3. Getting the code

Source file: doc/eng-us/install/03-getting-the-code.md Last changed 2026-11-07
https://doc.exponential.earth/learninminutes/exphelp/viewer/install/03-getting-the-code.md

This chapter puts the Exponential code on your machine. There are three ways: a `git clone` followed by `composer install`, with every se7enxweb package arriving as a Git clone with its full history (**the recommended way for anyone who builds websites**), `composer create-project` (the quickest way to a site you only run) and `composer require` into an empty directory (when the installation should be a project of your own). All three end with the same files. The chapter explains which version each way gives you and how to choose one, how the extensions arrive as Composer packages, how the Exponential Velocity package arrives, what is different on PHP 8.0 to 8.3, and closes with a tour of every top-level entry of the installation root and the permissions to set before you install.

> **Warning**
>
> `.htaccess` **slows down every request.** When `.htaccess` files are enabled (`AllowOverride` anything but `None`), let alone used, Apache looks for them and reads and evaluates them on every request, in every directory of the path, and page loading times go up drastically (measured). The only real fix is to switch the feature off for the virtual host: `AllowOverride None` in the HTTPS virtual host, and the rules in the virtual host's own configuration as mod_rewrite rules. That is far faster in production speed tests and well worth the time it takes to optimize a production configuration. Switch over right away: copy the rules of the example HTTPS virtual host into your virtual host (with the static cache, also `staticcache_offload.conf`), set `AllowOverride None`, run `apachectl configtest`, reload Apache and test the site. The same rules for other servers: nginx, LiteSpeed, Caddy, IIS, Velocity (built in); where virtual host rules go in a control panel: Plesk, cPanel, DirectAdmin, ISPConfig. **On LiteSpeed or OpenLiteSpeed,** the server's own page cache can also answer anonymous visitors and crawlers without PHP: Page caching with LiteSpeed and OpenLiteSpeed.

Contents · Previous: 2. Requirements · Next: 4. Choosing an install method

3.1 Versions, tags and branches

Exponential is published as the Composer package `se7enxweb/exponential` from the Git repository github.com/se7enxweb/exponential.

What	Name	Gives you
Release tags	`v6.0.0` to `v6.0.14` (the 6.0 line); older lines use tags without v, such as `4.4.0`	an exact, permanent release
Default branch	`main`	the current development of the 6.0 line, today 6.0.15
Release branches	`6.0.7` to `6.0.14`	the state of each release line

The current line, 6.0.15, follows the tag `v6.0.14`; the `main` branch identifies itself as `6.0.15stable` (`lib/version.php`) and carries the changes described in Changelog 6.0.15. To

see what is published now:

Shell

```
git ls-remote --tags https://github.com/se7enxweb/exponential.git 'v6*'
```

or, inside a clone, after `git fetch --tags`:

Shell

```
git tag -l 'v*' --sort=version:refname | tail -5
```

which prints the five newest tags, the newest last:

```
v6.0.10
v6.0.11
v6.0.12
v6.0.13
v6.0.14
```

Always sort by version (`--sort=version:refname` or `sort -V`), never by name: by name `v6.0.10` sorts before `v6.0.8`, and a script that takes the last line of a name sort picks the wrong release. The project's releases are listed with their notes on GitHub Releases.

A release tag never changes. Once published, a tag and the package version Packagist built from it are permanent; a fix always arrives as the next version. You can pin a tag and rely on getting the same files every time, which is what makes a pinned version the right choice for production.

Which version needs which PHP. Read this before you choose, because Composer refuses a version whose PHP requirement your PHP does not meet:

Version	`"php"` **in its** `composer.json`	On PHP 8.0
`v6.0.0` to `v6.0.7`	`^7.4 || ^8.1`	not installable
`v6.0.8` to `v6.0.14`	`^8.1` and later 8.x	not installable
6.0.15 (`~6.0.15`; `main`, `dev-main` for Composer)	`^8.0` and later 8.x	installable

None of the tags up to `v6.0.14` accepts PHP 8.0: on PHP 8.0, take the 6.0.15 line, with `~6.0.15` (the tag `v6.0.15` once it is published) or from `main`, which carries it.

`composer.json` gives `main` the branch alias `6.0.x-dev` (`"extra": { "branch-alias": { "dev-main": "6.0.x-dev" } }`). Installed from `main`, the root package is then version `6.0.x-dev` as well as `dev-main`, so a package that requires `se7enxweb/exponential` with a version constraint such as `^6.0.14` (expqueue does) resolves against it. Without the alias `dev-main` satisfies no version constraint and such an install stops with "could not be resolved".

