Proxmox Backup Documentation

Release 0.8.14-1

Proxmox Support Team

Sep 02, 2020

TABLE OF CONTENTS

CHAPTER

ONE

INTRODUCTION

1.1 What is Proxmox Backup Server

Proxmox Backup Server is an enterprise-class, client-server backup software package that backs up *virtual machine*s, *container*s, and physical hosts. It is specially optimized for the Proxmox Virtual Environment platform and allows you to back up your data securely, even between remote sites, providing easy management with a web-based user interface.

Proxmox Backup Server supports deduplication, compression, and authenticated encryption (AE). Using *Rust* as the implementation language guarantees high performance, low resource usage, and a safe, high-quality codebase.

It features strong client-side encryption. Thus, it's possible to backup data to targets that are not fully trusted.

1.2 Architecture

Proxmox Backup Server uses a client-server model. The server stores the backup data and provides an API to create and manage data stores. With the API, it's also possible to manage disks and other server-side resources.

The backup client uses this API to access the backed up data. With the command line tool `proxmox-backup-client` you can create backups and restore data. For QEMU with Proxmox Virtual Environment we deliver an integrated client.

A single backup is allowed to contain several archives. For example, when you backup a *virtual machine*, each disk is stored as a separate archive inside that backup. The VM configuration itself is stored as an extra file. This way, it's easy to access and restore only important parts of the backup, without the need to scan the whole backup.

1.3 Main Features

Support for Proxmox VE The Proxmox Virtual Environment is fully supported and you can easily backup *virtual machine*s and *container*s.

Performance The whole software stack is written in *Rust*, in order to provide high speed and memory efficiency.

Deduplication Periodic backups produce large amounts of duplicate data. The deduplication layer avoids redundancy and minimizes the storage space used.

Incremental backups Changes between backups are typically low. Reading and sending only the delta reduces the storage and network impact of backups.

Data Integrity The built-in SHA-256 checksum algorithm ensures accuracy and consistency in your backups.

Remote Sync It is possible to efficiently synchronize data to remote sites. Only deltas containing new data are transferred.

Compression The ultra-fast Zstandard compression is able to compress several gigabytes of data per second.

Encryption Backups can be encrypted on the client-side, using AES-256 in Galois/Counter Mode (GCM) mode. This authenticated encryption (AE) mode provides very high performance on modern hardware.

Web interface Manage the Proxmox Backup Server with the integrated, web-based user interface.

Open Source No secrets. Proxmox Backup Server is free and open-source software. The source code is licensed under AGPL, v3.

Support Enterprise support will be available from Proxmox once the beta phase is over.

1.4 Reasons for Data Backup?

The main purpose of a backup is to protect against data loss. Data loss can be caused by both faulty hardware and human error.

A common mistake is to accidentally delete a file or folder which is still required. Virtualization can even amplify this problem, as deleting a whole virtual machine can be as easy as pressing a single button.

For administrators, backups can serve as a useful toolkit for temporarily storing data. For example, it is common practice to perform full backups before installing major software updates. If something goes wrong, you can easily restore the previous state.

Another reason for backups are legal requirements. Some data, especially business records, must be kept in a safe place for several years by law, so that they can be accessed if required.

In general, data loss is very costly as it can severely damage your business. Therefore, ensure that you perform regular backups and run restore tests.

1.5 Software Stack

Proxmox Backup Server consists of multiple components:

- A server-daemon providing, among other things, a RESTfull API, super-fast asynchronous tasks, lightweight usage statistic collection, scheduling events, strict separation of privileged and unprivileged execution environments
- A JavaScript management web interface
- A management CLI tool for the server (*proxmox-backup-manager*)
- A client CLI tool (*proxmox-backup-client*) to access the server easily from any *Linux amd64* environment

Aside from the web interface, everything is written in the Rust programming language.

> "The Rust programming language helps you write faster, more reliable software. High-level ergonomics and low-level control are often at odds in programming language design; Rust challenges that conflict. Through balancing powerful technical capacity and a great developer experience, Rust gives you the option to control low-level details (such as memory usage) without all the hassle traditionally associated with such control."

—The Rust Programming Language

Todo: further explain the software stack

1.6 Getting Help

1.6.1 Community Support Forum

We always encourage our users to discuss and share their knowledge using the Proxmox Community Forum. The forum is moderated by the Proxmox support team. The large user base is spread out all over the world. Needless to say that such a large forum is a great place to get information.

1.6.2 Mailing Lists

Proxmox Backup Server is fully open-source and contributions are welcome! Here is the primary communication channel for developers:

Mailing list for developers PBS Development List

1.6.3 Bug Tracker

Proxmox runs a public bug tracker at https://bugzilla.proxmox.com. If an issue appears, file your report there. An issue can be a bug as well as a request for a new feature or enhancement. The bug tracker helps to keep track of the issue and will send a notification once it has been solved.

1.7 License

1.8 History

Todo: Add development History of the product

CHAPTER

TWO

INSTALLATION

Proxmox Backup is split into a server and client part. The server part can either be installed with a graphical installer or on top of Debian from the provided package repository.

2.1 Debian Package Repositories

All Debian based systems use APT as a package management tool. The lists of repositories are defined in `/etc/apt/sources.list` and the `.list` files found in the `/etc/apt/sources.d/` directory. Updates can be installed directly with the `apt` command line tool, or via the GUI.

APT `sources.list` files list one package repository per line, with the most preferred source listed first. Empty lines are ignored and a `#` character anywhere on a line marks the remainder of that line as a comment. The information available from the configured sources is acquired by `apt update`.

Listing 1: File: `/etc/apt/sources.list`

```
deb http://ftp.debian.org/debian buster main contrib
deb http://ftp.debian.org/debian buster-updates main contrib

# security updates
deb http://security.debian.org/debian-security buster/updates main contrib
```

In addition, you need a package repository from Proxmox to get Proxmox Backup updates.

During the Proxmox Backup beta phase, only one repository (pbstest) will be available. Once released, an Enterprise repository for production use and a no-subscription repository will be provided.

2.1.1 SecureApt

The *Release* files in the repositories are signed with GnuPG. APT is using these signatures to verify that all packages are from a trusted source.

If you install Proxmox Backup Server from an official ISO image, the verification key is already installed.

If you install Proxmox Backup Server on top of Debian, download and install the key with the following commands:

```
# wget http://download.proxmox.com/debian/proxmox-ve-release-6.x.gpg -O /etc/apt/trusted.gpg.
↪d/proxmox-ve-release-6.x.gpg
```

Verify the SHA512 checksum afterwards with:

```
# sha512sum /etc/apt/trusted.gpg.d/proxmox-ve-release-6.x.gpg
```

The output should be:

```
acca6f416917e8e11490a08a1e2842d500b3a5d9f322c6319db0927b2901c3eae23cfb5cd5df6facf2b57399d3cfa52ad7769ebdd75d9
↪/etc/apt/trusted.gpg.d/proxmox-ve-release-6.x.gpg
```

and the md5sum:

```
# md5sum /etc/apt/trusted.gpg.d/proxmox-ve-release-6.x.gpg
```

Here, the output should be:

```
f3f6c5a3a67baf38ad178e5ff1ee270c /etc/apt/trusted.gpg.d/proxmox-ve-release-6.x.gpg
```

2.1.2 Proxmox Backup Beta Repository

During the public beta, there is a repository called `pbstest`. This one contains the latest packages and is heavily used by developers to test new features.

You can access this repository by adding the following line to `/etc/apt/sources.list`:

Listing 2: sources.list entry for `pbstest`

```
deb http://download.proxmox.com/debian/pbs buster pbstest
```

If you installed Proxmox Backup Server from the official beta ISO, you should have this repository already configured in `/etc/apt/sources.list.d/pbstest-beta.list`

2.2 Server installation

The backup server stores the actual backed up data and provides a web based GUI for various management tasks such as disk management.

Note: You always need a backup server. It is not possible to use Proxmox Backup without the server part.

The disk image (ISO file) provided by Proxmox includes a complete Debian system ("buster" for version 1.x) as well as all necessary packages for the Proxmox Backup server.

The installer will guide you through the setup process and allow you to partition the local disk(s), apply basic system configurations (e.g. timezone, language, network), and install all required packages. The provided ISO will get you started in just a few minutes, and is the recommended method for new and existing users.

Alternatively, Proxmox Backup server can be installed on top of an existing Debian system.

2.2.1 Install Proxmox Backup with the Installer

Download the ISO from https://www.proxmox.com/downloads. It includes the following:

- The Proxmox Backup server installer, which partitions the local disk(s) with ext4, ext3, xfs or ZFS, and installs the operating system
- Complete operating system (Debian Linux, 64-bit)
- Our Linux kernel with ZFS support
- Complete tool-set to administer backups and all necessary resources
- Web based GUI management interface

Note: During the installation process, the complete server is used by default and all existing data is removed.

2.2.2 Install Proxmox Backup server on Debian

Proxmox ships as a set of Debian packages which can be installed on top of a standard Debian installation. After configuring the sysadmin_package_repositories, you need to run:

```
# apt-get update
# apt-get install proxmox-backup-server
```

The commands above keep the current (Debian) kernel and install a minimal set of required packages.

If you want to install the same set of packages as the installer does, please use the following:

```
# apt-get update
# apt-get install proxmox-backup
```

This will install all required packages, the Proxmox kernel with ZFS support, and a set of common and useful packages.

Caution: Installing Proxmox Backup on top of an existing Debian installation looks easy, but it assumes that the base system and local storage have been set up correctly. In general this is not trivial, especially when LVM or ZFS is used. The network configuration is completely up to you as well.

Note:

You can access the webinterface of the Proxmox Backup Server with your web browser, using HTTPS on port 8007. For example at

`https://<ip-or-dns-name>:8007`

2.2.3 Install Proxmox Backup server on Proxmox VE

After configuring the sysadmin_package_repositories, you need to run:

```
# apt-get update
# apt-get install proxmox-backup-server
```

Caution: Installing the backup server directly on the hypervisor is not recommended. It is safer to use a separate physical server to store backups. Should the hypervisor server fail, you can still access the backups.

Note: You can access the webinterface of the Proxmox Backup Server with your web browser, using HTTPS on port 8007. For example at `https://<ip-or-dns-name>:8007`

2.3 Client installation

2.3.1 Install Proxmox Backup client on Debian

Proxmox ships as a set of Debian packages to be installed on top of a standard Debian installation. After configuring the sysadmin_package_repositories, you need to run:

```
# apt-get update
# apt-get install proxmox-backup-client
```

2.3.2 Installing from source

Todo: Add section “Installing from source”

2.3.3 Installing statically linked binary

Todo: Add section “Installing statically linked binary”

CHAPTER

THREE

BACKUP MANAGEMENT

3.1 Terminology

3.1.1 Backup Content

When doing deduplication, there are different strategies to get optimal results in terms of performance and/or deduplication rates. Depending on the type of data, it can be split into *fixed* or *variable* sized chunks.

Fixed sized chunking requires minimal CPU power, and is used to backup virtual machine images.

Variable sized chunking needs more CPU power, but is essential to get good deduplication rates for file archives.

The Proxmox Backup Server supports both strategies.

File Archives: `<name>.pxar`

A file archive stores a full directory tree. Content is stored using the *Proxmox File Archive Format (.pxar)*, split into variable-sized chunks. The format is optimized to achieve good deduplication rates.

Image Archives: `<name>.img`

This is used for virtual machine images and other large binary data. Content is split into fixed-sized chunks.

Binary Data (BLOBs)

This type is used to store smaller (< 16MB) binary data such as configuration files. Larger files should be stored as image archive.

> **Caution:** Please do not store all files as BLOBs. Instead, use the file archive to store whole directory trees.

Catalog File: `catalog.pcat1`

The catalog file is an index for file archives. It contains the list of files and is used to speed up search operations.

The Manifest: `index.json`

The manifest contains the list of all backup files, their sizes and checksums. It is used to verify the consistency of a backup.

