the stored records:

```
tux > sudo iscsiadm -m discovery
10.44.171.99:3260,1 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:

```
tux > sudo 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_cmdsn = 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`:

```
sudo iscsiadm -m node -n iqn.2006-02.com.example.iserv:systems \
-p ip:port --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:

```
sudo iscsiadm -m node -n iqn.2006-02.com.example.iserv:systems \
-p ip:port --op=delete
```

Important: No Confirmation

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 **`sudo iscsiadm -m node`** command.

14.4 Using iSCSI Disks when Installing

Booting from an iSCSI disk on AMD64/Intel 64 and IBM POWER architectures is supported when iSCSI-enabled firmware is used.

To use iSCSI disks during installation, it is necessary to add the following parameter to the boot option line:

```
withiscsi=1
```

During installation, an additional screen appears that provides the option to attach iSCSI disks to the system and use them in the installation process.

Note: Mount Point Support

iSCSI devices will appear asynchronously during the boot process. While the initrd guarantees that those devices are set up correctly for the root file system, there are no such guarantees for any other file systems or mount points like `/usr`. Hence any system mount points like `/usr` or `/var` are not supported. To use those devices, ensure correct synchronization of the respective services and devices.

14.5 Troubleshooting iSCSI

This section describes some known issues and possible solutions for iSCSI target and iSCSI initiator issues.

14.5.1 Portal Error When Setting Up Target LUNs on an iSCSI LIO Target Server

When adding or editing an iSCSI LIO target group, you get an error:

```
Problem setting network portal IP_ADDRESS:3260
```

The `/var/log/YasT2/y2log` log file contains the following error:

```
find: `/sys/kernel/config/target/iscsi': No such file or directory
```

This problem occurs if the iSCSI LIO Target Server software is not currently running. To resolve this issue, exit YaST, manually start iSCSI LIO at the command line with **`systemctl start target`**, then try again.

You can also enter the following to check if **configfs**, **iscsi_target_mod**, and **target_core_mod** are loaded. A sample response is shown.

```
tux > sudo lsmod | grep iscsi
iscsi_target_mod      295015  0
target_core_mod       346745  4
iscsi_target_mod,target_core_pscsi,target_core_iblock,target_core_file
configfs               35817  3 iscsi_target_mod,target_core_mod
scsi_mod              231620  16
iscsi_target_mod,target_core_pscsi,target_core_mod,sg,sr_mod,mptctl,sd_mod,
scsi_dh_rdac,scsi_dh_emc,scsi_dh_alua,scsi_dh_hp_sw,scsi_dh,libata,mptspi,
mptscsih,scsi_transport_spi
```

14.5.2 iSCSI LIO Targets Are Not Visible from Other Computers

If you use a firewall on the target server, you must open the iSCSI port that you are using to allow other computers to see the iSCSI LIO targets. For information, see *Section 14.2.1, "iSCSI LIO Target Service Start-up and Firewall Settings"*.

14.5.3 Data Packets Dropped for iSCSI Traffic

A firewall might drop packets if it gets too busy. The default for the SUSE Firewall is to drop packets after three minutes. If you find that iSCSI traffic packets are being dropped, you can consider configuring the SUSE Firewall to queue packets instead of dropping them when it gets too busy.

14.5.4 Using iSCSI Volumes with LVM

Use the troubleshooting tips in this section when using LVM on iSCSI targets.

14.5.4.1 Check if the iSCSI Initiator Discovery Occurs at Boot

When you set up the iSCSI Initiator, ensure that you enable discovery at boot time so that udev can discover the iSCSI devices at boot time and set up the devices to be used by LVM.

14.5.4.2 Check that iSCSI Target Discovery Occurs at Boot

Remember that `udev` provides the default setup for devices. Ensure that all of the applications that create devices are started at boot time so that **`udev`** can recognize and assign devices for them at system start-up. If the application or service is not started until later, **`udev`** does not create the device automatically as it would at boot time.

14.5.5 iSCSI Targets Are Mounted When the Configuration File Is Set to Manual

When Open-iSCSI starts, it can mount the targets even if the `node.startup` option is set to manual in the `/etc/iscsi/iscsid.conf` file if you manually modified the configuration file.

Check the `/etc/iscsi/nodes/`*`TARGET_NAME`*`/`*`IP_ADDRESS`*`,`*`PORT`*`/default` file. It contains a `node.startup` setting that overrides the `/etc/iscsi/iscsid.conf` file. Setting the mount option to manual by using the YaST interface also sets `node.startup = manual` in the `/etc/iscsi/nodes/`*`TARGET_NAME`*`/`*`IP_ADDRESS`*`,`*`PORT`*`/default` files.

14.6 iSCSI LIO Target Terminology

backstore

A physical storage object that provides the actual storage underlying an iSCSI endpoint.

CDB (command descriptor block)

The standard format for SCSI commands. CDBs are commonly 6, 10, or 12 bytes long, though they can be 16 bytes or of variable length.

CHAP (Challenge Handshake Authentication Protocol)

A point-to-point protocol (PPP) authentication method used to confirm the identity of one computer to another. After the Link Control Protocol (LCP) connects the two computers, and the CHAP method is negotiated, the authenticator sends a random Challenge to the peer. The peer issues a cryptographically hashed Response that depends upon the Challenge and a secret key. The authenticator verifies the hashed Response against its own calculation of the expected hash value, and either acknowledges the authentication or terminates the connection. CHAP is defined in the RFC 1994.

CID (connection identifier)

A 16-bit number, generated by the initiator, that uniquely identifies a connection between two iSCSI devices. This number is presented during the login phase.

endpoint

The combination of an iSCSI Target Name with an iSCSI TPG (IQN + Tag).

EUI (extended unique identifier)

A 64-bit number that uniquely identifies every device in the world. The format consists of 24 bits that are unique to a given company, and 40 bits assigned by the company to each device it builds.

initiator

The originating end of an SCSI session. Typically a controlling device such as a computer.

IPS (Internet Protocol storage)

The class of protocols or devices that use the IP protocol to move data in a storage network. FCIP (Fibre Channel over Internet Protocol), iFCP (Internet Fibre Channel Protocol), and iSCSI (Internet SCSI) are all examples of IPS protocols.

IQN (iSCSI qualified name)

A name format for iSCSI that uniquely identifies every device in the world (for example: `iqn.5886.com.acme.tapedrive.sn-a12345678`).

ISID (initiator session identifier)

A 48-bit number, generated by the initiator, that uniquely identifies a session between the initiator and the target. This value is created during the login process, and is sent to the target with a Login PDU.

MCS (multiple connections per session)

A part of the iSCSI specification that allows multiple TCP/IP connections between an initiator and a target.

MPIO (multipath I/O)

A method by which data can take multiple redundant paths between a server and storage.

network portal

The combination of an iSCSI endpoint with an IP address plus a TCP (Transmission Control Protocol) port. TCP port 3260 is the port number for the iSCSI protocol, as defined by IANA (Internet Assigned Numbers Authority).

SAM (SCSI architectural model)

A document that describes the behavior of SCSI in general terms, allowing for different types of devices communicating over various media.

target

The receiving end of an SCSI session, typically a device such as a disk drive, tape drive, or scanner.

target group (TG)

A list of SCSI target ports that are all treated the same when creating views. Creating a view can help simplify LUN (logical unit number) mapping. Each view entry specifies a target group, host group, and a LUN.

target port

The combination of an iSCSI endpoint with one or more LUNs.

target port group (TPG)

A list of IP addresses and TCP port numbers that determines which interfaces a specific iSCSI target will listen to.

target session identifier (TSID)

A 16-bit number, generated by the target, that uniquely identifies a session between the initiator and the target. This value is created during the login process, and is sent to the initiator with a Login Response PDU (protocol data units).

14.7 Additional Information

The iSCSI protocol has been available for several years. There are many reviews comparing iSCSI with SAN solutions, benchmarking performance, and there also is documentation describing hardware solutions. Important sources of more information about open-iscsi are:

- The *Open-iSCSI Project* (http://www.open-iscsi.com/)
- The online documentation and man pages for **`iscsiadm`**, **`iscsid`**, `ietd.conf`, and **`ietd`** and the example configuration file `/etc/iscsid.conf`.

15 Fibre Channel Storage over Ethernet Networks: FCoE

Many enterprise data centers rely on Ethernet for their LAN and data traffic, and on Fibre Channel networks for their storage infrastructure. Open Fibre Channel over Ethernet (FCoE) Initiator software allows servers with Ethernet adapters to connect to a Fibre Channel storage subsystem over an Ethernet network. This connectivity was previously reserved exclusively for systems with Fibre Channel adapters over a Fibre Channel fabric. The FCoE technology reduces complexity in the data center by aiding network convergence. This helps to preserve your existing investments in a Fibre Channel storage infrastructure and to simplify network management.

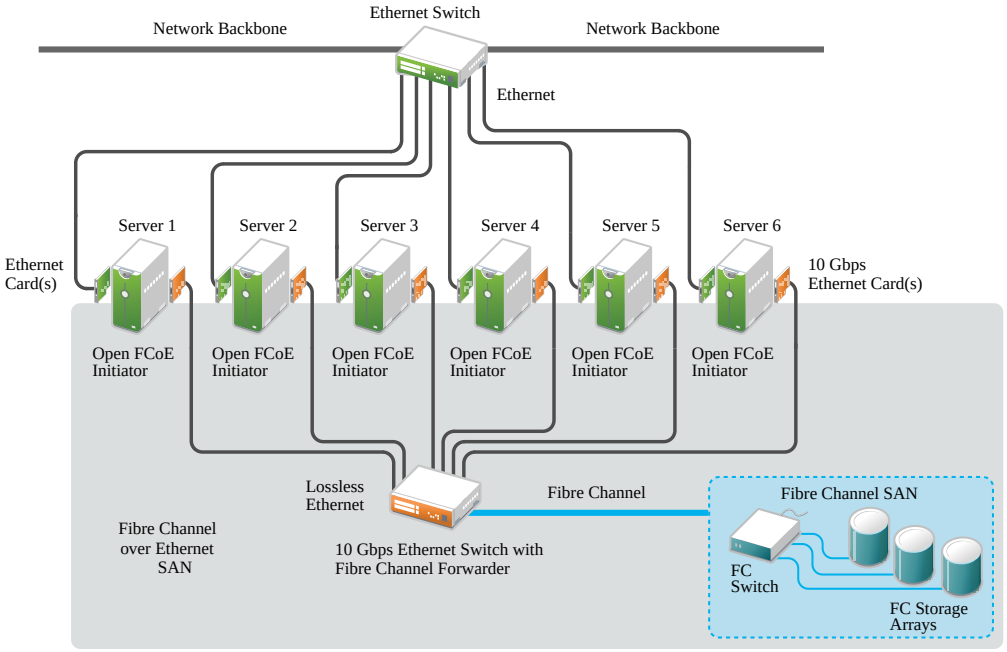


FIGURE 15.1: OPEN FIBRE CHANNEL OVER ETHERNET SAN

Open-FCoE allows you to run the Fibre Channel protocols on the host, instead of on proprietary hardware on the host bus adapter. It is targeted for 10 Gbps (gigabit per second) Ethernet adapters, but can work on any Ethernet adapter that supports pause frames. The initiator software provides a Fibre Channel protocol processing module and an Ethernet-based transport module. The Open-FCoE module acts as a low-level driver for SCSI. The Open-FCoE transport uses `net_device` to send and receive packets. Data Center Bridging (DCB) drivers provide the quality of service for FCoE.

FCoE is an encapsulation protocol that moves the Fibre Channel protocol traffic over Ethernet connections without changing the Fibre Channel frame. This allows your network security and traffic management infrastructure to work the same with FCoE as it does with Fibre Channel.

You might choose to deploy FCoE in your enterprise if the following conditions exist:

- Your enterprise already has a Fibre Channel storage subsystem and administrators with Fibre Channel skills and knowledge.
- You are deploying 10 Gbps Ethernet in the network.

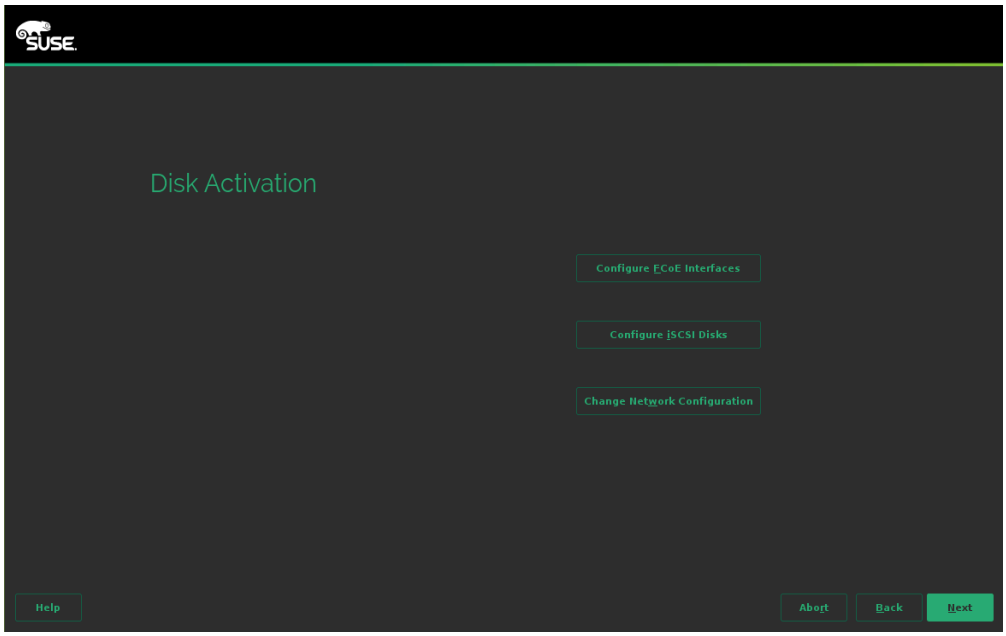
This section describes how to set up FCoE in your network.

15.1 Configuring FCoE Interfaces during the Installation

The YaST installation for SUSE Linux Enterprise Server allows you to configure FCoE disks during the operating system installation if FCoE is enabled at the switch for the connections between the server and the Fibre Channel storage infrastructure. Some system BIOS types can automatically detect the FCoE disks, and report the disks to the YaST Installation software. However, automatic detection of FCoE disks is not supported by all BIOS types. To enable automatic detection in this case, you can add the **`withfcoe`** option to the kernel command line when you begin the installation:

```
withfcoe=1
```

When the FCoE disks are detected, the YaST installation offers the option to configure FCoE instances at that time. On the Disk Activation page, select *Configure FCoE Interfaces* to access the FCoE configuration. For information about configuring the FCoE interfaces, see *Section 15.3, "Managing FCoE Services with YaST"*.



Note: Mount Point Support

FCoE devices will appear asynchronously during the boot process. While the initrd guarantees that those devices are set up correctly for the root file system, there are no such guarantees for any other file systems or mount points like `/usr`. Hence any system mount points like `/usr` or `/var` are not supported. To use those devices, ensure correct synchronization of the respective services and devices.

15.2 Installing FCoE and the YaST FCoE Client

You can set up FCoE disks in your storage infrastructure by enabling FCoE at the switch for the connections to a server. If FCoE disks are available when the SUSE Linux Enterprise Server operating system is installed, the FCoE Initiator software is automatically installed at that time.

If the FCoE Initiator software and the YaST FCoE Client software are not installed, use the following procedure to manually install them with the following command:

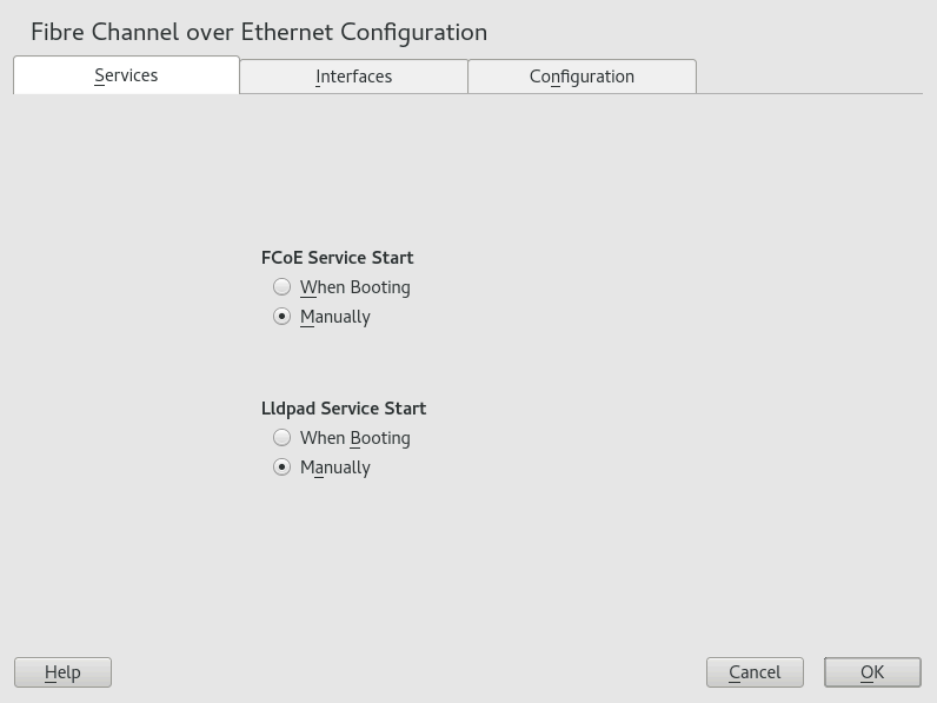
```
sudo zypper in yast2-fcoe-client fcoe-utils
```

Alternatively, use the YaST Software Manager to install the packages listed above.

15.3 Managing FCoE Services with YaST

You can use the YaST FCoE Client Configuration option to create, configure, and remove FCoE interfaces for the FCoE disks in your Fibre Channel storage infrastructure. To use this option, the FCoE Initiator service (the `fcoemon` daemon) and the Link Layer Discovery Protocol agent daemon (`llpad`) must be installed and running, and the FCoE connections must be enabled at the FCoE-capable switch.

1. Launch YaST and select *Network Services* › *FCoE Client Configuration*.



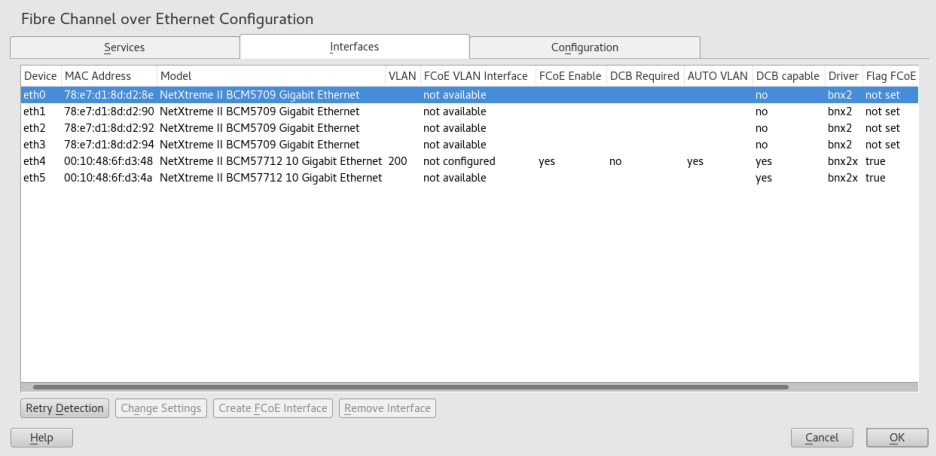
2. On the *Services* tab, view or modify the FCoE service and Lldpad (Link Layer Discovery Protocol agent daemon) service start time as necessary.

   - **FCoE Service Start:** Specifies whether to start the Fibre Channel over Ethernet service `fcoemon` daemon at the server boot time or manually. The daemon controls the FCoE interfaces and establishes a connection with the `llpad` daemon. The values are *When Booting* (default) or *Manually*.

   - **Lldpad Service Start:** Specifies whether to start the Link Layer Discovery Protocol agent `llpad` daemon at the server boot time or manually. The `llpad` daemon informs the `fcoemon` daemon about the Data Center Bridging features and the configuration of the FCoE interfaces. The values are *When Booting* (default) or *Manually*.

If you modify a setting, click *OK* to save and apply the change.

3. On the *Interfaces* tab, view information about all detected network adapters on the server, including information about VLAN and FCoE configuration. You can also create an FCoE VLAN interface, change settings for an existing FCoE interface, or remove an FCoE interface.



Use the *FCoE VLAN Interface* column to determine whether FCoE is available or not:

`Interface Name`

If a name is assigned to the interface, such as `eth4.200`, FCoE is available on the switch, and the FCoE interface is activated for the adapter.

Not Configured:

If the status is *not configured*, FCoE is enabled on the switch, but an FCoE interface has not been activated for the adapter. Select the adapter, then click *Create FCoE VLAN Interface* to activate the interface on the adapter.

Not Available:

If the status is *not available*, FCoE is not possible for the adapter because FCoE has not been enabled for that connection on the switch.

4. To set up an FCoE-enabled adapter that has not yet been configured, select it and click *Create FCoE VLAN Interface*. Confirm the query with *Yes*.
   The adapter is now listed with an interface name in the *FCoE VLAN Interface* column.

5. To change the settings for an adapter that is already configured, select it from the list, then click *Change Settings*.

The following options can be configured:

FCoE Enable

Enable or disable the creation of FCoE instances for the adapter.

DCB Required

Specifies whether Data Center Bridging is required for the adapter (usually this is the case).

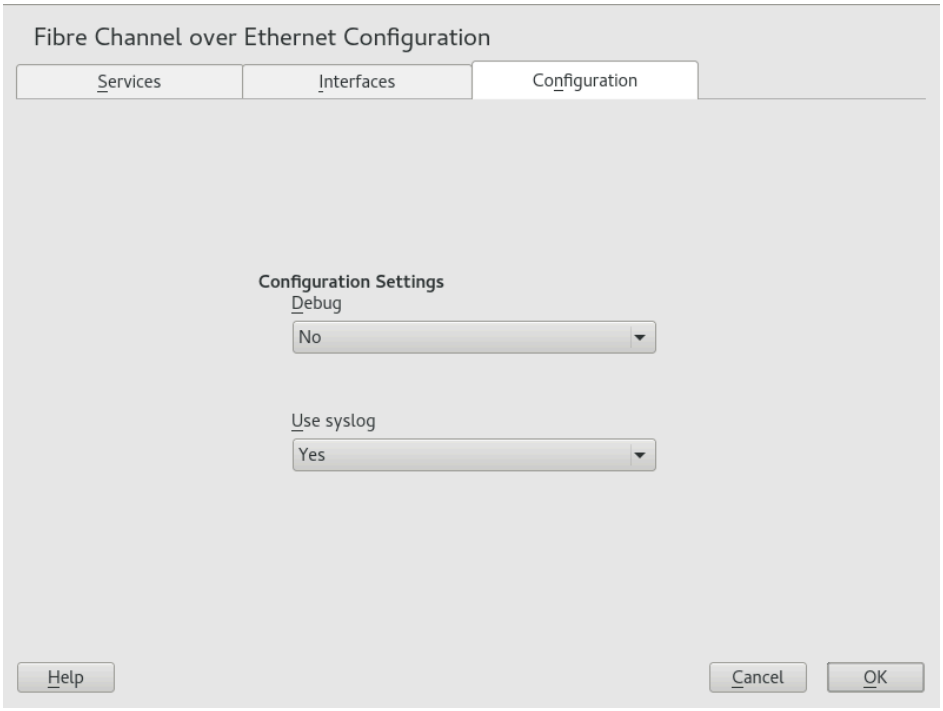
Auto VLAN

Specifies whether the `fcoemon` daemon creates the VLAN interfaces automatically.

If you modify a setting, click *Next* to save and apply the change. The settings are written to the `/etc/fcoe/cfg-ethX` file. The `fcoemon` daemon reads the configuration files for each FCoE interface when it is initialized.

6. To remove an interface that is already configured, select it from the list. Click *Remove Interface* and *Continue* to confirm. The FCoE Interface value changes to *not configured*.

7. On the *Configuration* tab, view or modify the general settings for the FCoE system service. You can enable or disable debugging messages from the FCoE service script and the `fcoemon` daemon and specify whether messages are sent to the system log.



8. Click *OK* to save and apply changes.

15.4 Configuring FCoE with Commands

1. Open a terminal.
2. Use YaST to configure the Ethernet network interface card, such as `eth2`.
3. Start the Link Layer Discovery Protocol agent daemon (`llpad`).

1. Open a terminal.
2. Enable Data Center Bridging on your Ethernet adapter.

```
tux > dcbtool sc eth2 dcb on
  Version:       2
  Command:       Set Config
  Feature:       DCB State
  Port:          eth2
  Status:        Successful
```

3. Enable and set the Priority Flow Control (PFC) settings for Data Center Bridging.

```
sudo dcbtool sc eth<x> pfc e:1 a:1 w:1
```

Argument setting values are:

e:<0|1>
: Controls feature enablement.

a:<0|1>
: Controls whether the feature is advertised via Data Center Bridging Exchange protocol to the peer.

w:<0|1>
: Controls whether the feature is willing to change its operational configuration based on what is received from the peer.

4. Enable the Data Center Bridging to accept the switch's priority setting for FCoE.

```
tux > sudo dcbtool sc eth2 app:fcoe e:1
  Version:       2
  Command:       Set Config
```

```
  Feature:       Application FCoE
  Port:          eth2
  Status:        Successful
```

5. Copy the default FCoE configuration file to `/etc/fcoe/cfg-eth2`.

```
sudo cp /etc/fcoe/cfg-ethx /etc/fcoe/cfg-eth2
```

6. Start the FCoE Initiator service.

```
systemctl start fcoe.service
```

7. Set up the Link Layer Discovery Protocol agent daemon (`llpad`) and the FCoE Initiator service to start when booting.

```
systemctl enable llpad fcoe
```

15.5 Managing FCoE Instances with the FCoE Administration Tool

The **`fcoeadm`** utility is the Fibre Channel over Ethernet (FCoE) management tool. It can be used to create, destroy, and reset an FCoE instance on a given network interface. The **`fcoeadm`** utility sends commands to a running `fcoemon` process via a socket interface. For information about **`fcoemon`**, see the **`man 8 fcoemon`**.

The **`fcoeadm`** utility allows you to query the FCoE instances about the following:

- Interfaces
- Target LUNs
- Port statistics

The **`fcoeadm`** utility is part of the `fcoe-utils` package. The general syntax for the command looks like the following:

```
fcoeadm
  [-c|--create] [<ethX>]
  [-d|--destroy] [<ethX>]
  [-r|--reset] [<ethX>]
  [-S|--Scan] [<ethX>]
  [-i|--interface] [<ethX>]
  [-t|--target] [<ethX>]
```

```
  [-l|--lun] [<ethX>]
  [-s|--stats <ethX>] [<interval>]
  [-v|--version]
  [-h|--help]
```

Refer to **`man 8 fcoeadm`** for details.

Examples

`fcoeadm -c eth2.101`

Create an FCoE instance on eth2.101.

`fcoeadm -d eth2.101`

Destroy an FCoE instance on eth2.101.

`fcoeadm -i eth3`

Show information about all FCoE instances on interface `eth3`. If no interface is specified, information for all interfaces that have FCoE instances created will be shown. The following example shows information on connection eth0.201:

```
tux > sudo fcoeadm -i eth0.201
  Description:      82599EB 10-Gigabit SFI/SFP+ Network Connection
  Revision:         01
  Manufacturer:     Intel Corporation
  Serial Number:    001B219B258C
  Driver:           ixgbe 3.3.8-k2
  Number of Ports:  1

      Symbolic Name:     fcoe v0.1 over eth0.201
      OS Device Name:    host8
      Node Name:         0x1000001B219B258E
      Port Name:         0x2000001B219B258E
      FabricName:        0x2001000573D38141
      Speed:             10 Gbit
      Supported Speed:   10 Gbit
      MaxFrameSize:      2112
      FC-ID (Port ID):   0x790003
      State:             Online
```

`fcoeadm -l eth3.101`

Show detailed information about all LUNs discovered on connection eth3.101. If no connection is specified, information about all LUNs discovered on all FCoE connections will be shown.

`fcoeadm -r eth2.101`

Reset the FCoE instance on eth2.101.

`fcoeadm -s eth3 3`

Show statistical information about a specific eth3 port that has FCoE instances, at an interval of three seconds. The statistics are displayed one line per time interval. If no interval is given, the default of one second is used.

`fcoeadm -t eth3`

Show information about all discovered targets from a given eth3 port having FCoE instances. After each discovered target, any associated LUNs are listed. If no instance is specified, targets from all ports that have FCoE instances are shown. The following example shows information of targets from the eth0.201 connection:

```
tux > sudo fcoeadm -t eth0.201
  Interface:        eth0.201
  Roles:            FCP Target
  Node Name:        0x200000D0231B5C72
  Port Name:        0x210000D0231B5C72
  Target ID:        0
  MaxFrameSize:     2048
  OS Device Name:   rport-8:0-7
  FC-ID (Port ID):  0x79000C
  State:            Online