3.2 Three ways to get the code

Way	Gives you	Best for
A. git clone and composer install	the kernel and every se7enxweb extension as Git clones with their full history	**building websites**: development machines, staging, and every site whose code you change (recommended)
B. composer create-project	release archives without history (or clones, with two options)	a site you only run, a quick evaluation

Way	Gives you	Best for
C. composer require into an empty directory	the kernel as a package of your own project	an installation managed as a project of your own

A. git clone and composer install (recommended)

The recommended way for anyone who builds websites with Exponential. With Git clones you see what changed (`git status`, `git diff`), why a line is there (`git log`, `git blame`), which change broke something (`git bisect`), can use a fix before it is released, keep your own changes on a branch that updates never overwrite, and send fixes back as pull requests. Develop with Git explains each of these step by step.

First tell Composer, once per user account, to install every `se7enxweb/*` package as a Git clone (the extensions, and since 6.0.15 the expcomponents libraries as well) and everything else (other libraries) as a release archive:

Shell

```
composer config --global 'preferred-install.se7enxweb/*' source
```

Then clone the kernel, start a branch of your own and install:

Shell

```
git clone https://github.com/se7enxweb/exponential.git
cd exponential
git switch -c my-site               # or: git switch -c my-site v6.0.14, to build on a release in
stead of main
composer install
```

The clone holds the kernel with its whole history (more than 27,000 commits, about 370 MB), the kernel-shipped extensions (`ezjscore`, `ezoe`, `ezformtoken`, `expservices`) and the designs. `composer install` adds `vendor/` and clones every extension `composer.json` requires into `extension/<name>`, each a Git repository of its own (67 on `main`; the first install takes a few minutes). Those extensions are not part of the kernel's repository, so `git status` lists them as untracked; that is expected. Each sits on the release tag Composer chose, so `git -C extension/<name> status` says `HEAD detached at <tag>`: normal, and a branch is one `git switch -c` away.

Check that the extensions are clones:

Shell

```
composer status -v                  # "No local changes"
ls -d extension/*/.git | wc -l      # the number of extensions installed as clones
git -C extension/eztags log --oneline -3
```

Without the `preferred-install` setting the extensions arrive as archives; `composer reinstall 'se7enxweb/*'` turns them into clones later (Develop with Git). To keep the setting with the project rather than your account, run the `composer config` command without `--global` inside the clone; it is then written into `composer.json` on your branch.

To update later, on your branch:

Shell

```
git fetch origin --tags
git merge origin/main              # or the release you move to: git merge v6.0.<n>
composer update
```

`composer update`, not `composer install`: the kernel's `composer.json` changed, and the lock file (your own, it is not part of the repository) has to follow it. `composer update se7enxweb/<package>` updates one extension. Always Git first, then Composer. Update with Git has the routine on the command line and in the administration (Git Manager, then Updates and packages), for every schedule, from hourly to only at each release, and an automatic update with cron or a systemd timer. Composer refuses to update a clone with uncommitted changes or with commits that exist only on your machine, so your work is never overwritten; When Composer refuses lists the messages and what to do. Moving the code is only half of an upgrade: the database update files of chapter 11 are the other half.

B. composer create-project

The quickest way to a site you only run. Composer downloads the package, makes the directory your project and installs every dependency:

Shell

```
cd /var/www
composer create-project se7enxweb/exponential exponential
cd exponential
```

Without a version, Composer takes the newest stable release. To pin one, or to follow the development branch:

Shell

```
composer create-project se7enxweb/exponential:v6.0.14 exponential
composer create-project se7enxweb/exponential:~6.0.15 exponential
composer create-project se7enxweb/exponential:dev-main exponential
```

`create-project` treats the downloaded `composer.json` as the root package, so its `config.allow-plugins` (which allows `se7enxweb/exponential-legacy-installer`) and its `post-install-cmd` script apply: the class maps are generated at the end (`php bin/php/ezpgenerateautoloads.php`). On a production server add `--no-dev`, which leaves out the test tools.

By default every package, the kernel included, arrives as a release archive without history. To get clones from `create-project` as well, give both options:

Shell

```
composer create-project se7enxweb/exponential exponential --prefer-source --keep-vcs
```

Without `--keep-vcs` Composer deletes the kernel's `.git` folder at the end. A site installed from archives can be turned into clones later, without reinstalling it: Develop with Git.