3.1.2 Backup Type

The backup server groups backups by *type*, where *type* is one of:

`vm` This type is used for *virtual machine*s. Typically consists of the virtual machine's configuration file and an image archive for each disk.

`ct` This type is used for *container*s. Consists of the container's configuration and a single file archive for the filesystem content.

`host` This type is used for backups created from within the backed up machine. Typically this would be a physical host but could also be a virtual machine or container. Such backups may contain file and image archives, there are no restrictions in this regard.

3.1.3 Backup ID

A unique ID. Usually the virtual machine or container ID. `host` type backups normally use the hostname.

3.1.4 Backup Time

The time when the backup was made.

3.1.5 Backup Group

The tuple `<type>/<ID>` is called a backup group. Such a group may contain one or more backup snapshots.

3.1.6 Backup Snapshot

The triplet `<type>/<ID>/<time>` is called a backup snapshot. It uniquely identifies a specific backup within a datastore.

Listing 1: Backup Snapshot Examples

```
vm/104/2019-10-09T08:01:06Z
host/elsa/2019-11-08T09:48:14Z
```

As you can see, the time format is RFC3399 with Coordinated Universal Time (UTC, identified by the trailing *Z*).

3.2 Backup Server Management

The command line tool to configure and manage the backup server is called **`proxmox-backup-manager`**.

3.2.1 DataStore

A datastore is a place where backups are stored. The current implementation uses a directory inside a standard unix file system (`ext4`, `xfs` or `zfs`) to store the backup data.

Datastores are identified by a simple *ID*. You can configure it when setting up the backup server.

Note: The *File Layout* requires the file system to support at least *65538* subdirectories per directory. That number comes from the 2^{16} pre-created chunk namespace directories, and the `.` and `..` default directory entries. This requirement excludes certain filesystems and filesystem configuration from being supported for a datastore. For example, `ext3` as a whole or `ext4` with the `dir_nlink` feature manually disabled.

3.2.2 Disk Management

Proxmox Backup Server comes with a set of disk utilities, which are accessed using the `disk` subcommand. This subcommand allows you to initialize disks, create various filesystems, and get information about the disks.

To view the disks connected to the system, use the `list` subcommand of `disk`:

```
# proxmox-backup-manager disk list
```

name	used	gpt	disk-type	size	model	wearout	status
sda	lvm	1	hdd	34359738368	QEMU_HARDDISK	-	passed
sdb	unused	1	hdd	68719476736	QEMU_HARDDISK	-	passed
sdc	unused	1	hdd	68719476736	QEMU_HARDDISK	-	passed

To initialize a disk with a new GPT, use the `initialize` subcommand:

```
# proxmox-backup-manager disk initialize sdX
```

You can create an `ext4` or `xfs` filesystem on a disk, using `fs create`. The following command creates an `ext4` filesystem and passes the `--add-datastore` parameter, in order to automatically create a datastore on the disk (in this case `sdd`). This will create a datastore at the location `/mnt/datastore/store1`:

```
# proxmox-backup-manager disk fs create store1 --disk sdd --filesystem ext4 --add-datastore␣
↪true
create datastore 'store1' on disk sdd
Percentage done: 1
...
Percentage done: 99
TASK OK
```

You can also create a `zpool` with various raid levels. The command below creates a mirrored `zpool` using two disks (`sdb` & `sdc`) and mounts it on the root directory (default):

```
# proxmox-backup-manager disk zpool create zpool1 --devices sdb,sdc --raidlevel mirror
create Mirror zpool 'zpool1' on devices 'sdb,sdc'
# "zpool" "create" "-o" "ashift=12" "zpool1" "mirror" "sdb" "sdc"

TASK OK
```

Note: You can also pass the `--add-datastore` parameter here, to automatically create a datastore from the disk.

You can use `disk fs list` and `disk zpool list` to keep track of your filesystems and zpools respectively.

If a disk supports S.M.A.R.T. capability, and you have this enabled, you can display S.M.A.R.T. attributes using the command:

```
# proxmox-backup-manager disk smart-attributes sdX
```

3.2.3 Datastore Configuration

You can configure multiple datastores. Minimum one datastore needs to be configured. The datastore is identified by a simple *name* and points to a directory on the filesystem. Each datastore also has associated retention settings of how many backup snapshots for each interval of `hourly`, `daily`, `weekly`, `monthly`, `yearly` as well as a time-independent number of backups to keep in that store. *Pruning* and *garbage collection* can also be configured to run periodically based on a configured *schedule* per datastore.

The following command creates a new datastore called `store1` on `/backup/disk1/store1`

```
# proxmox-backup-manager datastore create store1 /backup/disk1/store1
```

To list existing datastores run:

```
# proxmox-backup-manager datastore list
```

name	path	comment
store1	/backup/disk1/store1	This is my default storage.

You can change settings of a datastore, for example to set a prune and garbage collection schedule or retention settings using `update` subcommand and view a datastore with the `show` subcommand:

```
# proxmox-backup-manager datastore update store1 --keep-last 7 --prune-schedule daily --gc-
↪schedule 'Tue 04:27'
# proxmox-backup-manager datastore show store1
```

Name	Value
name	store1
path	/backup/disk1/store1
comment	This is my default storage.
gc-schedule	Tue 04:27
keep-last	7
prune-schedule	daily

Finally, it is possible to remove the datastore configuration:

```
# proxmox-backup-manager datastore remove store1
```

Note: The above command removes only the datastore configuration. It does not delete any data from the underlying directory.

File Layout

After creating a datastore, the following default layout will appear:

```
# ls -arilh /backup/disk1/store1
276493 -rw-r--r-- 1 backup backup       0 Jul  8 12:35 .lock
276490 drwxr-x--- 1 backup backup 1064960 Jul  8 12:35 .chunks
```

.lock is an empty file used for process locking.

The *.chunks* directory contains folders, starting from *0000* and taking hexadecimal values until *ffff*. These directories will store the chunked data after a backup operation has been executed.

```
# ls -arilh /backup/disk1/store1/.chunks
545824 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 ffff
545823 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fffe
415621 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fffd
415620 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fffc
353187 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fffb
344995 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fffa
144079 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fff9
144078 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fff8
144077 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fff7
...
403180 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 000c
403179 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 000b
403177 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 000a
402530 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0009
402513 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0008
402509 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0007
276509 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0006
276508 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0005
276507 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0004
276501 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0003
276499 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0002
276498 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0001
276494 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0000
276489 drwxr-xr-x 3 backup backup 4.0K Jul  8 12:35 ..
276490 drwxr-x--- 1 backup backup 1.1M Jul  8 12:35 .
```

3.2.4 User Management

Proxmox Backup Server supports several authentication realms, and you need to choose the realm when you add a new user. Possible realms are:

pam Linux PAM standard authentication. Use this if you want to authenticate as Linux system user (Users need to exist on the system).

pbs Proxmox Backup Server realm. This type stores hashed passwords in `/etc/proxmox-backup/shadow.json`.

After installation, there is a single user `root@pam`, which corresponds to the Unix superuser. You can use the `proxmox-backup-manager` command line tool to list or manipulate users:

```
# proxmox-backup-manager user list
```

userid	enable	expire	firstname	lastname	email	comment
root@pam	1					Superuser

The superuser has full administration rights on everything, so you normally want to add other users with less privileges:

```
# proxmox-backup-manager user create john@pbs --email john@example.com
```

The create command lets you specify many options like `--email` or `--password`. You can update or change any of them using the update command later:

```
# proxmox-backup-manager user update john@pbs --firstname John --lastname Smith
# proxmox-backup-manager user update john@pbs --comment "An example user."
```

Todo: Mention how to set password without passing plaintext password as cli argument.

The resulting user list looks like this:

```
# proxmox-backup-manager user list
```

userid	enable	expire	firstname	lastname	email	comment
john@pbs	1		John	Smith	john@example.com	An example user.
root@pam	1					Superuser

Newly created users do not have any permissions. Please read the next section to learn how to set access permissions.

If you want to disable a user account, you can do that by setting `--enable` to `0`

```
# proxmox-backup-manager user update john@pbs --enable 0
```

Or completely remove the user with:

```
# proxmox-backup-manager user remove john@pbs
```

3.2.5 Access Control

By default new users do not have any permission. Instead you need to specify what is allowed and what is not. You can do this by assigning roles to users on specific objects like datastores or remotes. The following roles exist:

NoAccess Disable Access - nothing is allowed.

Admin Can do anything.

Audit Can view things, but is not allowed to change settings.

DatastoreAdmin Can do anything on datastores.

DatastoreAudit Can view datastore settings and list content. But is not allowed to read the actual data.

DatastoreReader Can Inspect datastore content and can do restores.

DatastoreBackup Can backup and restore owned backups.

DatastorePowerUser Can backup, restore, and prune owned backups.

RemoteAdmin Can do anything on remotes.

RemoteAudit Can view remote settings.

RemoteSyncOperator Is allowed to read data from a remote.

You can use the `acl` subcommand to manage and monitor user permissions. For example, the command below will add the user `john@pbs` as a **DatastoreAdmin** for the data store `store1`, located at `/backup/disk1/store1`:

```
# proxmox-backup-manager acl update /datastore/store1 DatastoreAdmin --userid john@pbs
```

You can monitor the roles of each user using the following command:

```
# proxmox-backup-manager acl list
```

ugid	path	propagate	roleid
john@pbs	/datastore/disk1	1	DatastoreAdmin

A single user can be assigned multiple permission sets for different data stores.

Note: Naming convention is important here. For data stores on the host, you must use the convention `/datastore/{storename}`. For example, to set permissions for a data store mounted at `/mnt/backup/disk4/store2`, you would use `/datastore/store2` for the path. For remote stores, use the convention `/remote/{remote}/{storename}`, where `{remote}` signifies the name of the remote (see *Remote* below) and `{storename}` is the name of the data store on the remote.

3.2.6 Network Management

Proxmox Backup Server provides an interface for network configuration, through the `network` subcommand. This allows you to carry out some basic network management tasks such as adding, configuring and removing network interfaces.

To get a list of available interfaces, use the following command:

```
# proxmox-backup-manager network list
┌──────┬────────┬───────────┬────────┬─────────┬──────────────┬─────────┬──────────
↪slaves │
╞══════╪════════╪═══════════╪════════╪═════════╪══════════════╪═════════╪══════════
│ bond0 │ bond   │         1 │ manual │         │              │         │ ens18␣
↪ens19  │
├──────┼────────┼───────────┼────────┼─────────┼──────────────┼─────────┼──────────
│ ens18 │ eth    │         1 │ manual │         │              │         │ ␣
↪      │
├──────┼────────┼───────────┼────────┼─────────┼──────────────┼─────────┼──────────
│ ens19 │ eth    │         1 │ manual │         │              │         │ ␣
↪      │
├──────┼────────┼───────────┼────────┼─────────┼──────────────┼─────────┼──────────
│ vmbr0 │ bridge │         1 │ static │         │ x.x.x.x/x    │ x.x.x.x │ bond0 ␣
↪      │
└──────┴────────┴───────────┴────────┴─────────┴──────────────┴─────────┴──────────
```

To add a new network interface, use the `create` subcommand with the relevant parameters. The following command shows a template for creating a new bridge:

```
# proxmox-backup-manager network create vmbr1 --autostart true --cidr x.x.x.x/x --gateway x.x.
↪x.x --bridge_ports iface_name --type bridge
```

You can make changes to the configuration of a network interface with the `update` subcommand:

```
# proxmox-backup-manager network update vmbr1 --cidr y.y.y.y/y
```

You can also remove a network interface:

```
# proxmox-backup-manager network remove vmbr1
```

To view the changes made to the network configuration file, before committing them, use the command:

```
# proxmox-backup-manager network changes
```

If you would like to cancel all changes at this point, you can do this using:

```
# proxmox-backup-manager network revert
```

If you are happy with the changes and would like to write them into the configuration file, the command is:

```
# proxmox-backup-manager network reload
```

You can also configure DNS settings using the `dns` subcommand of `proxmox-backup-manager`.