LUN ID  Device Name   Capacity   Block Size  Description
------  -----------  ----------  ----------  ----------------------------
    40  /dev/sdqi     792.84 GB      512     IFT DS S24F-R2840-4 (rev 386C)
    72  /dev/sdpk     650.00 GB      512     IFT DS S24F-R2840-4 (rev 386C)
   168  /dev/sdgy       1.30 TB      512     IFT DS S24F-R2840-4 (rev 386C)
```

15.6 Additional Information

For information, see the follow documentation:

- For information about the Open-FCoE service daemon, see the `fcoemon(8)` man page.
- For information about the Open-FCoE Administration tool, see the `fcoeadm(8)` man page.
- For information about the Data Center Bridging Configuration tool, see the `dcbtool(8)` man page.

- For information about the Link Layer Discovery Protocol agent daemon, see the `lldpad(8)` man page.
- For general information, see the Open-FCoE home page: http://www.open-fcoe.org/dokuwiki/start.

16 NVMe-oF

This chapter describes how to set up an NVMe-oF host and target.

16.1 Overview

NVM Express (*NVMe*) is an interface standard for accessing non-volatile storage, commonly SSD disks. NVMe supports much higher speeds and has a lower latency than SATA.

NVMe-oF is an architecture to access NVMe storage over different networking fabrics, for example *RDMA* or *NVMe over Fibre Channel* (*FC-NVMe*). The role of NVMe-oF is similar to iSCSI. To increase the fault-tolerance, NVMe-oF has built-in support for multipathing.

The *NVMe host* is the machine that connects to an NVMe target. The *NVMe target* is the machine that shares its NVMe block devices.

NVMe is supported on SUSE Linux Enterprise Server 12 SP5. There are Kernel modules available for the NVMe block storage and NVMe-oF target and host.

To see if your hardware requires any special consideration, refer to *Section 16.4, "Special Hardware Configuration"*.

16.2 Setting Up an NVMe-oF Host

To use NVMe-oF, a target must be available with one of the supported networking methods. Supported are NVMe over Fibre Channel and RDMA. The following sections describe how to connect a host to an NVMe target.

16.2.1 Installing Command Line Client

To use NVMe-oF, you need the **`nvme`** command line tool. Install it with **`zypper`**:

```
tux > sudo zypper in nvme-cli
```

Use **`nvme --help`** to list all available subcommands. Man pages are available for **`nvme`** subcommands. Consult them by executing **`man nvme-`***`SUBCOMMAND`*. For example, to view the man page for the `discover` subcommand, execute **`man nvme-discover`**.

16.2.2 Discovering NVMe-oF Targets

To list available NVMe subsystems on the NVMe-oF target, you need the discovery controller address and service ID.

```
tux > sudo nvme discover -t TRANSPORT -a DISCOVERY_CONTROLLER_ADDRESS -s SERVICE_ID
```

Replace *TRANSPORT* with the underlying transport medium: `loop`, `rdma` or `fc`. Replace *DISCOVERY_CONTROLLER_ADDRESS* with the address of the discovery controller. For RDMA this should be an IPv4 address. Replace *SERVICE_ID* with the transport service ID. If the service is IP based, like RDMA, service ID specifies the port number. For Fibre Channel, the service ID is not required.

The NVMe hosts only see the subsystems they are allowed to connect to.

Example:

```
tux > sudo nvme discover -t rdma -a 10.0.0.1 -s 4420
```

For more details, see **man nvme-discover**.

16.2.3 Connecting to NVMe-oF Targets

After you have identified the NVMe subsystem, you can connect it with the **nvme connect** command.

```
tux > sudo nvme connect -t transport -a DISCOVERY_CONTROLLER_ADDRESS -s SERVICE_ID -
n SUBSYSTTEM_NQN
```

Replace *TRANSPORT* with the underlying transport medium: `loop`, `rdma` or `fc`. Replace *DISCOVERY_CONTROLLER_ADDRESS* with the address of the discovery controller. For RDMA this should be an IPv4 address. Replace *SERVICE_ID* with the transport service ID. If the service is IP based, like RDMA, this specifies the port number. Replace *SUBSYSTEM_NQN* with the NVMe qualified name of the desired subsystem as found by the discovery command. *NQN* is the abbreviation for *NVMe Qualified Name*. The NQN must be unique.

Example:

```
tux > sudo nvme connect -t rdma -a 10.0.0.1 -s 4420 -n nqn.2014-08.com.example:nvme:nvm-
subsystem-sn-d78432
```

Alternatively, use **nvme connect-all** to connect to all discovered namespaces. For advanced usage please see **man nvme-connect** and **man nvme-connect-all**.

16.2.4 Multipathing

NVMe native multipathing is disabled by default. To print the layout of the multipath devices, use the command **`nvme list-subsys`**. To enable NVMe native multipathing, add `nvme-core.multipath=on` as a boot parameter.

16.3 Setting Up an NVMe-oF Target

16.3.1 Installing Command Line Client

To configure an NVMe-oF target, you need the **`nvmetcli`** command line tool. Install it with **`zypper`**:

```
tux > sudo zypper in nvmetcli
```

The current documentation for **`nvmetcli`** is available at http://git.infradead.org/users/hch/nvmetcli.git/blob_plain/HEAD:/Documentation/nvmetcli.txt.

16.3.2 Configuration Steps

The following procedure provides an example of how to set up an NVMe-oF target.

The configuration is stored in a tree structure. Use the command **`cd`** to navigate. Use **`ls`** to list objects. You can create new objects with **`create`**.

1. Start the **`nvmectli`** interactive shell:

   ```
   tux > sudo nvmetcli
   ```

2. Create a new port:

   ```
   (nvmetcli)> cd ports
   (nvmetcli)> create 1
   (nvmetcli)> ls 1/
   o- 1
     o- referrals
     o- subsystems
   ```

3. Create an NVMe subsystem:

   ```
   (nvmetcli)> cd /subsystems
   ```

```
(nvmetcli)> create nqn.2014-08.org.nvmexpress:NVMf:uuid:c36f2c23-354d-416c-95de-
f2b8ec353a82
(nvmetcli)> cd nqn.2014-08.org.nvmexpress:NVMf:uuid:c36f2c23-354d-416c-95de-
f2b8ec353a82/
(nvmetcli)> ls
o- nqn.2014-08.org.nvmexpress:NVMf:uuid:c36f2c23-354d-416c-95de-f2b8ec353a82
  o- allowed_hosts
  o- namespaces
```

4. Create a new namespace and set an NVMe device to it.

```
(nvmetcli)> cd namespaces
(nvmetcli)> create 1
(nvmetcli)> cd 1
(nvmetcli)> set device path=/dev/nvme0n1
Parameter path is now '/dev/nvme0n1'.
```

5. Enable the previously created namespace:

```
(nvmetcli)> cd ..
(nvmetcli)> enable
The Namespace has been enabled.
```

6. Display the created namespace:

```
(nvmetcli)> cd ..
(nvmetcli)> ls
o- nqn.2014-08.org.nvmexpress:NVMf:uuid:c36f2c23-354d-416c-95de-f2b8ec353a82
  o- allowed_hosts
  o- namespaces
    o- 1
```

7. Allow all hosts to use the subsystem. Only do this in secure environments.

```
(nvmetcli)> set attr allow_any_host=1
Parameter allow_any_host is now '1'.
```

   Alternatively, you can allow only specific hosts to connect:

```
(nvmetcli)> cd nqn.2014-08.org.nvmexpress:NVMf:uuid:c36f2c23-354d-416c-95de-
f2b8ec353a82/allowed_hosts/
(nvmetcli)> create hostnqn
```

8. List all created objects:

```
(nvmetcli)> cd /
```

```
(nvmetcli)> ls
o- /
  o- hosts
  o- ports
  | o- 1
  |   o- referrals
  |   o- subsystems
  o- subsystems
    o- nqn.2014-08.org.nvmexpress:NVMf:uuid:c36f2c23-354d-416c-95de-f2b8ec353a82
      o- allowed_hosts
      o- namespaces
        o- 1
```

9. Make the target available via RDMA:

```
(nvmetcli)> cd ports/1/
(nvmetcli)> set addr adrfam=ipv4 trtype=rdma traddr=10.0.0.1 trsvcid=4420
Parameter trtype is now 'rdma'.
Parameter adrfam is now 'ipv4'.
Parameter trsvcid is now '4420'.
Parameter traddr is now '10.0.0.1'.
```

Alternatively, you can make it available with Fibre Channel:

```
(nvmetcli)> cd ports/1/
(nvmetcli)> set addr adrfam=fc trtype=fc
 traddr=nn-0x1000000044001123:pn-0x2000000055001123 trsvcid=none
```

16.3.3 Back Up and Restore Target Configuration

You can save the target configuration in a JSON file with the following commands:

```
tux > sudo nvmetcli
(nvmetcli)> saveconfig nvme-target-backup.json
```

To restore the configuration, use:

```
(nvmetcli)> restore nvme-target-backup.json
```

You can also wipe the current configuration:

```
(nvmetcli)> clear
```

16.4 Special Hardware Configuration

16.4.1 Overview

Some hardware needs special configuration to work correctly. Skim the titles of the following sections to see if you are using any of the mentioned devices or vendors.

16.4.2 Broadcom

If you are using the *Broadcom Emulex LightPulse Fibre Channel SCSI* driver, add a Kernel configuration parameter on the target and host for the `lpfc` module:

```
tux > sudo echo "options lpfc lpfc_enable_fc4_type=3" > /etc/modprobe.d/lpfc.conf
```

Make sure that the Broadcom adapter firmware has at least version 11.2.156.27. Also make sure that you have the current versions of `nvme-cli`, `nvmetlci` and the Kernel installed.

To enable a Fibre Channel port as an NVMe target, set a module parameter `lpfc_enable_nvmet= COMMA_SEPARATED_WWPNS`. Only listed WWPNs will be configured for target mode. A Fibre Channel port can either be configured as target *or* as initiator.

16.4.3 Marvell

FC-NVMe is supported on QLE269x and QLE27xx adapters. FC-NVMe support is enabled by default in the Marvell® QLogic® QLA2xxx Fibre Channel driver.

To confirm NVMe is enabled, run the following command:

```
tux > cat /sys/module/qla2xxx/parameters/ql2xnvmeenable
```

A resulting `1` suggests NVMe is enabled, a `0` indicates it is disabled.

Next, ensure that the Marvell adapter firmware has at least version 8.08.204 by checking the output of the following command:

```
tux > cat /sys/class/scsi_host/host0/fw_version
```

Last, ensure that the latest versions available for SUSE Linux Enterprise Server of `nvme-cli`, `QConvergeConsoleCLI`, and the Kernel are installed. You can, for example, run