C. composer require into an empty directory

When the installation should be a project of your own, with your own `composer.json`:

Shell

```
mkdir exponential
cd exponential
composer require se7enxweb/exponential
```

The package has the type `ezpublish-legacy`. The installer plugin (`LegacyKernelInstaller`) installs a package of that type into the project root (`.`), so the kernel files land in the directory you are in, not in `vendor/`. In a Symfony project (one whose `composer.json` sets `extra.symfony-app-dir`) it goes to `ezpublish_legacy/` instead; a project can name another directory with `extra.ezpublish-legacy-dir`.

Two things differ from ways A and B:

- Your new `composer.json` does not yet allow the installer plugin. Composer 2.2 and later asks whether to trust `se7enxweb/exponential-legacy-installer`; answer yes, or add it beforehand:

  Shell

  ```
  composer config allow-plugins.se7enxweb/exponential-legacy-installer true
  ```

  If you answer no, the kernel lands in `vendor/se7enxweb/exponential/` and the extensions in `vendor/` as well, where nothing finds them.

- Composer runs only the scripts of the root package, which is now yours. Generate the class maps yourself after each install or update:

  Shell

  ```
  php bin/php/ezpgenerateautoloads.php
  ```

Check the result

Whichever way you took:

Shell

```
php bin/php/console --version
ls vendor/autoload.php
```

The first line of the console's output names the version:

```
console (Exponential) 6.0.15stable
```

If `vendor/autoload.php` is missing, Composer did not finish; run `composer install` again and read its output from the first error, not the last line. The usual causes are a missing PHP extension (Composer names the `ext-*` it wants), a PHP that is too old for the version you asked for (3.1), or a plugin that was not allowed.

3.3 Installing on PHP 8.0 to 8.3

On the current line (`main`, 6.0.15) nothing special is needed: a plain `composer install` works on every PHP from 8.0 to 8.5. The development requirement `"phpunit/phpunit": "^9.6 || ^10.5 || ^11.5 || ^12.0 || ^13.4"` lets Composer pick the PHPUnit that fits the PHP in use (9.6 on 8.0, 10.5 on 8.1, 11.5 on 8.2, 12 on 8.3, 13.4 on 8.4 and 8.5). That range matters even with `--no-dev`: the repository ships no `composer.lock`, and without a lock file Composer

resolves `require-dev` too, although it does not install it. The project's own PHP 8.0 check (the job `php80` of `.github/workflows/quality.yml`) installs exactly as a site does:

Shell

```
composer install --no-dev --prefer-dist
```

The test suite itself uses PHPUnit 10 features, so it runs on PHP 8.1 and later; on PHP 8.0 the range only lets the install resolve.

The release tags `v6.0.13` and `v6.0.14` are different. They ask for `"phpunit/phpunit": "13.0.0"`, which needs PHP 8.4.1, so on PHP 8.1 to 8.3 a plain install of those tags stops with a PHP version conflict on PHPUnit. Remove the two development requirements first, then install what a site needs:

Shell

```
composer create-project --no-install se7enxweb/exponential:v6.0.14 exponential
cd exponential
composer remove --dev --no-update --no-interaction phpunit/phpunit zetacomponents/php-generator
composer install --no-dev --prefer-dist
```

`composer remove --no-update` changes only `composer.json` and installs nothing; the second command installs. In a Git clone of those tags the last two commands are enough, and `git checkout composer.json` puts the file back afterwards.

On a production server, `--no-dev` is right on every PHP version: the test tools are not needed to run a site.

3.4 The Velocity package

As of 6.0.15 Exponential Velocity, the Internet / Web Application Server and the recommended engine for every stage, is part of Exponential. It is the package `se7enxweb/exponential-velocity`, and the root `composer.json` **requires** it (it was only suggested before 6.0.15):

JSON

```
"se7enxweb/exponential-velocity": "~0.0.4.57"
```

Velocity 0.0.4.54 runs on PHP 8.0 and later (`"php": ">=8.0"`) and needs the `sockets` extension (`"ext-sockets": "*"`), so the same PHP that runs the kernel runs the server, the stock PHP 8.0 of Red Hat Enterprise Linux 9 included. A plain `composer install` installs it; nothing more is needed. Until 0.0.4.54 the engine needed PHP 8.1, and `composer.json` only suggested it; an installation made in that time adds it with:

Shell

```
composer require se7enxweb/exponential-velocity:~0.0.4.57
```

`~0.0.4.57` accepts every later `0.0.4.x` release (`>=0.0.4.57 <0.0.5.0`), so Composer installs the newest of them. The last position of a Velocity version counts on past 9 (`0.0.4.9`, `0.0.4.10`, ...), so a version sort, not a name sort, finds the newest.