3.2.7 Remote

A remote refers to a separate Proxmox Backup Server installation and a user on that installation, from which you can *sync* datastores to a local datastore with a *Sync Job*.

To add a remote, you need its hostname or ip, a userid and password on the remote, and its certificate fingerprint. To get the fingerprint, use the `proxmox-backup-manager cert info` command on the remote.

```
# proxmox-backup-manager cert info |grep Fingerprint
Fingerprint (sha256): 64:d3:ff:3a:50:38:53:5a:9b:f7:50:...:ab:fe
```

Using the information specified above, add the remote with:

```
# proxmox-backup-manager remote create pbs2 --host pbs2.mydomain.example --userid sync@pam --
↪password 'SECRET' --fingerprint 64:d3:ff:3a:50:38:53:5a:9b:f7:50:...:ab:fe
```

Use the `list`, `show`, `update`, `remove` subcommands of `proxmox-backup-manager remote` to manage your remotes:

```
# proxmox-backup-manager remote update pbs2 --host pbs2.example
# proxmox-backup-manager remote list
```

name	host	userid	fingerprint	comment
pbs2	pbs2.example	sync@pam	64:d3:ff:3a:50:38:53:5a:9b:f7:50:...:ab:fe	

```
# proxmox-backup-manager remote remove pbs2
```

3.2.8 Sync Jobs

Sync jobs are configured to pull the contents of a datastore on a *Remote* to a local datastore. You can either start the sync job manually on the GUI or provide it with a *schedule* to run regularly. The `proxmox-backup-manager sync-job` command is used to manage sync jobs:

```
# proxmox-backup-manager sync-job create pbs2-local --remote pbs2 --remote-store local --
↪store local --schedule 'Wed 02:30'
# proxmox-backup-manager sync-job update pbs2-local --comment 'offsite'
# proxmox-backup-manager sync-job list
```

id	store	remote	remote-store	schedule	comment
pbs2-local	local	pbs2	local	Wed 02:30	offsite

```
# proxmox-backup-manager sync-job remove pbs2-local
```

3.2.9 Garbage Collection

You can monitor and run *garbage collection* on the Proxmox Backup Server using the `garbage-collection` subcommand of `proxmox-backup-manager`. You can use the `start` subcommand to manually start garbage collection on an entire data store and the `status` subcommand to see attributes relating to the *garbage collection*.

3.3 Backup Client usage

The command line client is called **`proxmox-backup-client`**.

3.3.1 Repository Locations

The client uses the following notation to specify a datastore repository on the backup server.

[[username@]server:]datastore

The default value for `username` ist `root`. If no server is specified, the default is the local host (`localhost`).

You can pass the repository with the `--repository` command line option, or by setting the `PBS_REPOSITORY` environment variable.

3.3.2 Environment Variables

`PBS_REPOSITORY` The default backup repository.

`PBS_PASSWORD` When set, this value is used for the password required for the backup server.

`PBS_ENCRYPTION_PASSWORD` When set, this value is used to access the secret encryption key (if protected by password).

`PBS_FINGERPRINT` When set, this value is used to verify the server certificate (only used if the system CA certificates cannot validate the certificate).

3.3.3 Output Format

Most commands support the `--output-format` parameter. It accepts the following values:

`text` Text format (default). Structured data is rendered as a table.

`json` JSON (single line).

`json-pretty` JSON (multiple lines, nicely formatted).

Please use the following environment variables to modify output behavior:

`PROXMOX_OUTPUT_FORMAT` Defines the default output format.

`PROXMOX_OUTPUT_NO_BORDER` If set (to any value), do not render table borders.

`PROXMOX_OUTPUT_NO_HEADER` If set (to any value), do not render table headers.

Note: The `text` format is designed to be human readable, and not meant to be parsed by automation tools. Please use the `json` format if you need to process the output.

3.3.4 Creating Backups

This section explains how to create a backup from within the machine. This can be a physical host, a virtual machine, or a container. Such backups may contain file and image archives. There are no restrictions in this case.

Note: If you want to backup virtual machines or containers on Proxmox VE, see *Proxmox VE integration*.

For the following example you need to have a backup server set up, working credentials and need to know the repository name. In the following examples we use `backup-server:store1`.

```
# proxmox-backup-client backup root.pxar:/ --repository backup-server:store1
Starting backup: host/elsa/2019-12-03T09:35:01Z
Client name: elsa
skip mount point: "/boot/efi"
skip mount point: "/dev"
skip mount point: "/run"
skip mount point: "/sys"
Uploaded 12129 chunks in 87 seconds (564 MB/s).
End Time: 2019-12-03T10:36:29+01:00
```

This will prompt you for a password and then uploads a file archive named `root.pxar` containing all the files in the `/` directory.

Caution: Please note that the proxmox-backup-client does not automatically include mount points. Instead, you will see a short `skip mount point` notice for each of them. The idea is to create a separate file archive for each mounted disk. You can explicitly include them using the `--include-dev` option (i.e. `--include-dev /boot/efi`). You can use this option multiple times for each mount point that should be included.

The `--repository` option can get quite long and is used by all commands. You can avoid having to enter this value by setting the environment variable `PBS_REPOSITORY`. Note that if you would like this to remain set over multiple sessions, you should instead add the below line to your `.bashrc` file.

```
# export PBS_REPOSITORY=backup-server:store1
```

After this you can execute all commands without specifying the `--repository` option.

One single backup is allowed to contain more than one archive. For example, if you want to backup two disks mounted at `/mmt/disk1` and `/mnt/disk2`:

```
# proxmox-backup-client backup disk1.pxar:/mnt/disk1 disk2.pxar:/mnt/disk2
```

This creates a backup of both disks.

The backup command takes a list of backup specifications, which include the archive name on the server, the type of the archive, and the archive source at the client. The format is:

<archive-name>.<type>:<source-path>

Common types are `.pxar` for file archives, and `.img` for block device images. To create a backup of a block device run the following command:

```
# proxmox-backup-client backup mydata.img:/dev/mylvm/mydata
```

Excluding files/folders from a backup

Sometimes it is desired to exclude certain files or folders from a backup archive. To tell the Proxmox Backup client when and how to ignore files and directories, place a text file called `.pxarexclude` in the filesystem hierarchy. Whenever the backup client encounters such a file in a directory, it interprets each line as glob match patterns for files and directories that are to be excluded from the backup.

The file must contain a single glob pattern per line. Empty lines are ignored. The same is true for lines starting with #, which indicates a comment. A `!` at the beginning of a line reverses the glob match pattern from an exclusion to an explicit inclusion. This makes it possible to exclude all entries in a directory except for a few single files/subdirectories. Lines ending in `/` match only on directories. The directory containing the `.pxarexclude` file is considered to be the root of the given patterns. It is only possible to match files in this directory and its subdirectories.

\ is used to escape special glob characters. ? matches any single character. * matches any character, including an empty string. ** is used to match subdirectories. It can be used to, for example, exclude all files ending in `.tmp` within the directory or subdirectories with the following pattern `**/*.tmp`. `[...]` matches a single character from any of the provided characters within the brackets. `[!...]` does the complementary and matches any single character not contained within the brackets. It is also possible to specify ranges with two characters separated by `-`. For example, `[a-z]` matches any lowercase alphabetic character and `[0-9]` matches any one single digit.

The order of the glob match patterns defines whether a file is included or excluded, that is to say later entries override previous ones. This is also true for match patterns encountered deeper down the directory tree, which can override a previous exclusion. Be aware that excluded directories will **not** be read by the backup client. Thus, a `.pxarexclude` file in an excluded subdirectory will have no effect. `.pxarexclude` files are treated as regular files and will be included in the backup archive.

For example, consider the following directory structure:

```
# ls -aR folder
folder/:
.  ..  .pxarexclude  subfolder0  subfolder1

folder/subfolder0:
.  ..  file0  file1  file2  file3  .pxarexclude

folder/subfolder1:
.  ..  file0  file1  file2  file3
```

The different `.pxarexclude` files contain the following:

```
# cat folder/.pxarexclude
/subfolder0/file1
/subfolder1/*
!/subfolder1/file2
```

```
# cat folder/subfolder0/.pxarexclude
file3
```

This would exclude `file1` and `file3` in `subfolder0` and all of `subfolder1` except `file2`.

Restoring this backup will result in:

```
ls -aR restored
restored/:
.  ..  .pxarexclude  subfolder0  subfolder1

restored/subfolder0:
.  ..  file0  file2  .pxarexclude

restored/subfolder1:
.  ..  file2
```

3.3.5 Encryption

Proxmox Backup supports client-side encryption with AES-256 in GCM mode. To set this up, you first need to create an encryption key:

```
# proxmox-backup-client key create my-backup.key
Encryption Key Password: **********
```

The key is password protected by default. If you do not need this extra protection, you can also create it without a password:

```
# proxmox-backup-client key create /path/to/my-backup.key --kdf none
```

Having created this key, it is now possible to create an encrypted backup, by passing the `--keyfile` parameter, with the path to the key file.

```
# proxmox-backup-client backup etc.pxar:/etc --keyfile /path/to/my-backup.key
Password: *********
Encryption Key Password: **************
...
```

Note: If you do not specify the name of the backup key, the key will be created in the default location `~/.config/proxmox-backup/encryption-key.json`. `proxmox-backup-client` will also search this location by default, in case the `--keyfile` parameter is not specified.

You can avoid entering the passwords by setting the environment variables `PBS_PASSWORD` and `PBS_ENCRYPTION_PASSWORD`.

Using a master key to store and recover encryption keys

You can also use `proxmox-backup-client key` to create an RSA public/private key pair, which can be used to store an encrypted version of the symmetric backup encryption key alongside each backup and recover it later.

To set up a master key:

1. Create an encryption key for the backup:

   ```
   # proxmox-backup-client key create
   creating default key at: "~/.config/proxmox-backup/encryption-key.json"
   Encryption Key Password: **********
   ...
   ```

   The resulting file will be saved to `~/.config/proxmox-backup/encryption-key.json`.

2. Create an RSA public/private key pair:

   ```
   # proxmox-backup-client key create-master-key
   Master Key Password: *********
   ...
   ```

   This will create two files in your current directory, `master-public.pem` and `master-private.pem`.

3. Import the newly created `master-public.pem` public certificate, so that `proxmox-backup-client` can find and use it upon backup.

   ```
   # proxmox-backup-client key import-master-pubkey /path/to/master-public.pem
   Imported public master key to "~/.config/proxmox-backup/master-public.pem"
   ```

4. With all these files in place, run a backup job:

   ```
   # proxmox-backup-client backup etc.pxar:/etc
   ```

   The key will be stored in your backup, under the name `rsa-encrypted.key`.

   Note: The `--keyfile` parameter can be excluded, if the encryption key is in the default path. If you specified another path upon creation, you must pass the `--keyfile` parameter.

5. To test that everything worked, you can restore the key from the backup:

   ```
   # proxmox-backup-client restore /path/to/backup/ rsa-encrypted.key /path/to/target
   ```

   Note: You should not need an encryption key to extract this file. However, if a key exists at the default location (`~/.config/proxmox-backup/encryption-key.json`) the program

will prompt you for an encryption key password. Simply moving `encryption-key.json` out of this directory will fix this issue.

6. Then, use the previously generated master key to decrypt the file:

```
# openssl rsautl -decrypt -inkey master-private.pem -in rsa-encrypted.key -out /path/to/
↪target
Enter pass phrase for ./master-private.pem: *********
```

7. The target file will now contain the encryption key information in plain text. The success of this can be confirmed by passing the resulting `json` file, with the `--keyfile` parameter, when decrypting files from the backup.