```
root # zypper lu && zypper pchk
```

to check for updates and patches.

For more details on installation, please refer to the FC-NVMe sections of the following Marvell user guides:

- http://driverdownloads.qlogic.com/QLogicDriverDownloads_UI/ShowEula.aspx?resourceid=32769&docid=96728&ProductCategory=39&Product=1259&Os=126
- http://driverdownloads.qlogic.com/QLogicDriverDownloads_UI/ShowEula.aspx?resourceid=32761&docid=96726&ProductCategory=39&Product=1261&Os=126

16.5 More Information

For more details about the abilities of the **`nvme`** command, refer to **`nvme nvme-help`**.

The following links provide a basic introduction to NVMe and NVMe-oF:

- http://nvmexpress.org/
- http://www.nvmexpress.org/wp-content/uploads/NVMe_Over_Fabrics.pdf
- https://storpool.com/blog/demystifying-what-is-nvmeof

17 Managing multipath I/O for devices

This section describes how to manage failover and path load balancing for multiple paths between the servers and block storage devices by using Multipath I/O (MPIO).

17.1 Understanding multipath I/O

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.

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.

Multipathing provides fault tolerance against connection failures, but not against failures of the storage device itself. The latter is achieved with complementary techniques like mirroring.

17.1.1 Multipath terminology

Storage array

A hardware device with many disks and multiple fabrics connections (controllers) that provides SAN storage to clients. Storage arrays typically have RAID and failover features and support multipathing. Historically, active/passive (failover) and active/active (load-balancing) storage array configurations were distinguished. These concepts still exist but they are merely special cases of the concepts of path groups and access states supported by modern hardware.

Host, host system

The computer running SUSE Linux Enterprise Server which acts as a client system for a *storage array*.

Multipath map, multipath device

A set of *path devices*. It represents a storage volume on a storage array and is seen as a single block device by the host system.

Path device

A member of a multipath map, typically a SCSI device. Each path device represents a unique connection between the host computer and the actual storage volume, for example, a logical unit from an iSCSI session.

WWID

"World Wide Identifier". `multipath-tools` uses the WWID to determine which low-level devices should be assembled into a multipath map. The WWID must be distinguished from the configurable *map name* (see *Section 17.12, "Multipath device names and WWIDs"*).

uevent, udev event

An event sent by the kernel to user space and processed by the **`udev`** subsystem. Uevents are generated when devices are added, removed, or change their properties.

Device mapper

A framework in the Linux kernel for creating virtual block devices. I/O operations to mapped devices are redirected to the underlying block devices. Device mappings may be stacked. The device mapper implements its own event signaling, also known as "device mapper events" or "dm events".

initramfs

The initial RAM file system, also referred to as "initial RAM disk" (initrd) for historical reasons (see *Book "Administration Guide", Chapter 11 "Introduction to the boot process", Section 11.1 "Terminology"*).

ALUA

"Asymmetric Logical Unit Access", a concept introduced with the SCSI standard SCSI-3. Storage volumes can be accessed via multiple ports, which are organized in port groups with different states (active, standby, etc.). ALUA defines SCSI commands to query the port groups and their states and change the state of a port group. Modern storage arrays that support SCSI usually support ALUA, too.

17.2 Hardware support

The multipathing drivers and tools are available on all architectures supported by SUSE Linux Enterprise Server. The generic, protocol-agnostic driver works with most multipath-capable storage hardware on the market. Some storage array vendors provide their own multipathing management tools. Consult the vendor's hardware documentation to determine what settings are required.

17.2.1 Multipath implementations: device mapper and NVMe

The traditional, generic implementation of multipathing under Linux uses the device mapper framework. For most device types like SCSI devices, device mapper multipathing is the only available implementation. Device mapper multipath is highly configurable and flexible.

The Linux *NVM Express* (NVMe) kernel subsystem implements multipathing natively in the kernel. This implementation creates less computational overhead for NVMe devices, which are typically fast devices with very low latencies. Native NVMe multipathing requires no user space component. For details, refer to *Section 16.2.4, "Multipathing"*.

This chapter documents device mapper multipath and its user-space component, `multipath-tools`. `multipath-tools` also has limited support for native NVMe multipathing (see *Section 17.13, "Miscellaneous options"*).

17.2.2 Storage array autodetection for multipathing

Device mapper multipath is a generic technology. Multipath device detection requires only that the low-level (for example, SCSI) devices are detected by the kernel, and that device properties reliably identify multiple low-level devices as being different "paths" to the same volume rather than actually different devices.

The `multipath-tools` package detects storage arrays by their vendor and product names. It provides built-in configuration defaults for a large variety of storage products. Consult the hardware documentation of your storage array: some vendors provide specific recommendations for Linux multipathing configuration.

If you need to apply changes to the built-in configuration for your storage array, read *Section 17.8, "Multipath configuration"*.

Important: Disclaimer about built-in hardware properties

`multipath-tools` has built-in presets for many storage arrays. The existence of such presets for a given storage product *does not imply* that the vendor of the storage product has tested the product with `dm-multipath`, nor that the vendor endorses or supports the use of `dm-multipath` with the product. Always consult the original vendor documentation for support-related questions.

17.2.3 Storage arrays that require specific hardware handlers

Some storage arrays require special commands for failover from one path to the other, or non-standard error-handling methods. These special commands and methods are implemented by hardware handlers in the Linux kernel. Modern SCSI storage arrays support the "Asymmetric Logical Unit Access" (ALUA) hardware handler defined in the SCSI standard. Besides ALUA, the SLE kernel contains hardware handlers for Netapp E-Series (RDAC), the Dell/EMC CLARiiON CX family of arrays, and legacy arrays from HP.

Since Linux kernel 4.4, the Linux kernel has automatically detected hardware handlers for most arrays, including all arrays supporting ALUA. The only requirement is that the device handler modules are loaded at the time the respective devices are probed. The `multipath-tools` package ensures this by installing appropriate configuration files. Once a device handler is attached to a given device, it cannot be changed anymore.

17.3 Planning for multipathing

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

17.3.1 Prerequisites

- The storage array you use for the multipathed device must support multipathing. For more information, see *Section 17.2, "Hardware support"*.
- 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.

- 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.
- 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.

17.3.2 Multipath installation types

We distinguish installation types by the way the root device is handled. *Section 17.4, "Installing SUSE Linux Enterprise Server on multipath systems"* describes how the different setups are created during and after installation.

17.3.2.1 Root file system on multipath (SAN-boot)

The root file system is on a multipath device. This is typically the case for diskless servers that use SAN storage exclusively. On such systems, multipath support is required for booting, and multipathing must be enabled in the initramfs.

17.3.2.2 Root file system on a local disk

The root file system (and possibly some other file systems) is on local storage, for example, on a directly attached SATA disk or local RAID, but the system additionally uses file systems in the multipath SAN storage. This system type can be configured in three different ways:

Multipath setup for local disk

All block devices are part of multipath maps, including the local disk. The root device appears as a degraded multipath map with just one path. This configuration is created if multipathing was enabled during the initial system installation with YaST.

Local disk is excluded from multipath

In this configuration, multipathing is enabled in the initramfs, but the root device is explicitly excluded from multipath (see *Section 17.11.1, "The `blacklist` section in `multipath.conf`"*). *Procedure 17.1, "Disabling multipathing for the root disk after installation"* describes how to set up this configuration.

Multipath disabled in the initramfs

This setup is created if multipathing was not enabled during the initial system installation with YaST. This configuration is rather fragile; consider using one of the other options instead.

17.3.3 Disk management tasks

Use third-party SAN array management tools or the user interface of your storage array to create logical devices and assign them to hosts. Make sure to configure the host credentials correctly on both sides.

You can add or remove volumes to a running host, but detecting the changes may require rescanning SCSI targets and reconfiguring multipathing on the host. See *Section 17.14.6, "Scanning for new devices without rebooting"*.

Note: Storage processors

On some disk arrays, the storage array manages the traffic through storage processors. One processor is active and the other one is passive until there is a failure. If you are connected to the passive storage processor, you might not see the expected LUNs, or you might see the LUNs but encounter I/O errors when you try to access them.

If a disk array has more than one storage processor, ensure that the SAN switch has a connection to the active storage processor that owns the LUNs you want to access.

17.3.4 Software RAID and complex storage stacks

Multipathing is set up on top of basic storage devices such as SCSI disks. In a multi-layered storage stack, multipathing is always the bottom layer. Other layers such as software RAID, Logical Volume Management, block device encryption, etc. are layered on top of it. Therefore, 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.

17.3.5 High-availability solutions

High-availability solutions for clustering storage resources run on top of the multipathing service on each node. Make sure that the configuration settings in the `/etc/multipath.conf` file on each node are consistent across the cluster.

Make sure that multipath devices have the same name across all devices. Refer to *Section 17.12, "Multipath device names and WWIDs"* for details.

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

Special care must be taken when using multipathing together with clustering software that relies on shared storage for fencing, such as **`pacemaker`** with **`sbd`**. See *Section 17.9.2, "Queuing policy on clustered servers"* for details.

17.4 Installing SUSE Linux Enterprise Server on multipath systems

No special installation parameters are required for the installation of SUSE Linux Enterprise Server on systems with multipath hardware.

17.4.1 Installing without connected multipath devices

You may want to perform installation on a local disk, without configuring the fabric and the storage first, with the intention to add multipath SAN devices to the system later. In this case, the installation will proceed like on a non-multipath system. After installation, `multipath-tools` will be installed, but the `systemd` service `multipathd.service` will be disabled. The system will be configured as described in *Multipath disabled in the initramfs* in *Section 17.3.2.2, "Root file system on a local disk"*. Before adding SAN hardware, you will need to enable and start `multipathd.service`. We recommend creating a `blacklist` entry in the `/etc/multipath.conf` for the root device (see *Section 17.11.1, "The `blacklist` section in `multipath.conf`"*).

17.4.2 Installing with connected multipath devices

If multipath devices are connected to the system at installation time, YaST will detect them and display a pop-up window asking you whether multipath should be enabled before entering the partitioning stage.

If you select “No” at this prompt (not recommended), the installation will proceed as in *Section 17.4.1, “Installing without connected multipath devices”*. In the partitioning stage, do not use/edit devices that will later be part of a multipath map.

If you select “Yes” at the multipath prompt, **`multipathd`** will run during the installation. No device will be added to the `blacklist` section of `/etc/multipath.conf`, thus all SCSI and DASD devices, including local disks, will appear as multipath devices in the partitioning dialogs. After installation, all SCSI and DASD devices will be multipath devices, as described in *Section 17.3.2.1, “Root file system on multipath (SAN-boot)”*.

PROCEDURE 17.1: DISABLING MULTIPATHING FOR THE ROOT DISK AFTER INSTALLATION

This procedure assumes that you installed on a local disk and enabled multipathing during installation, so that the root device is on multipath now, but you prefer to set up the system as described in *Local disk is excluded from multipath* in *Section 17.3.2.2, “Root file system on a local disk”*.

1. Check your system for `/dev/mapper/...` references to your local root device, and replace them with references that will still work if the device is not a multipath map anymore (see *Section 17.12.4, “Referring to multipath maps”*). If the following command finds no references, you do not need to apply changes:

```
tux > sudo grep -rl /dev/mapper/ /etc
```

2. Switch to `by-uuid` persistent device policy for **`dracut`** (see *Section 17.7.4.2, “Persistent device names in the initramfs”*):

```
tux > echo 'persistent_policy="by-uuid"' | \
      sudo tee /etc/dracut.conf.d/10-persistent-policy.conf
```

3. Determine the WWID of the root device:

```
tux > multipathd show paths format "%i %d %w %s"
0:2:0:0 sda 3600605b009e7ed501f0e45370aaeb77f IBM,ServeRAID M5210
...
```

This command prints all paths devices with their WWIDs and vendor/product information. You will be able to identify the root device (here, the ServeRAID device) and note the WWID.

4. Create a blacklist entry in `/etc/multipath.conf` (see *Section 17.11.1, "The `blacklist` section in `multipath.conf`"*) with the WWID you just determined (do *not* apply these settings just yet):

```
blacklist {
    wwid 3600605b009e7ed501f0e45370aaeb77f
}
```

5. Rebuild the initramfs:

```
tux > sudo dracut -f
```

6. Reboot. Your system should boot with a non-multipath root disk.

17.5 Updating SLE on multipath systems

When updating your system online, you can proceed as described in *Book "Deployment Guide", Chapter 22 "Upgrading Online"*.

The offline update of your system is similar to the fresh installation as described in *Section 17.4, "Installing SUSE Linux Enterprise Server on multipath systems"*. There is no `blacklist`, thus if the user selects to enable multipath, the root device will appear as a multipath device, even if it is normally not. When **`dracut`** builds the initramfs during the update procedure, it sees a different storage stack as it would see on the booted system. See *Section 17.7.4.2, "Persistent device names in the initramfs"* and *Section 17.12.4, "Referring to multipath maps"*.

17.6 Multipath management tools

The multipathing support in SUSE Linux Enterprise Server is based on the Device Mapper Multipath module of the Linux kernel and the `multipath-tools` user space package.

The generic multipathing capability is handled by the Device Mapper Multipath (DM-MP) module. For details, refer to *Section 17.6.1, "Device mapper multipath module"*.

The packages `multipath-tools` and `kpartx` provide tools that handle automatic path discovery and grouping. The tools are the following:

`multipathd`

The daemon to set up and monitor multipath maps, and a command-line client to communicate with the daemon process. See *Section 17.6.2, "The* **`multipathd`** *daemon"*.

`multipath`

The command-line tool for multipath operations. See *Section 17.6.3, "The* **`multipath`** *command"*.

`kpartx`

The command-line tool for managing "partitions" on multipath devices. See *Section 17.7.3, "Partitions on multipath devices and* **`kpartx`***"*.

`mpathpersist`

The command-line tool for managing SCSI persistent reservations. See *Section 17.6.4, "SCSI persistent reservations and* **`mpathpersist`***"*.

17.6.1 Device mapper multipath module

The Device Mapper Multipath (DM-MP) module `dm-multipath.ko` provides the generic multipathing capability for Linux. DM-MPIO is the preferred solution for multipathing on SUSE Linux Enterprise Server for SCSI and DASD devices, and can be used for NVMe devices as well.

Note: Using DM-MP for NVMe devices

To disable native NVMe multipathing and use device mapper multipath instead (*not recommended*), boot with the kernel parameter `nvme-core.multipath=0`.

The Device Mapper Multipath module handles the following tasks:

- Distributing load over multiple paths inside the active path group.
- Noticing I/O errors on path devices, and marking these as failed, so that no I/O will be sent to them.

- Switching path groups when all paths in the active path group have failed.
- Either failing or queuing I/O on the multipath device if all paths have failed, depending on configuration.

The following tasks are handled by the user-space components in the `multipath-tools` package, not by the Device Mapper Multipath module:

- Discovering devices representing different paths to the same storage device and assembling multipath maps from them.
- Collecting path devices with similar properties into path groups.
- Monitoring path devices actively for failure or reinstantiation.
- Monitoring of addition and removal of path devices.
- The Device Mapper Multipath module does not provide an easy-to-use user interface for setup and configuration.

For details about the components from the `multipath-tools` package, refer to *Section 17.6.2, "The* **`multipathd`** *daemon"*.

Note: Failures that multipath prevents

DM-MPIO protects against failures in the paths to the device, and not failures in the device itself, such as media errors. The latter kind of errors must be prevented by other means, such as replication.

17.6.2 The **`multipathd`** daemon

`multipathd` is the most important part of a modern Linux device mapper multipath setup. It is normally started through the systemd service `multipathd.service` (see *Section 17.7.1, "Enabling, starting, and stopping multipath services"*).

`multipathd` serves the following tasks (some of them depend on the configuration):

- On startup, detects path devices and sets up multipath maps from detected devices.
- Monitors uevents and device mapper events, adding or removing path mappings to multipath maps as necessary and initiating failover or failback operations.

- Sets up new maps on the fly when new path devices are discovered.
- Checks path devices at regular intervals to detect failure, and tests failed paths to reinstate them if they become operational again.
- When all paths fail, **multipathd** either fails the map, or switches the map device to queuing mode for a given time interval.
- Handles path state changes and switches path groups or regroups paths, as necessary.
- Tests paths for "marginal" state, i.e. shaky fabrics conditions that cause path state flipping between operational and non-operational.
- Handles SCSI persistent reservation keys for path devices if configured. See *Section 17.6.4, "SCSI persistent reservations and* **mpathpersist**".

multipathd also serves as a command-line client to process interactive commands by sending them to the running daemon. The general syntax to send commands to the daemon is as follows:

```
tux > sudo multipathd COMMAND
```

or

```
tux > sudo multipathd -k'COMMAND'
```

There is also an interactive mode that allows sending multiple subsequent commands:

```
tux > sudo multipathd -k
```

Note: How multipath and multipathd work together

Many **multipathd** commands have **multipath** equivalents. For example, **multipathd show topology** does the same thing as **multipath -ll**. The notable difference is that the multipathd command inquires the internal state of the running **multipathd** daemon, whereas multipath obtains information directly from the kernel and I/O operations.

If the multipath daemon is running, we recommend making modifications to the system by using the **multipathd** commands. Otherwise, the daemon may notice configuration changes and react to them. In some situations, the daemon might even try to undo the applied changes. **multipath** automatically delegates certain possibly dangerous commands, like destroying and flushing maps, to **multipathd** if a running daemon is detected.

The list below describes frequently used **multipathd** commands:

show topology
: Shows the current map topology and properties. See *Section 17.14.2, "Interpreting multipath I/O status"*.

show paths
: Shows the currently known path devices.

show paths format "*FORMAT STRING***"**
: Shows the currently known path devices using a format string. Use **show wildcards** to see a list of supported format specifiers.

show maps
: Shows the currently configured map devices.

show maps format *FORMAT STRING*
: Shows the currently configured map devices using a format string. Use **show wildcards** to see a list of supported format specifiers.

show config local
: Shows the current configuration that multipathd is using.

reconfigure
: Rereads configuration files, rescans devices, and sets up maps again. This is basically equivalent to a restart of **multipathd**. A few options cannot be modified without a restart. They are mentioned in the man page `multipath.conf(5)`. The `reconfigure` command reloads only map devices that have changed in some way.

del map *MAP DEVICE NAME*
: Unconfigure and delete the given map device and its partitions. *MAP DEVICE NAME* can be a device node name like `dm-0`, a WWID, or a map name. The command fails if the device is in use.

switchgroup map *MAP DEVICE NAME* **group** *N*
: Switch to the path group with the given numeric index (starting at 1). This is useful for maps with manual failback (see *Section 17.9, "Configuring policies for failover, queuing, and failback"*).

Additional commands are available to modify path states, enable or disable queuing, and more. See `multipathd(8)` for details.

17.6.3 The `multipath` command

Even though multipath setup is mostly automatic and handled by `multipathd`, `multipath` is still useful for some administration tasks. Several examples of the command usage follow:

multipath

Detects path devices and configures all multipath maps that it finds.

multipath -d

Similar to `multipath`, but does not set up any maps ("dry run").

multipath *DEVICENAME*

Configures a specific multipath device. *DEVICENAME* can denote a member path device by its device node name (`/dev/sdb`) or device number in `major:minor` format. Alternatively, it can be the WWID or name of a multipath map.

multipath -f *DEVICENAME*

Unconfigures ("flushes") a multipath map and its partition mappings. The command will fail if the map or one of its partitions is in use. See above for possible values of *DEVICENAME*.

multipath -F

Unconfigures ("flushes") all multipath maps and their partition mappings. The command will fail for maps in use.

multipath -ll

Displays the status and topology of all currently configured multipath devices. See *Section 17.14.2, "Interpreting multipath I/O status"*.

multipath -ll *DEVICENAME*

Displays the status of a specified multipath device. See above for possible values of *DEVICENAME*.

multipath -t

Shows the internal hardware table and the active configuration of multipath. Refer to `multipath.conf(5)` for details about the configuration parameters.

multipath -T

Has a similar function as the `multipath -t` command but shows only hardware entries matching the hardware detected on the host.

The option `-v` controls the verbosity of the output. The provided value overrides the `verbosity` option in `/etc/multipath.conf`. See *Section 17.13, "Miscellaneous options"*.

17.6.4 SCSI persistent reservations and `mpathpersist`

The **`mpathpersist`** utility is used to manage SCSI persistent reservations on Device Mapper Multipath devices. Persistent reservations serve to restrict access to SCSI Logical Units to certain SCSI initiators. In multipath configurations, it is important to use the same reservation keys for all I_T nexuses (paths) for a given volume; otherwise, creating a reservation on one path device would cause I/O errors on other paths.

Use this utility with the `reservation_key` attribute in the `/etc/multipath.conf` file to set persistent reservations for SCSI devices. If (and only if) this option is set, the **`multipathd`** daemon checks persistent reservations for newly discovered paths or reinstated paths.

You can add the attribute to the `defaults` section or the `multipaths` section of `multipath.conf`. For example:

```
multipaths {
    multipath {
        wwid             3600140508dbcf02acb448188d73ec97d
        alias            yellow
        reservation_key  0x123abc
    }
}
```

After setting the `reservation_key` parameter for all mpath devices applicable for persistent management, reload the configuration using **`multipathd reconfigure`**.

Note: Using "`reservation_key file`"

If the special value `reservation_key file` is used in the `defaults` section of `multipath.conf`, reservation keys can be managed dynamically in the file `/etc/multipath/prkeys` using **`mpathpersist`**.

This is the recommended way to handle persistent reservations with multipath maps.

Use the command **`mpathpersist`** to query and set persistent reservations for multipath maps consisting of SCSI devices. Refer to the manual page `mpathpersist(8)` for details. The command-line options are the same as those of the **`sg_persist`** from the `sg3_utils` package. The `sg_persist(8)` manual page explains the semantics of the options in detail.

In the following examples, *`DEVICE`* denotes a device mapper multipath device like `/dev/mapper/mpatha`. The commands below are listed with long options for better readability. All options have single-letter replacements, like in **`mpathpersist -oGS 123abc`** *`DEVICE`*.

mpathpersist --in --read-keys *DEVICE*

Read the registered reservation keys for the device.

mpathpersist --in --read-reservation *DEVICE*

Show existing reservations for the device.

mpathpersist --out --register --param-sark=123abc *DEVICE*

Register a reservation key for the device. This will add the reservation key for all I_T nexuses (path devices) on the host.

mpathpersist --out --reserve --param-rk=123abc --prout-type=5 *DEVICE*

Create a reservation of type 5 (“write exclusive - registrants only”) for the device, using the previously registered key.

mpathpersist --out --release --param-rk=123abc --prout-type=5 *DEVICE*

Release a reservation of type 5 for the device.

mpathpersist --out --register-ignore --param-sark=0 *DEVICE*

Delete a previously existing reservation key from the device.

17.7 Configuring the system for multipathing

17.7.1 Enabling, starting, and stopping multipath services

To enable multipath services to start at boot time, run the following command:

```
tux > sudo systemctl enable multipathd
```

To manually start the service in the running system, enter:

```
tux > sudo systemctl start multipathd
```

To restart the service, enter:

```
tux > sudo systemctl restart multipathd
```

In most situations, restarting the service is not necessary. To simply have **multipathd** reload its configuration, run:

```
tux > sudo systemctl reload multipathd
```

To check the status of the service, enter:

```
tux > sudo systemctl status multipathd
```

To stop the multipath services in the current session, run:

```
tux > sudo systemctl stop multipathd multipathd.socket
```

Stopping the service does not remove existing multipath maps. To remove unused maps, run:

```
tux > sudo multipath -F
```

Warning: Keep `multipathd.service` enabled

We strongly recommend keeping `multipathd.service` always enabled and running on systems with multipath hardware. The service does support **`systemd`**'s socket activation mechanism, but it is discouraged to rely on that. Multipath maps will not be set up during boot if the service is disabled.

Note: Disabling multipath

If you need to disable multipath despite the warning above, for example, because a third-party multipathing software is going to be deployed, proceed as follows. Be sure that the system uses no hard-coded references to multipath devices (see *Section 17.15.2, "Understanding device referencing issues"*).

To disable multipathing *just for a single system boot*, use the kernel parameter `multipath=off`. This affects both the booted system and the initramfs, which does not need to be rebuilt in this case.

To disable multipathd services *permanently*, so that they will not be started on future system boots, run the following commands:

```
tux > sudo systemctl disable multipathd multipathd.socket
tux > sudo dracut --force --omit multipath
```

(Whenever you disable or enable the multipath services, rebuild the `initramfs`. See *Section 17.7.4, "Keeping the initramfs synchronized"*.)

If you want to make sure multipath devices do not get set up, *even when running* **multipath** *manually*, add the following lines at the end of `/etc/multipath.conf` before rebuilding the initramfs:

```
blacklist {
    wwid .*
}
```

17.7.2 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.
- If SUSE Linux Enterprise Server ships no driver for the host bus adapter (HBA), install a Linux driver from the HBA vendor. See the vendor's specific instructions for more details.

If multipath devices are detected and `multipathd.service` is enabled, multipath maps should be created automatically. If this does not happen, *Section 17.15.3, "Troubleshooting steps in emergency mode"* lists some shell commands that can be used to examine the situation. When the LUNs are not seen by the HBA driver, check the zoning setup in the SAN. In particular, check whether LUN masking is active and whether the LUNs are correctly assigned to the server.

If the HBA driver can see LUNs, but no corresponding block devices are created, additional kernel parameters may be needed. See *TID 3955167: Troubleshooting SCSI (LUN) Scanning Issues* in the SUSE Knowledge base at https://www.suse.com/support/kb/doc.php?id=3955167.

17.7.3 Partitions on multipath devices and **kpartx**

Multipath maps can have partitions like their path devices. Partition table scanning and device node creation for partitions is done in user space by the **kpartx** tool. **kpartx** is automatically invoked by udev rules; there is usually no need to run it manually. See *Section 17.12.4, "Referring to multipath maps"* for ways to refer to multipath partitions.

Note: Disabling invocation of **kpartx**

The `skip_kpartx` option in `/etc/multipath.conf` can be used to disable invocation of **`kpartx`** on selected multipath maps. This may be useful on virtualization hosts, for example.

Partition tables and partitions on multipath devices can be manipulated as usual, using YaST or tools like **`fdisk`** or **`parted`**. Changes applied to the partition table will be noted by the system when the partitioning tool exits. If this does not work (usually because a device is busy), try **`multipathd reconfigure`**, or reboot the system.

17.7.4 Keeping the initramfs synchronized

Important

Make sure that the initial RAM file system (initramfs) and the booted system behave consistently regarding the use of multipathing for all block devices. Rebuild the initramfs after applying multipath configuration changes.

If multipathing is enabled in the system, it also needs to be enabled in the `initramfs` and vice versa. The only exception to this rule is the option *Multipath disabled in the initramfs* in *Section 17.3.2.2, "Root file system on a local disk"*.

The multipath configuration must be synchronized between the booted system and the initramfs. Therefore, if you change any of the files: `/etc/multipath.conf`, `/etc/multipath/wwids`, `/etc/multipath/bindings`, or other configuration files, or udev rules related to device identification, rebuild initramfs using the command:

```
tux > sudo dracut -f
```

If the `initramfs` and the system are not synchronized, the system will not boot properly, and the start-up procedure may result in an emergency shell. See *Section 17.15, "Troubleshooting MPIO"* for instructions on how to avoid or repair such a scenario.

17.7.4.1 Enabling or disabling multipathing in the initramfs

Special care must be taken if the initramfs is rebuilt in non-standard situations, for example, from a rescue system or after booting with the kernel parameter `multipath=off`. **`dracut`** will automatically include multipathing support in the initramfs if and only if it detects that the root file system is on a multipath device while the initramfs is being built. In such cases, it is necessary to enable or disable multipathing explicitly.

To enable multipath support in the `initramfs`, run the command:

```
tux > sudo dracut --force --add multipath
```

To disable multipath support in `initramfs`, run the command:

```
tux > sudo dracut --force --omit multipath
```

17.7.4.2 Persistent device names in the initramfs

When **`dracut`** generates the initramfs, it must refer to disks and partitions to be mounted in a persistent manner, to make sure the system will boot correctly. When **`dracut`** detects multipath devices, it will use the DM-MP device names such as

```
/dev/mapper/3600a098000aad73f00000a3f5a275dc8-part1
```

for this purpose by default. This is good if the system *always* runs in multipath mode. But if the system is started without multipathing, as described in *Section 17.7.4.1, "Enabling or disabling multipathing in the initramfs"*, booting with such an initramfs will fail, because the `/dev/mapper` devices will not exist. See *Section 17.12.4, "Referring to multipath maps"* for another possible problem scenario, and some background information.

To prevent this from happening, change **`dracut`**'s persistent device naming policy by using the `--persistent-policy` option. We recommend setting the `by-uuid` use policy:

```
tux > sudo dracut --force --omit multipath --persistent-policy=by-uuid
```

See also *Procedure 17.1, "Disabling multipathing for the root disk after installation"* and *Section 17.15.2, "Understanding device referencing issues"*.

17.8 Multipath configuration

The built-in `multipath-tools` defaults work well for most setups. If customizations are needed, a configuration file needs to be created. The main configuration file is `/etc/multipath.conf`. In addition, files in `/etc/multipath/conf.d/` are taken into account. See *Section 17.8.2.1, "Additional configuration files and precedence rules"* for additional information.

Important: Vendor recommendations and built-in hardware defaults

Some storage vendors publish recommended values for multipath options in their documentation. These values often represent what the vendor has tested in his environment and found most suitable for the storage product. See the disclaimer in *Section 17.2.2, "Storage array autodetection for multipathing"*.

`multipath-tools` has built-in defaults for many storage arrays that are derived from the published vendor recommendations. Run **`multipath -T`** to see the current settings for your devices and compare them to vendor recommendations.

17.8.1 Creating `/etc/multipath.conf`

It is recommended that you create a minimal `/etc/multipath.conf` that just contains those settings you want to change. In many cases, you do not need to create `/etc/multipath.conf` at all.

If you prefer working with a configuration template that contains all possible configuration directives, run:

```
multipath -T >/etc/multipath.conf
```

See also *Section 17.14.1, "Best practices for configuration"*.

17.8.2 `multipath.conf` syntax

The `/etc/multipath.conf` file uses a hierarchy of sections, subsections, and option/value pairs.

- White space separates tokens. Consecutive white space characters are collapsed into a single space, unless quoted (see below).
- The hash (`#`) and exclamation mark (`!`) characters cause the rest of the line to be discarded as a comment.
- Sections and subsections are started with a section name and an opening brace (`{`) on the same line, and end with a closing brace (`}`) on a line on its own.
- Options and values are written on one line. Line continuations are unsupported.
- Options and section names must be keywords. The allowed keywords are documented in `multipath.conf(5)` .
- Values may be enclosed in double quotes (`"`). They must be enclosed in quotes if they contain white space or comment characters. A double quote character inside a value is represented by a pair of double quotes (`""`).
- The values of some options are POSIX regular expressions (see `regex(7)`). They are case sensitive and not anchored, so " `bar` " matches " `rhabarber` ", but not "Barbie".

The following example illustrates the syntax:

```
section {
    subsection {
        option1 value
        option2      "complex value!"
        option3    "value with ""quoted"" word"
    } ! subsection end
} # section end
```

17.8.2.1 Additional configuration files and precedence rules

After `/etc/multipath.conf` , the tools read files matching the pattern `/etc/multipath.conf.d/*.conf` . The additional files follow the same syntax rules as `/etc/multipath.conf` . Sections and options can occur multiple times. If *the same option in the same section* is set in multiple files, or on multiple lines in the same file, the last value takes precedence. Separate precedence rules apply between `multipath.conf` sections, see below.

17.8.3 `multipath.conf` sections

The `/etc/multipath.conf` file is organized into the following sections. Some options can occur in more than one section. See `multipath.conf(5)` for details.

defaults

General default settings.

> **Important: Overriding built-in device properties**
>
> Built-in hardware-specific device properties take precedence over the settings in the `defaults` section. Changes must therefore be made in the `devices` section or in the `overrides` section.

blacklist

Lists devices to ignore. See *Section 17.11.1, "The `blacklist` section in `multipath.conf`"*.

blacklist_exceptions

Lists devices to be multipathed even though they are matched by the blacklist. See *Section 17.11.1, "The `blacklist` section in `multipath.conf`"*.

devices

Settings specific to the storage controller. This section is a collection of `device` subsections. Values in this section override values for the same options in the `defaults` section, and the built-in settings of `multipath-tools`.

`device` entries in the `devices` section are matched against the vendor and product of a device using regular expressions. These entries will be "merged", setting all options from matching sections for the device. If the same option is set in multiple matching `device` sections, the last device entry takes precedence, even if it is less "specific" than preceding entries. This applies also if the matching entries appear in different configuration files (see *Section 17.8.2.1, "Additional configuration files and precedence rules"*). In the following example, a device `SOMECORP STORAGE` will use `fast_io_fail_tmo 15`.

```
devices {
    device {
        vendor SOMECORP
        product STOR
        fast_io_fail_tmo 10
    }
```

```
    device {
        vendor SOMECORP
        product .*
        fast_io_fail_tmo 15
    }
}
```

multipaths

Settings for individual multipath devices. This section is a list of `multipath` subsections. Values override the `defaults` and `devices` sections.

overrides

Settings that override values from all other sections.

17.8.4 Applying `multipath.conf` modifications

To apply the configuration changes, run:

```
tux > sudo multipathd reconfigure
```

Do not forget to synchronize with the configuration in the initramfs. See *Section 17.7.4, "Keeping the initramfs synchronized"*.

Warning: Do not apply settings using `multipath`

Do not apply new settings with the `multipath` command while `multipathd` is running. This may result in an inconsistent and possibly broken setup.

Note: Verifying a modified setup

It is possible to test modified settings first before they are applied, by running:

```
tux > sudo multipath -d -v2
```

This command shows new maps to be created with the proposed topology, but not whether maps will be removed/flushed. To obtain more information, run with increased verbosity:

```
tux > sudo multipath -d -v3 2>&1 | less
```

17.9 Configuring policies for failover, queuing, and failback

The goal of multipath I/O is to provide connectivity fault tolerance between the storage system and the server. The desired default behavior depends on whether the server is a stand-alone server or a node in a high-availability cluster.

This section discusses the most important `multipath-tools` configuration parameters for achieving fault tolerance.

polling_interval

The time interval (in seconds) between health checks for path devices. The default is 5 seconds. Failed devices are checked at this time interval. For healthy devices, the time interval may be increased up to `max_polling_interval` seconds.

detect_checker

If this is set to `yes` (default, recommended), **`multipathd`** automatically detects the best path checking algorithm.

path_checker

The algorithm used to check path state. If you need to enable the checker, disable `detect_checker` as follows:

```
defaults {
    detect_checker no
}
```

The following list contains only the most important algorithms. See `multipath.conf(5)` for the full list.

tur

Send TEST UNIT READY command. This is the default for SCSI devices with ALUA support.

directio

Read a device sector using asynchronous I/O (aio).

rdac

Device-specific checker for NetAPP E-Series and similar arrays.

none

No path checking is performed.

checker_timeout

If a device does not respond to a path checker command in the given time, it is considered failed. The default is the kernel's SCSI command timeout for the device (usually 30 seconds).

fast_io_fail_tmo

If an error on the SCSI transport layer is detected (for example on a Fibre Channel remote port), the kernel transport layer waits for this amount of time (in seconds) for the transport to recover. After that, the path device fails with “transport offline” state. This is very useful for multipath, because it allows a quick path failover for a frequently occurring class of errors. The value must match typical time scale for reconfiguration in the fabric. The default value of 5 seconds works well for Fibre Channel. Other transports, like iSCSI, may require longer timeouts.

dev_loss_tmo

If a SCSI transport endpoint (for example a Fibre Channel remote port) is not reachable any more, the kernel waits for this amount of time (in seconds) for the port to reappear until it removes the SCSI device node for good. Device node removal is a complex operation which is prone to race conditions or deadlocks and should best be avoided. We therefore recommend setting this to a high value. The special value `infinity` is supported. The default is 10 minutes. To avoid deadlock situations, **`multipathd`** ensures that I/O queuing (see `no_path_retry`) is stopped before `dev_loss_tmo` expires.

no_path_retry

Determine what happens if all paths of a given multipath map have failed. The possible values are:

fail

Fail I/O on the multipath map. This will cause I/O errors in upper layers such as mounted file systems. The affected file systems, and possibly the entire host, will enter degraded mode.

queue

I/O on the multipath map is queued in the device mapper layer and sent to the device when path devices become available again. This is the safest option to avoid losing data, but it can have negative effects if the path devices do not get reinstated for a long time. Processes reading from the device will hang in uninterruptible sleep (`D`) state. Queued data occupies memory, which becomes unavailable for processes. Eventually, memory will be exhausted.

N

N is a positive integer. Keep the map device in queuing mode for *N* polling intervals. When the time elapses, **multipathd** fails the map device. If polling_interval is 5 seconds and no_path_retry is 6, **multipathd** will queue I/O for approximately 6 * 5s = 30s before failing I/O on the map device.

flush_on_last_del

If set to yes and all path devices of a map are deleted (as opposed to just failed), fail all I/O in the map before removing it. The default is no.

deferred_remove

If set to yes and all path devices of a map are deleted, wait for holders to close the file descriptors for the map device before flushing and removing it. If paths reappear before the last holder closed the map, the deferred remove operation will be cancelled. The default is no.

failback

If a failed path device in an inactive path group recovers, **multipathd** reevaluates the path group priorities of all path groups (see *Section 17.10, "Configuring path grouping and priorities"*). After the reevaluation, the highest-priority path group may be one of the currently inactive path groups. This parameter determines what happens in this situation.

Important: Observe vendor recommendations

The optimal failback policy depends on the property of the storage device. It is therefore strongly encouraged to verify failback settings with the storage vendor.

manual

Nothing happens unless the administrator runs a **multipathd switchgroup** (see *Section 17.6.2, "The* **multipathd** *daemon"*).

immediate

The highest-priority path group is activated immediately. This is often beneficial for performance, especially on stand-alone servers, but it should not be used for arrays on which the change of the path group is a costly operation.

followover

Like `immediate`, but only perform failback when the path that has just become active is the only healthy path in its path group. This is useful for cluster configurations: It keeps a node from automatically failing back when another node requested a failover before.

`N`

`N` is a positive integer. Wait for *`N`* polling intervals before activating the highest priority path group. If the priorities change again during this time, the wait period starts anew.

eh_deadline

Set an approximate upper limit for the time (in seconds) spent in SCSI error handling if devices are unresponsive and SCSI commands time out without error response. When the deadline has elapsed, the kernel will perform a full HBA reset.

After modifying the `/etc/multipath.conf` file, apply your settings as described in *Section 17.8.4, "Applying `multipath.conf` modifications"*.

17.9.1 Queuing policy on stand-alone servers

When you configure multipath I/O for a stand-alone server, a `no_path_retry` setting with value `queue` protects the server operating system from receiving I/O errors as long as possible. It queues messages until a multipath failover occurs. If "infinite" queuing is not desired (see above), select a numeric value that is deemed high enough for the storage paths to recover under ordinary circumstances (see above).

17.9.2 Queuing policy on clustered servers

When you configure multipath I/O for a node in a high-availability cluster, you want multipath to report the I/O failure to trigger the resource failover instead of waiting for a multipath failover to be resolved. In cluster environments, you must modify the `no_path_retry` setting so that the cluster node receives an I/O error in relation to the cluster verification process (recommended to be 50% of the heartbeat tolerance) if the connection is lost to the storage system. In addition, you want the multipath `failback` to be set to `manual` or `followover` to avoid a ping-pong of resources because of path failures.

17.10 Configuring path grouping and priorities

Path devices in multipath maps are grouped in *path groups*, also called *priority groups*. Only one path group receives I/O at any given time. **`multipathd`** assigns *priorities* to path groups. Out of the path groups with active paths, the group with the highest priority is activated according to the configured failback policy for the map (see *Section 17.9, "Configuring policies for failover, queuing, and failback"*). The priority of a path group is the average of the priorities of the active path devices in the path group. The priority of a path device is an integer value calculated from the device properties (see the description of the `prio` option below).

This section describes the `multipath.conf` configuration parameters relevant for priority determination and path grouping.

path_grouping_policy

Specifies the method used to combine paths into groups. Only the most important policies are listed here; see `multipath.conf(5)` for other less frequently used values.

failover

One path per path group. This setting is useful for traditional "active/passive" storage arrays.

multibus

All paths in one path group. This is useful for traditional "active/active" arrays.

group_by_prio

Path devices with the same path priority are grouped together. This option is useful for modern arrays that support asymmetric access states, like ALUA. Combined with the `alua` or `sysfs` priority algorithms, the priority groups set up by **`multipathd`** will match the primary target port groups that the storage array reports through ALUA-related SCSI commands.

Using the same policy names, the path grouping policy for a multipath map can be changed temporarily with the command:

```
tux > sudo multipath -p POLICY_NAME MAP_NAME
```

marginal_pathgroups

If set to `on` or `fpin`, "marginal" path devices are sorted into a separate path group. This is independent of the path grouping algorithm in use. See *Section 17.13.1, "Handling unreliable ("marginal") path devices"*.

detect_prio

If this is set to `yes` (default, recommended), **`multipathd`** automatically detects the best algorithm to set the priority for a storage device and ignores the `prio` setting. In practice, this means using the `sysfs` prio algorithm if ALUA support is detected.

prio

Determines the method to derive priorities for path devices. If you override this, disable `detect_prio` as follows:

```
defaults {
    detect_prio no
}
```

The following list contains only the most important methods. Several other methods are available, mainly to support legacy hardware. See `multipath.conf(5)` for the full list.

alua

Uses SCSI-3 ALUA access states to derive path priority values. The optional `exclusive_pref_bit` argument can be used to change the behavior for devices that have the ALUA "preferred primary target port group" (PREF) bit set:

```
  prio alua
  prio_args exclusive_pref_bit
```

If this option is set, "preferred" paths get a priority bonus over other active/optimized paths. Otherwise, all active/optimized paths are assigned the same priority.

sysfs

Like `alua`, but instead of sending SCSI commands to the device, it obtains the access states from `sysfs`. This causes less I/O load than `alua`, but is not suitable for all storage arrays with ALUA support.

const

Uses a constant value for all paths.

path_latency

Measures I/O latency (time from I/O submission to completion) on path devices, and assigns higher priority to devices with lower latency. See `multipath.conf(5)` for details. This algorithm is still experimental.

weightedpath

Assigns a priority to paths based on their name, serial number, Host:Bus:Target:Lun ID (HBTL), or Fibre Channel WWN. The priority value does not change over time. The method requires a `prio_args` argument, see `multipath.conf(5)` for details. For example:

```
prio weightedpath
prio_args "hbtl 2:.*:.*:.* 10 hbtl 3:.*:.*:.* 20 hbtl .* 1"
```

This assigns devices on SCSI host 3 a higher priority than devices on SCSI host 2, and all others a lower priority.

prio_args

Some `prio` algorithms require extra arguments. These are specified in this option, with syntax depending on the algorithm. See above.

hardware_handler

The name of a kernel module that the kernel uses to activate path devices when switching path groups. This option has no effect with recent kernels because hardware handlers are autodetected. See *Section 17.2.3, "Storage arrays that require specific hardware handlers"*.

path_selector

The name of a kernel module that is used for load balancing between the paths of the active path group. The available choices depend on the kernel configuration. For historical reasons, the name must always be enclosed in quotes and followed by a "0" in `multipath.conf`, like this:

```
    path_selector "queue-length 0"
```

service-time

Estimates the time pending I/O will need to complete on all paths, and selects the path with the lowest value. This is the default.

historical-service-time

Estimates future service time based on the historical service time (about which it keeps a moving average) and the number of outstanding requests. Estimates the time pending I/O will need to complete on all paths, and selects the path with the lowest value.

queue-length

Selects the path with the lowest number of currently pending I/O requests.

round-robin

Switches paths in round-robin fashion. The number of requests submitted to a path before switching to the next one can be adjusted with the options `rr_min_io_rq` and `rr_weight`.

io-affinity

This path selector currently does not work with `multipath-tools`.

After modifying the `/etc/multipath.conf` file, apply your settings as described in *Section 17.8.4, "Applying `multipath.conf` modifications"*.

17.11 Selecting devices for multipathing

On systems with multipath devices, you might want to avoid setting up multipath maps on some devices (typically local disks). `multipath-tools` offers various means for configuring which devices should be considered multipath path devices.

Note: multipath on local disks

In general, there is nothing wrong with setting up "degraded" multipath maps with just a single device on top of local disks. It works fine and requires no extra configuration. However, some administrators find this confusing or generally oppose this sort of unnecessary multipathing. Also, the multipath layer causes a slight performance overhead. See also *Section 17.3.2.2, "Root file system on a local disk"*.

After modifying the `/etc/multipath.conf` file, apply your settings as described in *Section 17.8.4, "Applying `multipath.conf` modifications"*.

17.11.1 The `blacklist` section in `multipath.conf`

The `/etc/multipath.conf` file can contain a `blacklist` section that lists all devices that should be ignored by **`multipathd`** and **`multipath`**. The following example illustrates possible ways of excluding devices:

```
blacklist {
    wwid 3600605b009e7ed501f0e45370aaeb77f ❶
    device {  ❷
        vendor ATA
```

```
        product .*
    }
    protocol scsi:sas ❸
    property SCSI_IDENT_LUN_T10 ❹
    devnode "!^dasd[a-z]*" ❺
}
```

❶ `wwid` entries are ideal for excluding specific devices, for example, the root disk.

❷ This `device` section excludes all ATA devices (the regular expression for `product` matches anything).

❸ Excluding by `protocol` allows excluding devices using certain bus types, here SAS. Other common protocol values are `scsi:fcp`, `scsi:iscsi`, and `ccw`. See `multipath.conf(5)` for more. To see the protocols that paths in your systems are using, run:

```
tux > sudo multipathd show paths format "%d %P"
```

❹ This `property` entry excludes devices that have a certain udev property (no matter what the value of the property is).

❺ Excluding devices by `devnode` is only recommended for classes of devices using regular expressions, like in the example, which excludes everything but DASD devices. Using this for individual devices like `sda` is discouraged because device node names are not persistent. The example illustrates special syntax that is only supported in the `blacklist` and `blacklist_exceptions` sections: Prefixing the regular expression with an exclamation mark (`!`) negates the match. Note that the exclamation mark must appear within double quotes.

By default, `multipath-tools` ignores all devices except SCSI, DASD, or NVMe. Technically, the built-in devnode exclude list is this negated regular expression:

```
    devnode !^(sd[a-z]|dasd[a-z]|nvme[0-9])
```

17.11.2 The `blacklist exceptions` section in `multipath.conf`

Sometimes it is desired to configure only very specific devices for multipathing. In this case, devices are excluded by default, and exceptions are defined for devices that should be part of a multipath map. The `blacklist_exceptions` section exists for this purpose. It is typically used like in the following example, which excludes everything except storage with product string “NETAPP”:

```
blacklist {
     wwid .*
```

```
}
blacklist_exceptions {
     device {
          vendor ^NETAPP$
          product .*
     }
}
```

The `blacklist_exceptions` section supports all methods described for the `blacklist` section above.

The `property` directive in `blacklist_exceptions` is mandatory because every device *must* have at least one of the "allowed" udev properties to be considered a path device for multipath (the value of the property does not matter). The built-in default for `property` is

```
    property (SCSI_IDENT_|ID_WWN)
```

Only devices that have at least one udev property matching this regular expression will be included.

17.11.3 Other options affecting device selection

Besides the `blacklist` options, there are several other settings in `/etc/multipath.conf` that affect which devices are considered multipath path devices.

find_multipaths

This option controls the behavior of **`multipath`** and **`multipathd`** when a device that is not excluded is first encountered. The possible values are:

greedy

Every device non-excluded by `blacklist` in `/etc/multipath.conf` is included. This is the default on SUSE Linux Enterprise. If this setting is active, the only way to prevent devices from being added to multipath maps is setting them as excluded.

strict

Every device is excluded, even if it is not present in the `blacklist` section of `/etc/multipath.conf`, unless its WWID is listed in the file `/etc/multipath/wwids`. It requires manual maintenance of the WWIDs file (see note below).

yes

Devices are included if they meet the conditions for `strict`, or if at least one other device with the same WWID exists in the system.

Note: Maintaining `/etc/multipath/wwids`

`multipath-tools` keeps a record of previously setup multipath maps in the file `/etc/multipath/wwids` (the "WWIDs file"). Devices with WWIDs listed in this file are considered multipath path devices. The file is important for multipath device selection for all values of `find_multipaths` except `greedy`.

If `find_multipaths` is set to `yes`, **`multipathd`** adds WWIDs to `/etc/multipath/wwids` after setting up new maps, so that these maps will be detected more quickly in the future.

The WWIDs file can be manually modified:

```
tux > sudo multipath -a 3600a098000aad1e3000064e45f2c2355 ❶
tux > sudo multipath -w /dev/sdf ❷
```

❶ This command adds the given WWID to `/etc/multipath/wwids`.

❷ This command removes the WWID of the given device.

In the `strict` mode, this is the only way to add new multipath devices. After modifying the WWIDs file, run **`multipathd reconfigure`** to apply the changes. We recommend rebuilding the initramfs after applying changes to the WWIDs file (see *Section 17.7.4, "Keeping the initramfs synchronized"*).

allow_usb_devices

If this option is set to `yes`, USB storage devices are considered for multipathing. The default is `no`.

17.12 Multipath device names and WWIDs

`multipathd` and **`multipath`** internally use WWIDs to identify devices. WWIDs are also used as map names by default. For convenience, `multipath-tools` supports assigning simpler, easier memorizable names to multipath devices.

17.12.1 WWIDs and device Identification

It is crucial for multipath operation to reliably detect devices that represent paths to the same storage volume. `multipath-tools` uses the device's World Wide Identification (WWID) for this purpose (sometimes also referred to as Universally Unique ID (UUID) or Unique ID (UID — do not confuse with "User ID"). The WWID of a map device is always the same as the WWID of its path devices.

By default, WWIDs of path devices are inferred from udev properties of the devices, which are determined in udev rules, either by reading device attributes from the sysfs file system or by using specific I/O commands. To see the udev properties of a device, run:

```
tux > udevadm info /dev/sdx
```

The udev properties used by `multipath-tools` to derive WWIDs are:

- `ID_SERIAL` for SCSI devices (do not confuse this with the device's "serial number")
- `ID_UID` for DASD devices
- `ID_WWN` for NVMe devices

Warning: Avoid changing WWIDs

It is impossible to change the WWID of a multipath map which is in use. If the WWID of mapped path devices changes because of a configuration change, the map needs to be destroyed, and a new map needs to be set up with the new WWID. This cannot be done while the old map is in use. In extreme cases, data corruption may result from WWID changes. It must therefore be *strictly avoided* to apply configuration changes that would cause map WWIDs to change.

It is allowed to enable the `uid_attrs` option in `/etc/multipath.conf`, see *Section 17.13, "Miscellaneous options"*.

17.12.2 Setting aliases for multipath maps

Arbitrary map names can be set in the `multipaths` section of `/etc/multipath.conf` as follows:

```
multipaths {
    multipath {
```

```
        wwid 3600a098000aad1e3000064e45f2c2355
        alias postgres
    }
}
```

Aliases are expressive, but they need to be assigned to each map individually, which may be cumbersome on large systems.

17.12.3 Using autogenerated user-friendly names

`multipath-tools` also supports autogenerated aliases, so-called "user-friendly names". The naming scheme of the aliases follows the pattern: mpath *`INDEX`*, where *`INDEX`* is a lower case letter (starting with `a`). So the first autogenerated alias is `mpatha`, the next one is `mpathb`, `mpathc` to `mpathz`. After that follows `mpathaa`, `mpathab`, and so on.

Map names are only useful if they are persistent. `multipath-tools` keeps track of the assigned names in the file `/etc/multipath/bindings` (the "bindings file"). When a new map is created, the WWID is first looked up in this file. If it is not found, the lowest available user-friendly name is assigned to it.

Explicit aliases as discussed in *Section 17.12.2, "Setting aliases for multipath maps"* take precedence over user-friendly names.

The following options in `/etc/multipath.conf` affect user-friendly names:

user_friendly_names

If set to `yes`, user-friendly names are assigned and used. Otherwise, the WWID is used as a map name unless an alias is configured.

alias_prefix

The prefix used to create user-friendly names, `mpath` by default.

Warning: Map names in high-availability clusters

For cluster operations, device names must be identical across all nodes in the cluster. The `multipath-tools` configuration must be kept synchronized between nodes. If `user_friendly_names` is used, **`multipathd`** may modify the `/etc/multipath/bindings` file at runtime. Such modifications must be replicated dynamically to all nodes. The same applies to `/etc/multipath/wwids` (see *Section 17.11.3, "Other options affecting device selection"*).

Note: Changing map names at runtime

It is possible to change map names at runtime. Use any of the methods described in this section and run **`multipathd reconfigure`**, and the map names will change without disrupting the system operation.

17.12.4 Referring to multipath maps

Technically, multipath maps are Device Mapper devices, which have generic names of the form `/dev/dm-n` with an integer number *n*. These names are not persistent. They should *never* be used to refer to the multipath maps. **`udev`** creates various symbolic links to these devices, which are more suitable as persistent references. These links differ with respect to their invariance against certain configuration changes. The following typical example shows various symlinks all pointing to the same device.

```
/dev/disk/by-id/dm-name-mpathb❶ -> ../../dm-1
/dev/disk/by-id/dm-uuid-mpath-3600a098000aad73f00000a3f5a275dc8❷ -> ../../dm-1
/dev/disk/by-id/scsi-3600a098000aad73f00000a3f5a275dc8❸ -> ../../dm-1
/dev/disk/by-id/wwn-0x600a098000aad73f00000a3f5a275dc8❹ -> ../../dm-1
/dev/mapper/mpathb❺ -> ../dm-1
```

❶ ❺ These two links use the map name to refer to the map. Thus, the links will change if the map name changes, for example, if you enable or disable user-friendly names.

❷ This link uses the device mapper UUID, which is the WWID used by `multipath-tools` prefixed by the string `dm-uuid-mpath-`. It is independent of the map name.
The device mapper UUID is the preferred form to ensure that *only multipath devices* are referenced. For example, the following line in `/etc/lvm/lvm.conf` rejects all devices except multipath maps:

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

❸ ❹ These are links that would normally point to path devices. The multipath device took them over, because it has a higher udev link priority (see `udev(7)`). When the map is destroyed or multipathing is turned off, they will still exist and point to one of the path devices instead. This provides a means to reference a device by its WWID, whether or not multipathing is active.

For **partitions** on multipath maps created by the `kpartx` tool, there are similar symbolic links, derived from the parent device name or WWID and the partition number:

```
/dev/disk/by-id/dm-name-mpatha-part2 -> ../../dm-5
/dev/disk/by-id/dm-uuid-part2-mpath-3600a098000aad1e300000b4b5a275d45 -> ../../dm-5
/dev/disk/by-id/scsi-3600a098000aad1e300000b4b5a275d45-part2 -> ../../dm-5
/dev/disk/by-id/wwn-0x600a098000aad1e300000b4b5a275d45-part2 -> ../../dm-5
/dev/disk/by-partuuid/1c2f70e0-fb91-49f5-8260-38eacaf7992b -> ../../dm-5
/dev/disk/by-uuid/f67c49e9-3cf2-4bb7-8991-63568cb840a4 -> ../../dm-5
/dev/mapper/mpatha-part2 -> ../dm-5
```

Note that partitions often have `by-uuid` links, too, referring not to the device itself but to the file system it contains. These links are often preferable. They are invariant even if the file system is copied to a different device or partition.

Warning: Map names in the initramfs

When `dracut` builds an initramfs, it creates hard-coded references to devices in the initramfs, using `/dev/mapper/$MAP_NAME` references by default. These hard-coded references will not be found during boot if the map names used in the initramfs do not match the names used during building the initramfs, causing boot failure. Normally this will not happen, because `dracut` will add all multipath configuration files to the initramfs. But problems can occur if the initramfs is built from a different environment, for example, in the rescue system or during an offline update. To prevent this boot failure, change `dracut`'s `persistent_policy` setting, as explained in *Section 17.7.4.2, "Persistent device names in the initramfs"*.

17.13 Miscellaneous options

This section lists some useful `multipath.conf` options that were not mentioned so far. See `multipath.conf(5)` for a full list.

verbosity

Controls the log verbosity of both `multipath` and `multipathd`. The command-line option `-v` overrides this setting for both commands. The value can be between 0 (only fatal errors) and 4 (verbose logging). The default is 2.

uid_attrs

This option enables an optimization for processing udev events, so-called "uevent merging". It is useful in environments in which hundreds of path devices may fail or reappear simultaneously. In order to make sure that path WWIDs do not change (see *Section 17.12.1, "WWIDs and device Identification"*), the value should be set exactly like this:

```
defaults {
     uid_attrs "sd:ID_SERIAL dasd:ID_UID nvme:ID_WWN"
}
```

skip_kpartx

If set to `yes` for a multipath device (default is `no`), do not create partition devices on top of the given device (see *Section 17.7.3, "Partitions on multipath devices and* **`kpartx`***"*). Useful for multipath devices used by virtual machines. Previous SUSE Linux Enterprise Server releases achieved the same effect with the parameter "`features 1 no_partitions`".

max_sectors_kb

Limits the maximum amount of data sent in a single I/O request for all path devices of the multipath map.

ghost_delay

On active/passive arrays, it can happen that passive paths (in "ghost" state) are probed before active paths. If the map was activated immediately and I/O was sent, this would cause a possibly costly path activation. This parameter specifies the time (in seconds) to wait for active paths of the map to appear before activating the map. The default is `no` (no ghost delay).

recheck_wwid

If set to `yes` (default is `no`), double-checks the WWID of restored paths after failure, and removes them if the WWID has changed. This is a safety measure against data corruption.

enable_foreign

`multipath-tools` provides a plugin API for other multipathing backends than Device Mapper multipath. The API supports monitoring and displaying information about the multipath topology using standard commands like **`multipath -ll`**. Modifying the topology is unsupported.

The value of `enable_foreign` is a regular expression to match against foreign library names. The default value is "`NONE`".

SUSE Linux Enterprise Server ships the `nvme` plugin, which adds support for the native NVMe multipathing (see *Section 17.2.1, "Multipath implementations: device mapper and NVMe"*). To enable the `nvme` plugin, set

```
defaults {
    enable_foreign nvme
}
```

17.13.1 Handling unreliable ("marginal") path devices

Unstable conditions in the fabric can cause path devices to behave erratically. They exhibit frequent I/O errors, recover, and fail again. Such path devices are also denoted "marginal" or "shaky" paths. This section summarizes the options that `multipath-tools` provides to deal with this problem.

Note: multipathd's marginal path checking algorithm

If a path device exhibits a second failure (good → bad transition) before `marginal_path_double_failed_time` elapses after the first failure, **`multipathd`** starts monitoring the path at a rate of 10 requests per second, for a monitoring period of `marginal_path_err_sample_time`. If the error rate during the monitoring period exceeds `marginal_path_err_rate_threshold`, the path is classified as marginal. After `marginal_path_err_recheck_gap_time`, the path transitions to normal state again.

This algorithm is used if all four numeric `marginal_path_` parameters are set to a positive value, and `marginal_pathgroups` is not set to `fpin`.

marginal_path_double_failed_time

Maximum time (in seconds) between two path failures that triggers path monitoring.

marginal_path_err_sample_time

Length (in seconds) of the path monitoring interval.

marginal_path_err_rate_threshold

Minimum error rate (per thousand I/Os).

marginal_path_err_recheck_gap_time

Time (in seconds) to keep the path in marginal state.

A simpler algorithm using the parameters `san_path_err_threshold`, `san_path_err_forget_rate`, and `san_path_err_recovery time` is also available. See the "Shaky paths detection" section in `multipath.conf(5)`.

17.14 Best practice

17.14.1 Best practices for configuration

The large number of configuration directives is daunting at first. Usually, you can get good results with an empty configuration, unless you are in a clustering environment.

Here are some general recommendations for stand-alone servers. They are *not mandatory*. See the documentation of the respective parameters in the previous sections for background information.