Warning: Without their key, backed up files will be inaccessible. Thus, you should keep keys ordered and in a place that is separate from the contents being backed up. It can happen, for example, that you back up an entire system, using a key on that system. If the system then becomes inaccessable for any reason and needs to be restored, this will not be possible as the encryption key will be lost along with the broken system. In preparation for the worst case scenario, you should consider keeping a paper copy of this key locked away in a safe place.

3.3.6 Restoring Data

The regular creation of backups is a necessary step to avoiding data loss. More importantly, however, is the restoration. It is good practice to perform periodic recovery tests to ensure that you can access the data in case of problems.

First, you need to find the snapshot which you want to restore. The snapshot command provides a list of all the snapshots on the server:

```
# proxmox-backup-client snapshots
┌─────────────────────────────────┬──────────────┬────────────────────────────────────┐
│ snapshot                        │         size │ files                              │
╞═════════════════════════════════╪══════════════╪════════════════════════════════════╡
│ host/elsa/2019-12-03T09:30:15Z  │  51788646825 │ root.pxar catalog.pcat1 index.json │
├─────────────────────────────────┼──────────────┼────────────────────────────────────┤
│ host/elsa/2019-12-03T09:35:01Z  │  51790622048 │ root.pxar catalog.pcat1 index.json │
└─────────────────────────────────┴──────────────┴────────────────────────────────────┘
...
```

You can inspect the catalog to find specific files.

```
# proxmox-backup-client catalog dump host/elsa/2019-12-03T09:35:01Z
...
d "./root.pxar.didx/etc/cifs-utils"
l "./root.pxar.didx/etc/cifs-utils/idmap-plugin"
d "./root.pxar.didx/etc/console-setup"
...
```

The restore command lets you restore a single archive from the backup.

```
# proxmox-backup-client restore host/elsa/2019-12-03T09:35:01Z root.pxar /target/path/
```

To get the contents of any archive, you can restore the `index.json` file in the repository to the target path '-'. This will dump the contents to the standard output.

```
# proxmox-backup-client restore host/elsa/2019-12-03T09:35:01Z index.json -
```

Interactive Restores

If you only want to restore a few individual files, it is often easier to use the interactive recovery shell.

```
# proxmox-backup-client catalog shell host/elsa/2019-12-03T09:35:01Z root.pxar
Starting interactive shell
pxar:/ > ls
bin        boot       dev        etc        home       lib        lib32
...
```

The interactive recovery shell is a minimalistic command line interface that utilizes the metadata stored in the catalog to quickly list, navigate and search files in a file archive. To restore files, you can select them individually or match them with a glob pattern.

Using the catalog for navigation reduces the overhead considerably because only the catalog needs to be downloaded and, optionally, decrypted. The actual chunks are only accessed if the metadata in the catalog is not enough or for the actual restore.

Similar to common UNIX shells `cd` and `ls` are the commands used to change working directory and list directory contents in the archive. `pwd` shows the full path of the current working directory with respect to the archive root.

Being able to quickly search the contents of the archive is a commmonly needed feature. That's where the catalog is most valuable. For example:

```
pxar:/ > find etc/**/*.txt --select
"/etc/X11/rgb.txt"
pxar:/ > list-selected
etc/**/*.txt
pxar:/ > restore-selected /target/path
...
```

This will find and print all files ending in `.txt` located in `etc/` or a subdirectory and add the corresponding pattern to the list for subsequent restores. `list-selected` shows these patterns and `restore-selected` finally restores all files in the archive matching the patterns to `/target/path` on the local host. This will scan the whole archive.

With `restore /target/path` you can restore the sub-archive given by the current working directory to the local target path `/target/path` on your host. By additionally passing a glob pattern with `--pattern <glob>`, the restore is further limited to files matching the pattern. For example:

```
pxar:/ > cd /etc/
pxar:/etc/ > restore /target/ --pattern **/*.conf
...
```

The above will scan trough all the directories below `/etc` and restore all files ending in `.conf`.

Todo: Explain interactive restore in more detail

Mounting of Archives via FUSE

The *FUSE* implementation for the pxar archive allows you to mount a file archive as a read-only filesystem to a mountpoint on your host.

```
# proxmox-backup-client mount host/backup-client/2020-01-29T11:29:22Z root.pxar /mnt/
↪mountpoint
# ls /mnt/mountpoint
bin   dev  home  lib32  libx32      media  opt   root  sbin  sys  usr
boot  etc  lib   lib64  lost+found  mnt    proc  run   srv   tmp  var
```

This allows you to access the full contents of the archive in a seamless manner.

Note: As the FUSE connection needs to fetch and decrypt chunks from the backup server's datastore, this can cause some additional network and CPU load on your host, depending on the operations you perform on the mounted filesystem.

To unmount the filesystem use the `umount` command on the mountpoint:

```
# umount /mnt/mountpoint
```

3.3.7 Login and Logout

The client tool prompts you to enter the logon password as soon as you want to access the backup server. The server checks your credentials and responds with a ticket that is valid for two hours. The client tool automatically stores that ticket and uses it for further requests to this server.

You can also manually trigger this login/logout using the login and logout commands:

```
# proxmox-backup-client login
Password: **********
```

To remove the ticket, issue a logout:

```
# proxmox-backup-client logout
```

3.3.8 Pruning and Removing Backups

You can manually delete a backup snapshot using the `forget` command:

```
# proxmox-backup-client forget <snapshot>
```

Caution: This command removes all archives in this backup snapshot. They will be inaccessible and unrecoverable.

Although manual removal is sometimes required, the `prune` command is normally used to systematically delete older backups. Prune lets you specify which backup snapshots you want to keep. The following retention options are available:

`--keep-last <N>` Keep the last <N> backup snapshots.

`--keep-hourly <N>` Keep backups for the last <N> hours. If there is more than one backup for a single hour, only the latest is kept.

`--keep-daily <N>` Keep backups for the last <N> days. If there is more than one backup for a single day, only the latest is kept.

`--keep-weekly <N>` Keep backups for the last <N> weeks. If there is more than one backup for a single week, only the latest is kept.

> **Note:** Weeks start on Monday and end on Sunday. The software uses the ISO week date system and handles weeks at the end of the year correctly.

`--keep-monthly <N>` Keep backups for the last <N> months. If there is more than one backup for a single month, only the latest is kept.

`--keep-yearly <N>` Keep backups for the last <N> years. If there is more than one backup for a single year, only the latest is kept.

The retention options are processed in the order given above. Each option only covers backups within its time period. The next option does not take care of already covered backups. It will only consider older backups.

Unfinished and incomplete backups will be removed by the prune command unless they are newer than the last successful backup. In this case, the last failed backup is retained.

```
# proxmox-backup-client prune <group> --keep-daily 7 --keep-weekly 4 --keep-monthly 3
```

You can use the `--dry-run` option to test your settings. This only shows the list of existing snapshots and what actions prune would take.

```
# proxmox-backup-client prune host/elsa --dry-run --keep-daily 1 --keep-weekly 3
```

snapshot	keep
host/elsa/2019-12-04T13:20:37Z	1
host/elsa/2019-12-03T09:35:01Z	0
host/elsa/2019-11-22T11:54:47Z	1
host/elsa/2019-11-21T12:36:25Z	0
host/elsa/2019-11-10T10:42:20Z	1

Note: Neither the `prune` command nor the `forget` command free space in the chunk-store. The chunk-store still contains the data blocks. To free space you need to perform *Garbage Collection*.

3.3.9 Garbage Collection

The `prune` command removes only the backup index files, not the data from the data store. This task is left to the garbage collection command. It is recommended to carry out garbage collection on a regular basis.

The garbage collection works in two phases. In the first phase, all data blocks that are still in use are marked. In the second phase, unused data blocks are removed.

Note: This command needs to read all existing backup index files and touches the complete chunk-store. This can take a long time depending on the number of chunks and the speed of the underlying disks.

Note: The garbage collection will only remove chunks that haven't been used for at least one day (exactly 24h 5m). This grace period is necessary because chunks in use are marked by touching the chunk which updates the `atime` (access time) property. Filesystems are mounted with the `relatime` option by default. This results in a better performance by only updating the `atime` property if the last access has been at least 24 hours ago. The downside is, that touching a chunk within these 24 hours will not always update its `atime` property.

Chunks in the grace period will be logged at the end of the garbage collection task as *Pending removals*.

```
# proxmox-backup-client garbage-collect
starting garbage collection on store store2
Start GC phase1 (mark used chunks)
Start GC phase2 (sweep unused chunks)
percentage done: 1, chunk count: 219
```

(continues on next page)

(continued from previous page)

```
percentage done: 2, chunk count: 453
...
percentage done: 99, chunk count: 21188
Removed bytes: 411368505
Removed chunks: 203
Original data bytes: 327160886391
Disk bytes: 52767414743 (16 %)
Disk chunks: 21221
Average chunk size: 2486565
TASK OK
```

Todo: howto run garbage-collection at regular intervalls (cron)

3.3.10 Benchmarking

The backup client also comes with a benchmarking tool. This tool measures various metrics relating to compression and encryption speeds. You can run a benchmark using the `benchmark` subcommand of `proxmox-backup-client`:

```
# proxmox-backup-client benchmark
Uploaded 656 chunks in 5 seconds.
Time per request: 7659 microseconds.
TLS speed: 547.60 MB/s
SHA256 speed: 585.76 MB/s
Compression speed: 1923.96 MB/s
Decompress speed: 7885.24 MB/s
AES256/GCM speed: 3974.03 MB/s
```

Name	Value
TLS (maximal backup upload speed)	547.60 MB/s (93%)
SHA256 checksum computation speed	585.76 MB/s (28%)
ZStd level 1 compression speed	1923.96 MB/s (89%)
ZStd level 1 decompression speed	7885.24 MB/s (98%)
AES256 GCM encryption speed	3974.03 MB/s (104%)

Note: The percentages given in the output table correspond to a comparison against a Ryzen 7 2700X. The TLS test connects to the local host, so there is no network involved.

You can also pass the `--output-format` parameter to output stats in `json`, rather than the default table format.

3.4 Proxmox VE integration

You need to define a new storage with type 'pbs' on your Proxmox VE node. The following example uses `store2` as storage name, and assumes the server address is `localhost`, and you want to connect as `user1@pbs`.

```
# pvesm add pbs store2 --server localhost --datastore store2
# pvesm set store2 --username user1@pbs --password <secret>
```

If your backup server uses a self signed certificate, you need to add the certificate fingerprint to the configuration. You can get the fingerprint by running the following command on the backup server:

```
# proxmox-backup-manager cert  info |grep Fingerprint
Fingerprint (sha256): 64:d3:ff:3a:50:38:53:5a:9b:f7:50:...:ab:fe
```

Please add that fingerprint to your configuration to establish a trust relationship:

```
# pvesm set store2 --fingerprint  64:d3:ff:3a:50:38:53:5a:9b:f7:50:...:ab:fe
```

After that you should be able to see storage status with:

```
# pvesm status --storage store2
Name             Type     Status           Total            Used       Available        %
store2            pbs     active      3905109820      1336687816      2568422004   34.23%
```

3.5 Command Line Tools

3.5.1 proxmox-backup-client

This is just a test.

Note: No further info.

3.5.2 proxmox-backup-manager

This is just a test.

Note: No further info.

3.5.3 pxar

Description

`pxar` is a command line utility to create and manipulate archives in the *Proxmox File Archive Format (.pxar)*. It is inspired by casync file archive format, which caters to a similar use-case. The `.pxar` format is adapted to fulfill the specific needs of the Proxmox Backup Server, for example, efficient storage of hardlinks. The format is designed to reduce storage space needed on the server by achieving a high level of deduplication.

Creating an Archive

Run the following command to create an archive of a folder named `source`:

```
# pxar create archive.pxar /path/to/source
```

This will create a new archive called `archive.pxar` with the contents of the `source` folder.

Note: `pxar` will not overwrite any existing archives. If an archive with the same name is already present in the target folder, the creation will fail.

By default, `pxar` will skip certain mountpoints and will not follow device boundaries. This design decision is based on the primary use case of creating archives for backups. It makes sense to not back up the contents of certain temporary or system specific files. To alter this behavior and follow device boundaries, use the `--all-file-systems` flag.

It is possible to exclude certain files and/or folders from the archive by passing the `--exclude` parameter with `gitignore`-style match patterns.

For example, you can exclude all files ending in `.txt` from the archive by running:

```
# pxar create archive.pxar /path/to/source --exclude '**/*.txt'
```

Be aware that the shell itself will try to expand all of the glob patterns before invoking `pxar`. In order to avoid this, all globs have to be quoted correctly.

It is possible to pass the `--exclude` parameter multiple times, in order to match more than one pattern. This allows you to use more complex file exclusion/inclusion behavior. However, it is recommended to use `.pxarexclude` files instead for such cases.

For example you might want to exclude all `.txt` files except for a specific one from the archive. This is achieved via the negated match pattern, prefixed by `!`. All the glob patterns are relative to the `source` directory.

```
# pxar create archive.pxar /path/to/source --exclude '**/*.txt' --exclude '!/folder/file.txt'
```

Note: The order of the glob match patterns matters as later ones override previous ones. Permutations of the same patterns lead to different results.

`pxar` will store the list of glob match patterns passed as parameters via the command line, in a file called `.pxarexclude-cli` at the root of the archive. If a file with this name is already present in the source folder during archive creation, this file is not included in the archive and the file containing the new patterns is added to the archive instead, the original file is not altered.

A more convenient and persistent way to exclude files from the archive is by placing the glob match patterns in `.pxarexclude` files. It is possible to create and place these files in any directory of the filesystem tree. These files must contain one pattern per line, again later patterns win over previous ones. The patterns control file exclusions of files present within the given directory or further below it in the tree. The behavior is the same as described in *Creating Backups*.

Extracting an Archive

An existing archive, `archive.pxar`, is extracted to a `target` directory with the following command:

```
# pxar extract archive.pxar /path/to/target
```

If no target is provided, the content of the archive is extracted to the current working directory.

In order to restore only parts of an archive, single files, and/or folders, it is possible to pass the corresponding glob match patterns as additional parameters or to use the patterns stored in a file:

```
# pxar extract etc.pxar /restore/target/etc --pattern '**/*.conf'
```

The above example restores all `.conf` files encountered in any of the sub-folders in the archive `etc.pxar` to the target `/restore/target/etc`. A path to the file containing match patterns can be specified using the `--files-from` parameter.

List the Contents of an Archive

To display the files and directories contained in an archive `archive.pxar`, run the following command:

```
# pxar list archive.pxar
```

This displays the full path of each file or directory with respect to the archives root.

Mounting an Archive

`pxar` allows you to mount and inspect the contents of an archive via FUSE. In order to mount an archive named `archive.pxar` to the mountpoint `/mnt`, run the command:

```
# pxar mount archive.pxar /mnt
```

Once the archive is mounted, you can access its content under the given mountpoint.

```
# cd /mnt
# ls
bin   dev  home  lib32  libx32      media  opt   root  sbin  sys  usr
boot  etc  lib   lib64  lost+found  mnt    proc  run   srv   tmp  var
```

3.6 Service Daemons

3.6.1 `proxmox-backup-proxy`

This is just a test.

Note: No further info.

CHAPTER

FOUR

HOST SYSTEM ADMINISTRATION

Proxmox Backup is based on the famous Debian Linux distribution. That means that you have access to the whole world of Debian packages, and the base system is well documented. The Debian Administrator's Handbook is available online, and provides a comprehensive introduction to the Debian operating system.

A standard Proxmox Backup installation uses the default repositories from Debian, so you get bug fixes and security updates through that channel. In addition, we provide our own package repository to roll out all Proxmox related packages. This includes updates to some Debian packages when necessary.

We also deliver a specially optimized Linux kernel, where we enable all required virtualization and container features. That kernel includes drivers for ZFS, and several hardware drivers. For example, we ship Intel network card drivers to support their newest hardware.

The following sections will concentrate on backup related topics. They either explain things which are different on Proxmox Backup, or tasks which are commonly used on Proxmox Backup. For other topics, please refer to the standard Debian documentation.

4.1 ZFS on Linux

ZFS is a combined file system and logical volume manager designed by Sun Microsystems. There is no need to manually compile ZFS modules - all packages are included.

By using ZFS, it's possible to achieve maximum enterprise features with low budget hardware, but also high performance systems by leveraging SSD caching or even SSD only setups. ZFS can replace cost intense hardware raid cards by moderate CPU and memory load combined with easy management.

General ZFS advantages

- Easy configuration and management with GUI and CLI.
- Reliable
- Protection against data corruption
- Data compression on file system level
- Snapshots
- Copy-on-write clone
- Various raid levels: RAID0, RAID1, RAID10, RAIDZ-1, RAIDZ-2 and RAIDZ-3
- Can use SSD for cache
- Self healing
- Continuous integrity checking
- Designed for high storage capacities

- Asynchronous replication over network
- Open Source
- Encryption

4.1.1 Hardware

ZFS depends heavily on memory, so you need at least 8GB to start. In practice, use as much you can get for your hardware/budget. To prevent data corruption, we recommend the use of high quality ECC RAM.

If you use a dedicated cache and/or log disk, you should use an enterprise class SSD (e.g. Intel SSD DC S3700 Series). This can increase the overall performance significantly.

IMPORTANT: Do not use ZFS on top of hardware controller which has its own cache management. ZFS needs to directly communicate with disks. An HBA adapter is the way to go, or something like LSI controller flashed in `IT` mode.

4.1.2 ZFS Administration

This section gives you some usage examples for common tasks. ZFS itself is really powerful and provides many options. The main commands to manage ZFS are *zfs* and *zpool*. Both commands come with great manual pages, which can be read with:

```
# man zpool
# man zfs
```

Create a new zpool

To create a new pool, at least one disk is needed. The *ashift* should have the same sector-size (2 power of *ashift*) or larger as the underlying disk.

```
# zpool create -f -o ashift=12 <pool> <device>
```

Create a new pool with RAID-0

Minimum 1 disk

```
# zpool create -f -o ashift=12 <pool> <device1> <device2>
```

Create a new pool with RAID-1

Minimum 2 disks

```
# zpool create -f -o ashift=12 <pool> mirror <device1> <device2>
```

Create a new pool with RAID-10

Minimum 4 disks

```
# zpool create -f -o ashift=12 <pool> mirror <device1> <device2> mirror <device3> <device4>
```

Create a new pool with RAIDZ-1

Minimum 3 disks

```
# zpool create -f -o ashift=12 <pool> raidz1 <device1> <device2> <device3>
```

Create a new pool with RAIDZ-2

Minimum 4 disks

```
# zpool create -f -o ashift=12 <pool> raidz2 <device1> <device2> <device3> <device4>
```

Create a new pool with cache (L2ARC)

It is possible to use a dedicated cache drive partition to increase the performance (use SSD).

As *<device>* it is possible to use more devices, like it's shown in "Create a new pool with RAID*".

```
# zpool create -f -o ashift=12 <pool> <device> cache <cache_device>
```

Create a new pool with log (ZIL)

It is possible to use a dedicated cache drive partition to increase the performance (SSD).

As *<device>* it is possible to use more devices, like it's shown in "Create a new pool with RAID*".

```
# zpool create -f -o ashift=12 <pool> <device> log <log_device>
```

Add cache and log to an existing pool

If you have a pool without cache and log. First partition the SSD in 2 partition with *parted* or *gdisk*

Important: Always use GPT partition tables.

The maximum size of a log device should be about half the size of physical memory, so this is usually quite small. The rest of the SSD can be used as cache.

```
# zpool add -f <pool> log <device-part1> cache <device-part2>
```

Changing a failed device

```
# zpool replace -f <pool> <old device> <new device>
```

Changing a failed bootable device

Depending on how Proxmox Backup was installed it is either using *grub* or *systemd-boot* as bootloader.

The first steps of copying the partition table, reissuing GUIDs and replacing the ZFS partition are the same. To make the system bootable from the new disk, different steps are needed which depend on the bootloader in use.

```
# sgdisk <healthy bootable device> -R <new device>
# sgdisk -G <new device>
# zpool replace -f <pool> <old zfs partition> <new zfs partition>
```

Note: Use the *zpool status -v* command to monitor how far the resilvering process of the new disk has progressed.

With *systemd-boot*:

```
# pve-efiboot-tool format <new disk's ESP>
# pve-efiboot-tool init <new disk's ESP>
```

Note: *ESP* stands for EFI System Partition, which is setup as partition #2 on bootable disks setup by the {pve} installer since version 5.4. For details, see xref:sysboot_systemd_boot_setup[Setting up a new partition for use as synced ESP].

With *grub*:

Usually *grub.cfg* is located in */boot/grub/grub.cfg*

```
# grub-install <new disk>
# grub-mkconfig -o /path/to/grub.cfg
```

Activate E-Mail Notification

ZFS comes with an event daemon, which monitors events generated by the ZFS kernel module. The daemon can also send emails on ZFS events like pool errors. Newer ZFS packages ship the daemon in a separate package, and you can install it using *apt-get*:

```
# apt-get install zfs-zed
```

To activate the daemon it is necessary to edit */etc/zfs/zed.d/zed.rc* with your favourite editor, and uncomment the *ZED_EMAIL_ADDR* setting:

```
ZED_EMAIL_ADDR="root"
```

Please note Proxmox Backup forwards mails to *root* to the email address configured for the root user.

IMPORTANT: The only setting that is required is *ZED_EMAIL_ADDR*. All other settings are optional.

Limit ZFS Memory Usage

It is good to use at most 50 percent (which is the default) of the system memory for ZFS ARC to prevent performance shortage of the host. Use your preferred editor to change the configuration in */etc/modprobe.d/zfs.conf* and insert:

```
options zfs zfs_arc_max=8589934592
```

This example setting limits the usage to 8GB.

Important: If your root file system is ZFS you must update your initramfs every time this value changes:

```
# update-initramfs -u
```

SWAP on ZFS

Swap-space created on a zvol may generate some troubles, like blocking the server or generating a high IO load, often seen when starting a Backup to an external Storage.

We strongly recommend to use enough memory, so that you normally do not run into low memory situations. Should you need or want to add swap, it is preferred to create a partition on a physical disk and use it as swapdevice. You can leave some space free for this purpose in the advanced options of the installer. Additionally, you can lower the *swappiness* value. A good value for servers is 10:

```
# sysctl -w vm.swappiness=10
```

To make the swappiness persistent, open */etc/sysctl.conf* with an editor of your choice and add the following line:

```
vm.swappiness = 10
```

Table 1: Linux kernel *swappiness* parameter values :widths:auto

Value	Strategy
vm.swappiness = 0	The kernel will swap only to avoid an 'out of memory' condition
vm.swappiness = 1	Minimum amount of swapping without disabling it entirely.
vm.swappiness = 10	Sometimes recommended to improve performance when sufficient memory exists in a system.
vm.swappiness = 60	The default value.
vm.swappiness = 100	The kernel will swap aggressively.

ZFS Compression

To activate compression: .. code-block:: console

    # zpool set compression=lz4 <pool>

We recommend using the *lz4* algorithm, since it adds very little CPU overhead. Other algorithms such as *lzjb* and *gzip-N* (where *N* is an integer *1-9* representing the compression ratio, 1 is fastest and 9 is best compression) are also available. Depending on the algorithm and how compressible the data is, having compression enabled can even increase I/O performance.

You can disable compression at any time with: .. code-block:: console

    # zfs set compression=off <dataset>

Only new blocks will be affected by this change.

ZFS Special Device

Since version 0.8.0 ZFS supports *special* devices. A *special* device in a pool is used to store metadata, deduplication tables, and optionally small file blocks.

A *special* device can improve the speed of a pool consisting of slow spinning hard disks with a lot of metadata changes. For example workloads that involve creating, updating or deleting a large

number of files will benefit from the presence of a *special* device. ZFS datasets can also be configured to store whole small files on the *special* device which can further improve the performance. Use fast SSDs for the *special* device.