```
defaults {
    deferred_remove      yes
    find_multipaths      yes
    enable_foreign       nvme

}
devices {
    # A catch-all device entry.
    device {
        vendor               .*
        product              .*
        dev_loss_tmo         infinity
        no_path_retry        60            # 5 minutes
        path_grouping_policy group_by_prio
        path_selector        "historical-service-time 0"
        reservation_key      file          # if using SCSI persistent reservations
    }
    # Follow up with specific device entries below, they will take precedence.
}
```

After modifying the `/etc/multipath.conf` file, apply your settings as described in *Section 17.8.4, "Applying `multipath.conf` modifications"*.

17.14.2 Interpreting multipath I/O status

For a quick overview of the multipath subsystem, use **`multipath -ll`** or **`multipathd show topology`**. The output of these commands has the same format. The former command reads the kernel state, while the latter prints the status of the multipath daemon. Normally both states are equal. Here is an example of the output:

```
tux > sudo multipathd show topology
mpatha(1) (3600a098000aad1e300000b4b5a275d45(2)) dm-0(3) NETAPP,INF-01-00(4)
size=64G features='3 queue_if_no_path pg_init_retries 50'(5) hwhandler='1 alua'(6) wp=rw(7)
|-+- (8)policy='historical-service-time 2'(9) prio=50(10) status=active(11)
| |-(12) 3:0:0:1(13) sdb 8:16(14) active(15) ready(16) running(17)
| `- 4:0:0:1 sdf 8:80  active ready running
`-+- policy='historical-service-time 2' prio=10 status=enabled
  `- 4:0:1:1 sdj 8:144 active ready running
```

1. The map name.
2. The map WWID (if different from the map name).
3. The device node name of the map device.
4. The vendor and product name.
8. A path group. The indented lines below the path group list the path devices that belong to it.
9. The path selector algorithm used by the path group. The "2" can be ignored.
10. The priority of the path group.
11. The status of the path group (`active`, `enabled`, or `disabled`). The active path group is the one that I/O is currently sent to.
12. A path device.
13. The bus ID of the device (here, a SCSI Host:Bus:Target:Lun ID).
14. The device node name and major/minor number of the path device.
15. The kernel device mapper state of the path (`active` or `failed`).
16. The multipath path device state (see below).
17. The state of the path device in the kernel. This is a device-type specific value. For SCSI, it is either `running` or `offline`.

The multipath path device states are:

`ready`	The path is healthy and up

`ghost`	A passive path in an active/passive array
`faulty`	The path is down or unreachable
`i/o timeout`	A checker command timed out
`i/o pending`	Waiting for the completion of a path checker command
`delayed`	Path reinstantiation is delayed to avoid "flapping"
`shaky`	An unreliable path (emc path checker only)

17.14.3 Using LVM2 on multipath devices

LVM2 has built-in support for detecting multipath devices. It is activated by default in `/etc/lvm/lvm.conf`:

```
    multipath_component_detection=1
```

This works reliably only if LVM2 is also configured to obtain information about device properties from udev:

```
    external_device_info_source="udev"
```

It is also possible (although normally not necessary) to create a filter expression for LVM2 to ignore all devices except multipath devices. See *Section 17.12.4, "Referring to multipath maps"*.

17.14.4 Resolving stalled I/O

If all paths fail concurrently and I/O is queued, applications may stall for a long time. To resolve this, you can use the following procedure:

1. Enter the following command at a terminal prompt:

   ```
   tux > sudo multipathd disablequeueing map MAPNAME
   ```

   Replace *`MAPNAME`* with the correct WWID or mapped alias name for the device.

This command immediately causes all queued I/O to fail and propagates the error to the calling application. File systems will observe I/O errors and switch to read-only mode.

2. Reactivate queuing by entering the following command:

```
tux > sudo multipathd restorequeueing MAPNAME
```

17.14.5 MD RAID on multipath devices

MD RAID arrays on top of multipathing are set up automatically by the system's udev rules. No special configuration in `/etc/mdadm.conf` is necessary.

17.14.6 Scanning for new devices without rebooting

If your system has already been configured for multipathing and you need to add storage to the SAN, you can use the **`rescan-scsi-bus.sh`** script to scan for the new devices. The general syntax for the command follows:

```
tux > sudo rescan-scsi-bus.sh [-a] [-r] --hosts=2-3,5
```

Where the options have the following meaning:

-a
: the option ensures that all SCSI targets are scanned, otherwise only already existing targets will be scanned for new LUNs.

-r
: the option enables the removal of devices which have been removed on the storage side.

--hosts
: the option specifies the list of host bus adapters to scan (the default is to scan all).

Run **`rescan-scsi-bus.sh --help`** for help on additional options.

If **`multipathd`** is running and new SAN devices are discovered, they should be automatically set up as multipath maps according to the configuration described in *Section 17.11, "Selecting devices for multipathing"*.

Warning: Dell/EMC PowerPath environments

In EMC PowerPath environments, do not use the `rescan-scsi-bus.sh` utility provided with the operating system or the HBA vendor scripts for scanning the SCSI buses. To avoid potential file system corruption, EMC requires that you follow the procedure provided in the vendor documentation for EMC PowerPath for Linux.

17.15 Troubleshooting MPIO

If a system runs into emergency mode on a system with multipath, printing messages about missing devices, the reason is almost always one of these:

- Inconsistent configuration of multipath device selection
- Use of non-existing device references

17.15.1 Understanding device selection issues

A block device can only either be part of a multipath map or be used directly (mounted as file system, used as swap, LVM physical volume, or otherwise). If a device is already mounted, an attempt by multipathd to make it part of a multipath map will fail with a “Device or resource busy” error. Vice-versa, the same error results if **`systemd`** attempts to mount a device which has already been made part of a multipath map.

Storage device activation during boot is handled by a complex interaction between **`systemd`**, `udev`, **`multipathd`** and some other tools. `udev` rules play a central role. They set device properties that indicate to other subsystems how a device should be used. The multipath-related udev rules set the following properties for devices that are selected for multipathing:

```
SYSTEMD_READY=0
DM_MULTIPATH_DEVICE_PATH=1
```

Partition devices inherit these properties from their parents.

If these properties are not set correctly, if some tool does not respect them, or if they get set too late, a race condition between **`multipathd`** and some other subsystem may result. Only one of the contenders can win the race; the other one will see a “Device or resource busy” error.

One problem in this context is that the tools of the LVM2 suite do not evaluate udev properties by default. They rely on their own logic for determining whether a device is a multipath component, which sometimes does not match the logic of the rest of the system. A workaround for this is described in *Section 17.14.3, "Using LVM2 on multipath devices"*.

Note: Example of boot deadlock

Consider a system with multipathing where the root device is not multipathed, and no devices are excluded from multipath (see *Multipath disabled in the initramfs* in *Section 17.3.2.2, "Root file system on a local disk"*). The root file system is mounted in the initramfs. **`systemd`** switches to the root file system and **`multipathd`** starts up. Because the device is already mounted, **`multipathd`** fails to set up the multipath map for it. Because the root device is not configured in `blacklist`, it is considered a multipath device, and `SYSTEMD_READY=0` is set for it.

Later in the boot process, the system attempts to mount additional file systems like `/var` and `/home`. Usually, these file systems will be on the same device as the root file system, by default as BTRFS subvolumes of the root file system itself. But systemd cannot mount them because of `SYSTEMD_READY=0`. *We are in a deadlock*: The dm-multipath device cannot be created, and the underlying device is blocked for systemd. The additional file systems cannot be mounted, resulting in boot failure.

A solution to this problem already exists. **`multipathd`** detects this situation and releases the device to **`systemd`** which can then proceed mounting the file system. However, it is important to understand the general problem, which can still occur in more subtle ways.

17.15.2 Understanding device referencing issues

An example of a device referencing issue has been given in *Section 17.7.4.2, "Persistent device names in the initramfs"*. Typically, there are multiple symbolic links pointing to a device node (see *Section 17.12.4, "Referring to multipath maps"*). But these links do not always exist; **`udev`** creates them according to the current udev rules. For example, if multipathing is off, symbolic links under `/dev/mapper/` for multipath devices will be missing. Thus, any reference to a `/dev/mapper/` device will fail.

Such references can appear in various places, notably in `/etc/fstab` and `/etc/crypttab`, in the initramfs, or even on the kernel command line.

The safest way to circumvent this problem is to avoid using the kind of device references that are not persistent between boots or depend on system configuration. We generally recommend referring to file systems (and similar entities like swap space) by properties of the file system itself (like UUID or label) rather than the containing device. If such references are not available and device references are required, for example, in `/etc/crypttab`, the options should be evaluated carefully. For example, in *Section 17.12.4, "Referring to multipath maps"*, the best option might be the `/dev/disk/by-id/wwn-` link because it would also work with `multipath=off`.

17.15.3 Troubleshooting steps in emergency mode

As there are many error situations that differ in subtle ways, it is impossible to provide a step-by-step recovery guide. But with the background knowledge from the previous subsections, you should be able to figure out the problem if a system runs into emergency mode because of multipathing issues. Before you begin debugging, make sure you have checked the following questions:

- Is the multipath service enabled?
- Is the multipath dracut module included in the initramfs?
- Is my root device configured as a multipath device? If not, is the root device properly excluded from multipath as described in *Section 17.11.1, "The `blacklist` section in `multipath.conf`"*, or are you relying on the absence of the multipath module in the initramfs (see *Section 17.3.2.2, "Root file system on a local disk"*)?
- Does the system enter emergency mode before or after switching to the real root file system?

If you are unsure with respect to the last question, here is a sample dracut emergency prompt as it would be printed before switching root:

```
Generating "/run/initramfs/rdsosreport.txt"
Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.

You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or /boot
after mounting them and attach it to a bug report.

Give root password for maintenance
(or press Control-D to continue):
```

The mention of `rdsosreport.txt` is a clear indication that the system is still running from the initramfs. If you are still uncertain, log in and check for the existence of the file `/etc/initrd-release`. This file exists only in an initramfs environment.

If emergency mode is entered after switching root, the emergency prompt looks similar, but `rdsosreport.txt` is not mentioned:

```
Timed out waiting for device dev-disk-by\x2duuid-c4a...cfef77d.device.
[DEPEND] Dependency failed for Local File Systems.
[DEPEND] Dependency failed for Postfix Mail Transport Agent.
Welcome to emergency shell
Give root password for maintenance
(or press Control-D to continue):
```

PROCEDURE 17.2: STEPS FOR ANALYZING THE SITUATION IN EMERGENCY MODE

1. Try to figure out what failed by examining failed systemd units and the journal.

   ```
   root # systemctl --failed
   root # journalctl -b -o short-monotonic
   ```

   When looking at the journal, determine the *first* failed unit. When you have found the first failure, examine the messages before and around that point in time very carefully. Are there any warnings or other suspicious messages?
   Watch out for the root switch ("`Switching root.`") and for messages about SCSI devices, device mapper, multipath, and LVM2. Look for **`systemd`** messages about devices and file systems ("`Found device` …", "`Mounting` …", "`Mounted` …").

2. Examine the existing devices, both low-level devices and device mapper devices (note that some of the commands below may not be available in the initramfs):

   ```
   root # cat /proc/partitions
   root # ls -l /sys/class/block
   root # ls -l /dev/disk/by-id/* /dev/mapper/*
   root # dmsetup ls --tree
   root # lsblk
   root # lsscsi
   ```

   From the output of the commands above, you should get an idea whether the low-level devices were successfully probed, and whether any multipath maps and multipath partitions were set up.

3. If the device mapper multipath setup is not as you expect, examine the udev properties, in particular, `SYSTEMD_READY` (see above)

```
root # udevadm info -e
```

4. If the previous step showed unexpected udev properties, something may have gone wrong during udev rule processing. Check other properties, in particular, those used for device identification (see *Section 17.12.1, "WWIDs and device Identification"*). If the udev properties are correct, check the journal for **`multipathd`** messages again. Look for "`Device or resource busy`" messages.

5. If the system failed to mount or otherwise activate a device, it is often helpful to try activating this device manually:

```
root # mount /var
root # swapon -a
root # vgchange -a y
```

   Mostly, the manual activation will succeed and allow to proceed with system boot (usually by simply logging out from the emergency shell) and examine the situation further in the booted system.
   If manual activation fails, you will probably see error messages that provide clues about what is going wrong. You can also try the commands again with increased verbosity.

6. At this point, you should have some idea what went wrong (if not, contact SUSE support and be prepared to answer most of the questions raised above).
   You should be able to correct the situation with a few shell commands, exit the emergency shell, and boot successfully. You will still need to adjust your configuration to make sure the same problem will not occur again in the future.
   Otherwise, you will need to boot the rescue system, set up the devices manually to **`chroot`** into the real root file system, and attempt to fix the problem based on the insight you got in the previous steps. Be aware that in this situation, the storage stack for the root file system may differ from normal. Depending on your setup, you may have force addition or omission of dracut modules when building a new initramfs. See also *Section 17.7.4.1, "Enabling or disabling multipathing in the initramfs"*.

7. If the problem occurs frequently or even on every boot attempt, try booting with increased verbosity in order to get more information about the failure. The following kernel parameters, or a combination of them, are often helpful:

```
udev.log-priority=debug ❶
systemd.log_level=debug ❷
scsi_mod.scsi_logging_level=020400 ❸
rd.debug ❹
```

❶ Increase the log level of **`systemd-udevd`** and udev rule processing.

❷ Increase the log level of **`systemd`**.

❸ Increase the logging level of the kernel's SCSI subsystem.

❹ Trace the scripts in the initramfs.

In addition, it may make sense to enable logging for certain drivers and configure a serial console to capture the output during boot.

17.15.4 Technical information documents

For more information about troubleshooting multipath I/O issues on SUSE Linux Enterprise Server, see the following Technical Information Documents (TIDs) in the SUSE Knowledgebase:

- *Using LVM on local and SAN attached devices* (https://www.suse.com/support/kb/doc/?id=000016331)
- *Using LVM on Multipath (DM MPIO) Devices* (https://www.suse.com/support/kb/doc/?id=000017521)
- *HOWTO: Add, Resize and Remove LUN without restarting SLES* (https://www.suse.com/support/kb/doc/?id=000017762)

18 Managing Access Control Lists over NFSv4

There is no single standard for Access Control Lists (ACLs) in Linux beyond the simple read, write, and execute (`rwx`) flags for user, group, and others (`ugo`). One option for finer control is the *Draft POSIX ACLs*, which were never formally standardized by POSIX. Another is the NFSv4 ACLs, which were designed to be part of the NFSv4 network file system with the goal of making something that provided reasonable compatibility between POSIX systems on Linux and WIN32 systems on Microsoft Windows.

NFSv4 ACLs are not sufficient to correctly implement Draft POSIX ACLs so no attempt has been made to map ACL accesses on an NFSv4 client (such as using **`setfacl`**).

When using NFSv4, Draft POSIX ACLs cannot be used even in emulation and NFSv4 ACLs need to be used directly; that means while **`setfacl`** can work on NFSv3, it cannot work on NFSv4. To allow NFSv4 ACLs to be used on an NFSv4 file system, SUSE Linux Enterprise Server provides the `nfs4-acl-tools` package, which contains the following:

- **`nfs4-getfacl`**
- **`nfs4-setfacl`**
- **`nfs4-editacl`**

These operate in a generally similar way to **`getfacl`** and **`setfacl`** for examining and modifying NFSv4 ACLs. These commands are effective only if the file system on the NFS server provides full support for NFSv4 ACLs. Any limitation imposed by the server will affect programs running on the client in that some particular combinations of Access Control Entries (ACEs) might not be possible.

It is not supported to mount NFS volumes locally on the exporting NFS server.

Additional Information

For information, see *Introduction to NFSv4 ACLs* at http://wiki.linux-nfs.org/wiki/index.php/ACLs#Introduction_to_NFSv4_ACLs.

A GNU licenses

This appendix contains the GNU Free Documentation License version 1.2.

GNU Free Documentation License

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.