Important: The redundancy of the *special* device should match the one of the pool, since the *special* device is a point of failure for the whole pool.

Warning: Adding a *special* device to a pool cannot be undone!

Create a pool with *special* device and RAID-1:

```
# zpool create -f -o ashift=12 <pool> mirror <device1> <device2> special mirror <device3>
↪<device4>
```

Adding a *special* device to an existing pool with RAID-1:

```
# zpool add <pool> special mirror <device1> <device2>
```

ZFS datasets expose the *special_small_blocks=<size>* property. *size* can be *0* to disable storing small file blocks on the *special* device or a power of two in the range between *512B* to *128K*. After setting the property new file blocks smaller than *size* will be allocated on the *special* device.

Important: If the value for *special_small_blocks* is greater than or equal to the *recordsize* (default *128K*) of the dataset, *all* data will be written to the *special* device, so be careful!

Setting the *special_small_blocks* property on a pool will change the default value of that property for all child ZFS datasets (for example all containers in the pool will opt in for small file blocks).

Opt in for all file smaller than 4K-blocks pool-wide:

```
# zfs set special_small_blocks=4K <pool>
```

Opt in for small file blocks for a single dataset:

```
# zfs set special_small_blocks=4K <pool>/<filesystem>
```

Opt out from small file blocks for a single dataset:

```
# zfs set special_small_blocks=0 <pool>/<filesystem>
```

Troubleshooting

Corrupted cachefile

In case of a corrupted ZFS cachefile, some volumes may not be mounted during boot until mounted manually later.

For each pool, run:

```
# zpool set cachefile=/etc/zfs/zpool.cache POOLNAME
```

and afterwards update the *initramfs* by running:

```
# update-initramfs -u -k all
```

and finally reboot your node.

Sometimes the ZFS cachefile can get corrupted, and *zfs-import-cache.service* doesn't import the pools that aren't present in the cachefile.

Another workaround to this problem is enabling the *zfs-import-scan.service*, which searches and imports pools via device scanning (usually slower).

APPENDIX

A

COMMAND SYNTAX

A.1 proxmox-backup-client

`proxmox-backup-client backup {<backupspec>} [OPTIONS]`

Create (host) backup.

<backupspec> <array> List of backup source specifications ([<label.ext>:<path>] ...)

Optional parameters:

--backup-id <string> Backup ID.

--backup-time <integer> (1547797308 - N) Backup time (Unix epoch.)

--backup-type <string> Backup type.

--chunk-size <integer> (64 - 4096) (default=4096) Chunk size in KB. Must be a power of 2.

--crypt-mode <string> (default=encrypt) Defines whether data is encrypted (using an AEAD cipher), only signed, or neither.

--entries-max <integer> (default=1048576) Max number of entries to hold in memory.

--exclude <array> List of paths or patterns for matching files to exclude.

--include-dev <array> Include mountpoints with same st_dev number (see `man fstat`) as specified files.

--keyfd <integer> (0 - N) Pass an encryption key via an already opened file descriptor.

--keyfile <string> Path to encryption key. All data will be encrypted using this key.

--repository <string> Repository URL.

--skip-lost-and-found <boolean> Skip lost+found directory.

--verbose <boolean> Verbose output.

`proxmox-backup-client benchmark [OPTIONS]`

Run benchmark tests

Optional parameters:

--keyfile <string> Path to encryption key. All data will be encrypted using this key.

--output-format <string> Output format.

--repository <string> Repository URL.

--verbose <boolean> Verbose output.

`proxmox-backup-client catalog dump <snapshot> [OPTIONS]`

Dump catalog.

<snapshot> <string> Snapshot path.

Optional parameters:

--keyfd <integer> (0 - N) Pass an encryption key via an already opened file descriptor.

--keyfile <string> Path to encryption key.

--repository <string> Repository URL.

`proxmox-backup-client catalog shell <snapshot> <archive-name> [OPTIONS]`

Shell to interactively inspect and restore snapshots.

<snapshot> <string> Group/Snapshot path.

<archive-name> <string> Backup archive name.

Optional parameters:

--keyfd <integer> (0 - N) Pass an encryption key via an already opened file descriptor.

--keyfile <string> Path to encryption key.

--repository <string> Repository URL.

`proxmox-backup-client files <snapshot> [OPTIONS]`

List snapshot files.

<snapshot> <string> Snapshot path.

Optional parameters:

--output-format <string> Output format.

--repository <string> Repository URL.

`proxmox-backup-client forget <snapshot> [OPTIONS]`

Forget (remove) backup snapshots.

<snapshot> <string> Snapshot path.

Optional parameters:

--repository <string> Repository URL.

`proxmox-backup-client garbage-collect [OPTIONS]`

Start garbage collection for a specific repository.

Optional parameters:

--output-format <string> Output format.

--repository <string> Repository URL.

`proxmox-backup-client help [{<command>}] [OPTIONS]`

Get help about specified command (or sub-command).

<command> <array> Command. This may be a list in order to spefify nested sub-commands.

Optional parameters:

--verbose <boolean> Verbose help.

`proxmox-backup-client key change-passphrase [<path>] [OPTIONS]`

Change the encryption key's password.

<path> <string> Key file. Without this the default key's password will be changed.

Optional parameters:

--kdf <string> (default=scrypt) Key derivation function for password protected encryption keys.

`proxmox-backup-client key create [<path>] [OPTIONS]`

Create a new encryption key.

<path> <string> Output file. Without this the key will become the new default encryption key.

Optional parameters:

--kdf <string> (default=scrypt) Key derivation function for password protected encryption keys.

`proxmox-backup-client key create-master-key`

Create an RSA public/private key pair used to put an encrypted version of the symmetric backup encryption key onto the backup server along with each backup.

`proxmox-backup-client key import-master-pubkey <path>`

Import an RSA public key used to put an encrypted version of the symmetric backup encryption key onto the backup server along with each backup.

<path> <string> Path to the PEM formatted RSA public key.

`proxmox-backup-client list [OPTIONS]`

List backup groups.

Optional parameters:

--output-format <string> Output format.

--repository <string> Repository URL.

`proxmox-backup-client login [OPTIONS]`

Try to login. If successful, store ticket.

Optional parameters:

--repository <string> Repository URL.

`proxmox-backup-client logout [OPTIONS]`

Logout (delete stored ticket).

Optional parameters:

--repository <string> Repository URL.

`proxmox-backup-client mount <snapshot> <archive-name> <target> [OPTIONS]`

Mount pxar archive.

<snapshot> <string> Group/Snapshot path.

<archive-name> <string> Backup archive name.

<target> <string> Target directory path.

Optional parameters:

--keyfile <string> Path to encryption key.

--repository <string> Repository URL.

--verbose <boolean> (default=false) Verbose output.

`proxmox-backup-client prune <group> [OPTIONS]`

Prune a backup repository.

<group> <string> Backup group.

Optional parameters:

--dry-run <boolean> Just show what prune would do, but do not delete anything.

--keep-daily <integer> (1 - N) Number of daily backups to keep.

--keep-hourly <integer> (1 - N) Number of hourly backups to keep.

--keep-last <integer> (1 - N) Number of backups to keep.

--keep-monthly <integer> (1 - N) Number of monthly backups to keep.

--keep-weekly <integer> (1 - N) Number of weekly backups to keep.

--keep-yearly <integer> (1 - N) Number of yearly backups to keep.

--output-format <string> Output format.

--quiet <boolean> Minimal output - only show removals.

--repository <string> Repository URL.

`proxmox-backup-client restore <snapshot> <archive-name> <target> [OPTIONS]`

Restore backup repository.

<snapshot> <string> Group/Snapshot path.

<archive-name> <string> Backup archive name.

<target> <string> Target directory path. Use '-' to write to standard output.

We do not extraxt '.pxar' archives when writing to standard output.

Optional parameters:

--allow-existing-dirs <boolean> Do not fail if directories already exists.

--crypt-mode <string> (default=encrypt) Defines whether data is encrypted (using an AEAD cipher), only signed, or neither.

--keyfd <integer> (0 - N) Pass an encryption key via an already opened file descriptor.

--keyfile <string> Path to encryption key. All data will be encrypted using this key.

--repository <string> Repository URL.

`proxmox-backup-client snapshots [<group>] [OPTIONS]`

List backup snapshots.

<group> <string> Backup group.

Optional parameters:

--output-format <string> Output format.

--repository <string> Repository URL.

`proxmox-backup-client status [OPTIONS]`

Get repository status.

Optional parameters:

--output-format <string> Output format.

--repository <string> Repository URL.

`proxmox-backup-client task list [OPTIONS]`

List running server tasks for this repo user

Optional parameters:

--all <boolean> Also list stopped tasks.

--limit <integer> (1 - 1000) (default=50) The maximal number of tasks to list.

--output-format <string> Output format.

--repository <string> Repository URL.

`proxmox-backup-client task log <upid> [OPTIONS]`

Display the task log.

<upid> <string> Unique Process/Task ID.

Optional parameters:

--repository <string> Repository URL.

`proxmox-backup-client task stop <upid> [OPTIONS]`

Try to stop a specific task.

<upid> <string> Unique Process/Task ID.

Optional parameters:

--repository <string> Repository URL.

`proxmox-backup-client upload-log <snapshot> <logfile> [OPTIONS]`

Upload backup log file.

<snapshot> <string> Group/Snapshot path.

<logfile> <string> The path to the log file you want to upload.

Optional parameters:

--crypt-mode <string> (default=encrypt) Defines whether data is encrypted (using an AEAD cipher), only signed, or neither.

--keyfd <integer> (0 - N) Pass an encryption key via an already opened file descriptor.

--keyfile <string> Path to encryption key. All data will be encrypted using this key.

--repository <string> Repository URL.

`proxmox-backup-client version [OPTIONS]`

Show client and optional server version

Optional parameters:

--output-format <string> Output format.

--repository <string> Repository URL.

A.1.1 Catalog Shell Commands

Those command are available when you start an intercative restore shell:

```
proxmox-backup-client shell <snapshot> <name.pxar>
```

`cd [<path>]`

Change the current working directory to the new directory

<path> <string> target path.

`clear-selected`

Clear the list of files selected for restore.

`deselect <path>`

Deselect an entry for restore.

This will return an error if the entry was not found in the list of entries selected for restore.

<path> <string> path to entry to remove from list.

`exit`

Exit the shell

`find <pattern> [OPTIONS]`

Find entries in the catalog matching the given match pattern.

<pattern> <string> Match pattern for matching files in the catalog.

Optional parameters:

--select <boolean> (default=false) Add matching filenames to list for restore.

`help [{<command>}] [OPTIONS]`

Get help about specified command (or sub-command).

<command> <array> Command. This may be a list in order to spefify nested sub-commands.

Optional parameters:

--verbose <boolean> Verbose help.

`list-selected [OPTIONS]`

List entries currently selected for restore.

Optional parameters:

--patterns <boolean> (default=false) List match patterns instead of the matching files.

`ls [<path>]`

List the content of working directory or given path.

<path> <string> target path.

`pwd`

List the current working directory.

`restore <target> [OPTIONS]`

Restore the sub-archive given by the current working directory to target.

By further providing a pattern, the restore can be limited to a narrower subset of this sub-archive. If pattern is not present or empty, the full archive is restored to target.

<target> <string> target path for restore on local filesystem.

Optional parameters:

--pattern <string> match pattern to limit files for restore.

`restore-selected <target>`

Restore the selected entries to the given target path.

Target must not exist on the clients filesystem.

<target> <string> target path for restore on local filesystem.

`select <path>`

Select an entry for restore.

This will return an error if the entry is already present in the list or if an invalid path was provided.

<path> <string> target path.

`stat <path>`

Read the metadata for a given directory entry.

This is expensive because the data has to be read from the pxar archive, which means reading over the network.

<path> <string> target path.

A.2 proxmox-backup-manager

`proxmox-backup-manager acl list [OPTIONS]`

Access Control list.

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager acl update <path> <role> [OPTIONS]`

Update Access Control List (ACLs).

<path> <string> Access control path.

<role> <string> Role

Optional parameters:

--delete <boolean> Remove permissions (instead of adding it).

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--group <string> Group ID

--propagate <boolean> (default=true) Allow to propagate (inherit) permissions.

--userid <string> User ID

`proxmox-backup-manager cert info`

Display node certificate information.

`proxmox-backup-manager cert update [OPTIONS]`

Update node certificates and generate all needed files/directories.

Optional parameters:

`--force <boolean>` Force generation of new SSL certifate.

`proxmox-backup-manager datastore create <name> <path> [OPTIONS]`

Create new datastore config.

`<name> <string>` Datastore name.

`<path> <string>` Directory name

Optional parameters:

`--comment <string>` Comment (single line).

`--gc-schedule <string>` Run garbage collection job at specified schedule.

`--keep-daily <integer> (1 - N)` Number of daily backups to keep.

`--keep-hourly <integer> (1 - N)` Number of hourly backups to keep.

`--keep-last <integer> (1 - N)` Number of backups to keep.

`--keep-monthly <integer> (1 - N)` Number of monthly backups to keep.

`--keep-weekly <integer> (1 - N)` Number of weekly backups to keep.

`--keep-yearly <integer> (1 - N)` Number of yearly backups to keep.

`--prune-schedule <string>` Run prune job at specified schedule.

`proxmox-backup-manager datastore list [OPTIONS]`

Datastore list.

Optional parameters:

`--output-format <string>` Output format.

`proxmox-backup-manager datastore remove <name> [OPTIONS]`

Remove a datastore configuration.

`<name> <string>` Datastore name.

Optional parameters:

`--digest <string>` Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

`proxmox-backup-manager datastore show <name> [OPTIONS]`

Show datastore configuration

`<name> <string>` Datastore name.

Optional parameters:

`--output-format <string>` Output format.

`proxmox-backup-manager datastore update <name> [OPTIONS]`

Update datastore config.

<name> <string> Datastore name.

Optional parameters:

--comment <string> Comment (single line).

--delete <array> List of properties to delete.

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--gc-schedule <string> Run garbage collection job at specified schedule.

--keep-daily <integer> (1 - N) Number of daily backups to keep.

--keep-hourly <integer> (1 - N) Number of hourly backups to keep.

--keep-last <integer> (1 - N) Number of backups to keep.

--keep-monthly <integer> (1 - N) Number of monthly backups to keep.

--keep-weekly <integer> (1 - N) Number of weekly backups to keep.

--keep-yearly <integer> (1 - N) Number of yearly backups to keep.

--prune-schedule <string> Run prune job at specified schedule.

`proxmox-backup-manager disk fs create <name> --disk <string> [OPTIONS]`

Create a Filesystem on an unused disk. Will be mounted under '/mnt/datastore/<name>'.

<name> <string> Datastore name.

--disk <string> Block device name (/sys/block/<name>).

Optional parameters:

--add-datastore <boolean> Configure a datastore using the directory.

--filesystem <string>

`proxmox-backup-manager disk fs list [OPTIONS]`

List systemd datastore mount units.

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager disk initialize <disk> [OPTIONS]`

Initialize empty Disk with GPT

<disk> <string> Block device name (/sys/block/<name>).

Optional parameters:

--uuid <string> UUID for the GPT table.

`proxmox-backup-manager disk list [OPTIONS]`

Local disk list.

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager disk smart-attributes <disk> [OPTIONS]`

Show SMART attributes.

<disk> <string> Block device name (/sys/block/<name>).

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager disk zpool create <name> --devices <string> --raidlevel <string> [OPTIONS]`

create a zfs pool

<name> <string> Datastore name.

--devices <string> A list of disk names, comma separated.

--raidlevel <string> The ZFS RAID level to use.

Optional parameters:

--add-datastore <boolean> Configure a datastore using the zpool.

--ashift <integer> (9 - 16) (default=12) Pool sector size exponent.

--compression <string> (default=On) The ZFS compression algorithm to use.

`proxmox-backup-manager disk zpool list [OPTIONS]`

Local zfs pools.

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager dns get [OPTIONS]`

Read DNS settings

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager dns set [OPTIONS]`

Update DNS settings.

Optional parameters:

--delete <array> List of properties to delete.

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--dns1 <string> First name server IP address.

--dns2 <string> Second name server IP address.

--dns3 <string> Third name server IP address.

--search <string> Search domain for host-name lookup.

`proxmox-backup-manager garbage-collection start <store> [OPTIONS]`

Start garbage collection for a specific datastore.

<store> <string> Datastore name.

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager garbage-collection status <store> [OPTIONS]`

Show garbage collection status for a specific datastore.

<store> <string> Datastore name.

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager help [{<command>}] [OPTIONS]`

Get help about specified command (or sub-command).

<command> <array> Command. This may be a list in order to spefify nested sub-commands.

Optional parameters:

--verbose <boolean> Verbose help.

`proxmox-backup-manager network changes`

Show pending configuration changes (diff)

`proxmox-backup-manager network create <iface> [OPTIONS]`

Create network interface configuration.

<iface> <string> Network interface name.

Optional parameters:

--autostart <boolean> Autostart interface.

--bond_mode <string> Linux Bond Mode

--bridge_ports <string> A list of network devices, comma separated.

--bridge_vlan_aware <boolean> Enable bridge vlan support.

--cidr <string> IPv4 address with netmask (CIDR notation).

--cidr6 <string> IPv6 address with netmask (CIDR notation).

--comments <string> Comments (inet, may span multiple lines)

--comments6 <string> Comments (inet5, may span multiple lines)

--gateway <string> IPv4 address.

--gateway6 <string> IPv6 address.

--method <string> Interface configuration method

--method6 <string> Interface configuration method

--mtu <integer> (46 - 65535) (default=1500) Maximum Transmission Unit.

--slaves <string> A list of network devices, comma separated.

--type <string> Network interface type

`proxmox-backup-manager network list [OPTIONS]`

Network device list.

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager network reload`

Reload network configuration (requires ifupdown2).

`proxmox-backup-manager network remove <iface> [OPTIONS]`

Remove network interface configuration.

<iface> <string> Network interface name.

Optional parameters:

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

`proxmox-backup-manager network revert`

Revert network configuration (rm /etc/network/interfaces.new).

`proxmox-backup-manager network update <iface> [OPTIONS]`

Update network interface config.

<iface> <string> Network interface name.

Optional parameters:

--autostart <boolean> Autostart interface.

--bond_mode <string> Linux Bond Mode

--bridge_ports <string> A list of network devices, comma separated.

--bridge_vlan_aware <boolean> Enable bridge vlan support.

--cidr <string> IPv4 address with netmask (CIDR notation).

--cidr6 <string> IPv6 address with netmask (CIDR notation).

--comments <string> Comments (inet, may span multiple lines)

--comments6 <string> Comments (inet5, may span multiple lines)

--delete <array> List of properties to delete.

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--gateway <string> IPv4 address.

--gateway6 <string> IPv6 address.

--method <string> Interface configuration method

--method6 <string> Interface configuration method

--mtu <integer> (46 - 65535) (default=1500) Maximum Transmission Unit.

--slaves <string> A list of network devices, comma separated.

--type <string> Network interface type

`proxmox-backup-manager pull <remote> <remote-store> <local-store> [OPTIONS]`

Sync datastore from another repository

<remote> <string> Remote ID.

<remote-store> <string> Datastore name.

<local-store> <string> Datastore name.

Optional parameters:

--output-format <string> Output format.

--remove-vanished <boolean> (default=true) Delete vanished backups. This remove the local copy if the remote backup was deleted.

`proxmox-backup-manager remote create <name> --host <string> --password <string> --userid <string> [OPTIONS]`

Create new remote.

<name> <string> Remote ID.

--host <string> DNS name or IP address.

--password <string> Password or auth token for remote host.

--userid <string> User ID

Optional parameters:

--comment <string> Comment (single line).

--fingerprint <string> X509 certificate fingerprint (sha256).

`proxmox-backup-manager remote list [OPTIONS]`

List configured remotes.

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager remote remove <name> [OPTIONS]`

Remove a remote from the configuration file.

<name> <string> Remote ID.

Optional parameters:

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

`proxmox-backup-manager remote show <name> [OPTIONS]`

Show remote configuration

<name> <string> Remote ID.

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager remote update <name> [OPTIONS]`

Update remote configuration.

<name> <string> Remote ID.

Optional parameters:

--comment <string> Comment (single line).

--delete <array> List of properties to delete.

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--fingerprint <string> X509 certificate fingerprint (sha256).

--host <string> DNS name or IP address.

--password <string> Password or auth token for remote host.

--userid <string> User ID

`proxmox-backup-manager sync-job create <id> --remote <string> --remote-store <string> --store <string> [OPTIONS]`

Create a new sync job.

<id> <string> Job ID.

--remote <string> Remote ID.

--remote-store <string> Datastore name.

--store <string> Datastore name.

Optional parameters:

--comment <string> Comment (single line).

--remove-vanished <boolean> (default=true) Delete vanished backups. This remove the local copy if the remote backup was deleted.

--schedule <string> Run sync job at specified schedule.

`proxmox-backup-manager sync-job list [OPTIONS]`

Sync job list.

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager sync-job remove <id> [OPTIONS]`

Remove a sync job configuration

<id> <string> Job ID.

Optional parameters:

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

`proxmox-backup-manager sync-job show <id> [OPTIONS]`

Show sync job configuration

<id> <string> Job ID.

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager sync-job update <id> [OPTIONS]`

Update sync job config.

<id> <string> Job ID.

Optional parameters:

--comment <string> Comment (single line).

--delete <array> List of properties to delete.

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--remote <string> Remote ID.

--remote-store <string> Datastore name.

--remove-vanished <boolean> (default=true) Delete vanished backups. This remove the local copy if the remote backup was deleted.

--schedule <string> Run sync job at specified schedule.

--store <string> Datastore name.

`proxmox-backup-manager task list [OPTIONS]`

List running server tasks.

Optional parameters:

--all <boolean> Also list stopped tasks.

--limit <integer> (1 - 1000) (default=50) The maximal number of tasks to list.

--output-format <string> Output format.

`proxmox-backup-manager task log <upid>`

Display the task log.

<upid> <string> Unique Process/Task ID.

`proxmox-backup-manager task stop <upid>`

Try to stop a specific task.

<upid> <string> Unique Process/Task ID.

`proxmox-backup-manager user create <userid> [OPTIONS]`

Create new user.

<userid> <string> User ID

Optional parameters:

--comment <string> Comment (single line).

--email <string> E-Mail Address.

--enable <boolean> (default=true) Enable the account (default). You can set this to '0' to disable the account.

--expire <integer> (0 - N) (default=0) Account expiration date (seconds since epoch). '0' means no expiration date.

--firstname <string> First name.

--lastname <string> Last name.

--password <string> User Password.

`proxmox-backup-manager user list [OPTIONS]`

List configured users.

Optional parameters:

--output-format <string> Output format.

`proxmox-backup-manager user remove <userid> [OPTIONS]`

Remove a user from the configuration file.

<userid> <string> User ID

Optional parameters:

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

`proxmox-backup-manager user update <userid> [OPTIONS]`

Update user configuration.

<userid> <string> User ID

Optional parameters:

--comment <string> Comment (single line).

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--email <string> E-Mail Address.

--enable <boolean> (default=true) Enable the account (default). You can set this to '0' to disable the account.

--expire <integer> (0 - N) (default=0) Account expiration date (seconds since epoch). '0' means no expiration date.

--firstname <string> First name.

--lastname <string> Last name.

--password <string> User Password.

`proxmox-backup-manager verify <store> [OPTIONS]`

Verify backups

<store> <string> Datastore name.

Optional parameters:

--output-format <string> Output format.

A.3 pxar

`pxar create <archive> <source> [OPTIONS]`

Create a new .pxar archive.

<archive> <string> Archive name.

<source> <string> Source directory.

Optional parameters:

--all-file-systems <boolean> (default=false) Include mounted sudirs.

--entries-max <integer> (0 - 9223372036854775807) (default=1048576) Max number of entries loaded at once into memory

--exclude <array> List of paths or pattern matching files to exclude.

--no-acls <boolean> (default=false) Ignore access control list entries.

--no-device-nodes <boolean> (default=false) Ignore device nodes.

--no-fcaps <boolean> (default=false) Ignore file capabilities.

--no-fifos <boolean> (default=false) Ignore fifos.

--no-sockets <boolean> (default=false) Ignore sockets.

--no-xattrs <boolean> (default=false) Ignore extended file attributes.

--verbose <boolean> (default=false) Verbose output.

`pxar extract <archive> [<target>] [OPTIONS]`

Extract an archive.

<archive> <string> Archive name.

<target> <string> Target directory

Optional parameters:

--allow-existing-dirs <boolean> (default=false) Allows directories to already exist on restore.

--files-from <string> File containing match pattern for files to restore.

--no-acls <boolean> (default=false) Ignore access control list entries.

--no-device-nodes <boolean> (default=false) Ignore device nodes.

--no-fcaps <boolean> (default=false) Ignore file capabilities.

--no-fifos <boolean> (default=false) Ignore fifos.

--no-sockets <boolean> (default=false) Ignore sockets.

--no-xattrs <boolean> (default=false) Ignore extended file attributes.

--pattern <array> List of paths or pattern matching files to restore

--strict <boolean> (default=false) Stop on errors. Otherwise most errors will simply warn.

--verbose <boolean> (default=false) Verbose output.

`pxar help [{<command>}] [OPTIONS]`

Get help about specified command (or sub-command).

<command> <array> Command. This may be a list in order to spefify nested sub-commands.

Optional parameters:

--verbose <boolean> Verbose help.

`pxar list <archive> [OPTIONS]`

List the contents of an archive.

<archive> <string> Archive name.

Optional parameters:

--verbose <boolean> (default=false) Verbose output.

`pxar mount <archive> <mountpoint> [OPTIONS]`

Mount the archive to the provided mountpoint via FUSE.

<archive> <string> Archive name.

<mountpoint> <string> Mountpoint for the file system.

Optional parameters:

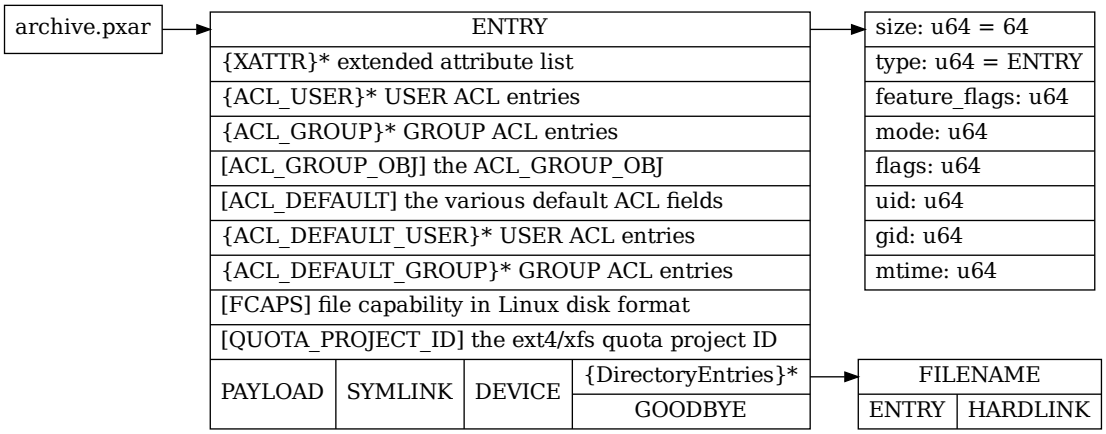
--verbose <boolean> (default=false) Verbose output, running in the foreground (for debugging).

APPENDIX

B

FILE FORMATS

B.1 Proxmox File Archive Format (`.pxar`)



APPENDIX

C

BACKUP PROTOCOL

Todo: add introduction to HTTP2 based backup protocols

C.1 Backup Protocol API

Todo: describe backup writer protocol

GET .

Directory index.

Returns: **<null>**

UPLOAD blob

Upload binary blob file.

Required properties:

`encoded-size <integer> (12 - 16777260)` Encoded blob size.

`file-name <string>` Backup archive name.

Returns: **<null>**

UPLOAD dynamic_chunk

Upload a new chunk.

Required properties:

`digest <string>` Chunk digest (SHA256).

`encoded-size <integer> (13 - 16777260)` Encoded chunk size.

`size <integer> (1 - 16777216)` Chunk size.

`wid <integer> (1 - 256)` Dynamic writer ID.

Returns: **<null>**

POST dynamic_close

Close dynamic index writer.

Required properties:

chunk-count <integer> (1 - N) Chunk count. This is used to verify that the server got all chunks.

csum <string> Digest list checksum.

size <integer> (1 - N) File size. This is used to verify that the server got all data.

wid <integer> (1 - 256) Dynamic writer ID.

Returns: **<null>**

POST dynamic_index

Create dynamic chunk index file.

Required properties:

archive-name <string> Backup archive name.

Returns: **<null>**

PUT dynamic_index

Append chunk to dynamic index writer.

Required properties:

digest-list <array> Chunk digest list.

offset-list <array> Chunk offset list.

wid <integer> (1 - 256) Dynamic writer ID.

Returns: **<null>**

POST finish

Mark backup as finished.

Returns: **<null>**

UPLOAD fixed_chunk

Upload a new chunk.

Required properties:

digest <string> Chunk digest (SHA256).

encoded-size <integer> (13 - 16777260) Encoded chunk size.

size <integer> (1 - 16777216) Chunk size.

wid <integer> (1 - 256) Fixed writer ID.

Returns: **<null>**

POST fixed_close

Close fixed index writer.

Required properties:

chunk-count <integer> (0 - N) Chunk count. This is used to verify that the server got all chunks. Ignored for incremental backups.

csum <string> Digest list checksum.

size <integer> (0 - N) File size. This is used to verify that the server got all data. Ignored for incremental backups.

wid <integer> (1 - 256) Fixed writer ID.

Returns: **<null>**

POST fixed_index

Create fixed chunk index file.

Required properties:

archive-name <string> Backup archive name.

size <integer> (1 - N) File size.

Optional properties:

reuse-csum <string> If set, compare last backup's csum and reuse index for incremental backup if it matches.

Returns: **<null>**

PUT fixed_index

Append chunk to fixed index writer.

Required properties:

digest-list <array> Chunk digest list.

offset-list <array> Chunk offset list.

wid <integer> (1 - 256) Fixed writer ID.

Returns: **<null>**

DOWNLOAD previous

Download archive from previous backup.

Required properties:

archive-name <string> Backup archive name.

Returns: **<null>**

UPLOAD speedtest

Test upload speed.

Returns: **<null>**

C.2 Reader Protocol API

Todo: describe backup reader protocol

GET .

Directory index.

Returns: **<null>**

UPLOAD blob

Upload binary blob file.

Required properties:

`encoded-size <integer> (12 - 16777260)` Encoded blob size.

`file-name <string>` Backup archive name.

Returns: **<null>**

UPLOAD dynamic_chunk

Upload a new chunk.

Required properties:

`digest <string>` Chunk digest (SHA256).

`encoded-size <integer> (13 - 16777260)` Encoded chunk size.

`size <integer> (1 - 16777216)` Chunk size.

`wid <integer> (1 - 256)` Dynamic writer ID.

Returns: **<null>**

POST dynamic_close

Close dynamic index writer.

Required properties:

`chunk-count <integer> (1 - N)` Chunk count. This is used to verify that the server got all chunks.

`csum <string>` Digest list checksum.

`size <integer> (1 - N)` File size. This is used to verify that the server got all data.

`wid <integer> (1 - 256)` Dynamic writer ID.

Returns: **<null>**

POST dynamic_index

Create dynamic chunk index file.

Required properties:

`archive-name <string>` Backup archive name.

Returns: **<null>**

PUT dynamic_index

Append chunk to dynamic index writer.

Required properties:

digest-list <array> Chunk digest list.

offset-list <array> Chunk offset list.

wid <integer> (1 - 256) Dynamic writer ID.

Returns: **<null>**

POST finish

Mark backup as finished.

Returns: **<null>**

UPLOAD fixed_chunk

Upload a new chunk.

Required properties:

digest <string> Chunk digest (SHA256).

encoded-size <integer> (13 - 16777260) Encoded chunk size.

size <integer> (1 - 16777216) Chunk size.

wid <integer> (1 - 256) Fixed writer ID.

Returns: **<null>**

POST fixed_close

Close fixed index writer.

Required properties:

chunk-count <integer> (0 - N) Chunk count. This is used to verify that the server got all chunks. Ignored for incremental backups.

csum <string> Digest list checksum.

size <integer> (0 - N) File size. This is used to verify that the server got all data. Ignored for incremental backups.

wid <integer> (1 - 256) Fixed writer ID.

Returns: **<null>**

POST fixed_index

Create fixed chunk index file.

Required properties:

archive-name <string> Backup archive name.

size <integer> (1 - N) File size.

Optional properties:

`reuse-csum <string>` If set, compare last backup's csum and reuse index for incremental backup if it matches.

Returns: **<null>**

PUT fixed_index

Append chunk to fixed index writer.

Required properties:

`digest-list <array>` Chunk digest list.

`offset-list <array>` Chunk offset list.

`wid <integer> (1 - 256)` Fixed writer ID.

Returns: **<null>**

DOWNLOAD previous

Download archive from previous backup.

Required properties:

`archive-name <string>` Backup archive name.

Returns: **<null>**

UPLOAD speedtest

Test upload speed.

Returns: **<null>**

APPENDIX

D

GLOSSARY

Virtual machine A virtual machine is a program that can execute an entire operating system inside an emulated hardware environment.

Container A container is an isolated user space. Programs run directly on the host's kernel, but with limited access to the host resources.

Datastore A place to store backups. A directory which contains the backup data. The current implementation is file-system based.

Rust Rust is a new, fast and memory-efficient system programming language. It has no runtime or garbage collector. Rust's rich type system and ownership model guarantee memory-safety and thread-safety. I can eliminate many classes of bugs at compile-time.

Sphinx Is a tool that makes it easy to create intelligent and beautiful documentation. It was originally created for the documentation of the Python programming language. It has excellent facilities for the documentation of software projects in a range of languages.

reStructuredText Is an easy-to-read, what-you-see-is-what-you-get plaintext markup syntax and parser system.

FUSE Filesystem in Userspace (FUSE) defines an interface which makes it possible to implement a filesystem in userspace as opposed to implementing it in the kernel. The fuse kernel driver handles filesystem requests and sends them to a userspace application.

Remote A remote Proxmox Backup Server installation and credentials for a user on it. You can pull datastores from a remote to a local datastore in order to have redundant backups.

Schedule Certain tasks, for example pruning and garbage collection, need to be performed on a regular basis. Proxmox Backup Server uses a subset of the systemd Time and Date Specification. The subset currently supports time of day specifications and weekdays, in addition to the shorthand expressions 'minutely', 'hourly', 'daily'. There is no support for specifying timezones, the tasks are run in the timezone configured on the server.

APPENDIX

E

GNU FREE DOCUMENTATION LICENSE

Version 1.3, 3 November 2008

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

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 noncommercially. 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 in order 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.

The "publisher" means any person or entity that distributes copies of the Document to the public.

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 noncommercially, 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 under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

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/licenses/.

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. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those

works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

- genindex

INDEX