The package lands in `vendor/se7enxweb/exponential-velocity/` (its type is `project`, so it is not moved into `extension/`). The console finds it there: `php bin/php/console exp:velocity status` reports the engine, and `exp:velocity start` starts it. Before the package is installed,

the `qbix` engine cannot start and says `velocity: no server script at ...`; the `php` engine (PHP's built-in server) works without it:

Shell

```
php bin/php/console exp:velocity start --engine=php
```

Velocity is also published as operating system packages (`.deb`, `.rpm`) and as a standalone binary; see Velocity packages and binaries. Before you upgrade it, read Velocity engine upgrade notes.

3.5 Extensions are Composer packages

Every extension except the four kernel-shipped ones is a Composer package of type `ezpublish-legacy-extension`. The installer plugin `se7enxweb/exponential-legacy-installer` (required by `composer.json`) places such a package in `extension/<name>`:

The package says	It is installed in
`"extra": { "ezpublish-legacy-extension-name": "eztags" }` (package `se7enxweb/eztags`)	`extension/eztags`
`"extra": { "ezpublish-legacy-extension-name": "explayouts_ui" }` (package `se7enxweb/explayouts-ui`)	`extension/explayouts_ui`
no `ezpublish-legacy-extension-name` (package `vendor/name`)	`extension/name`

So `composer.json` decides which extensions an installation has, and their versions:

JSON

```
"se7enxweb/eztags": "~2.4.11",
"se7enxweb/cjw_newsletter": "~4.2.0",
"se7enxweb/explayouts": "~1.4.15",
```

The tilde constraint floats the last position given: `~2.4.11` means `>=2.4.11 <2.5.0`. To add an extension, require its package, then activate it and regenerate the class maps:

Shell

```
composer require se7enxweb/ezfind
php bin/php/ezpgenerateautoloads.php -e
```

Being on disk does not make an extension active: it must be listed in `[ExtensionSettings] ActiveExtensions[]`. The installers activate the extensions of the site package; others you activate in the admin (**Setup > Extensions**) or in `settings/override/site.ini.append.php`:

INI

```
[ExtensionSettings]
ActiveExtensions[]=ezfind
```

Then clear the INI cache (`php bin/php/ezcache.php --clear-tag=ini`), and restart Velocity if it serves the site.

Two packages of `composer.json` arrive inactive on purpose and are switched on when you want them: the job queue `se7enxweb/expqueue` (`extension/expqueue`, expqueue) and Exponential Intelligence, the writing assistant of the administration, `se7enxweb/expai` (

`extension/expai`, expai). Both need tables of their own and a cron line; the assistant also a provider key in `settings/override/expai.ini.append.php`. Section 10.13 has the steps.

`composer.json`'s `suggest` block lists further extensions with a sentence each: `ezsi`, `ezscriptmonitor`, `ezfind`, `ezauthorize`, `bcurlaliaswithdash`, `sevenx_valkey`, `sevenx_valkey_cache`, `ezownerchange`, `adminaid`, `ezpm` and `hcaptcha` among them. Every extension has its own repository under github.com/se7enxweb and carries its version in `ezinfo.php` and `extension.xml`; the admin's extension list shows it.

> **Do not run `composer update` casually on a live site.** It moves every package to the newest version its constraint allows. Update one package deliberately (`composer update se7enxweb/<package>`) and read its release notes first. Clones that Composer made itself (way A) are safe: Composer refuses to update one with uncommitted changes or unpushed commits. A clone made **by hand** (`git clone` into `extension/<name>`) in place of a package Composer installed as an archive is not: Composer still treats the folder as an archive and replaces it, history and commits included. To turn an archive into a clone, let Composer do it with `composer reinstall <package>` after setting `preferred-install` (way A).

More: Extensions guide, Extension list and downloads, Default extension distribution, Extension loading order, Additional extension directories.

3.6 A tour of the installation root

The entries a fresh installation has, in alphabetical order. "Tracked" means the file or directory is part of the Git repository; the others are created by Composer or at run time.

Directories

Directory	Tracked	What it holds
`autoload/`	yes	`ezp_kernel.php`, the generated class map of the kernel. Regenerated with `php bin/php/ezpgenerateautoloads.php -k`.
`benchmarks/`	yes	Micro-benchmarks of kernel parts (templates, hashing).
`bin/`	yes	Command line tools: `bin/php/` (the PHP scripts, each also a console command `exp:<name>`, and the console itself, `bin/php/console`), `bin/shell/` (shell scripts, `shell:<name>`), `bin/mongodb/` (MongoDB index scripts), `bin/startup/`, `bin/linux/`, `bin/phpcs/`, `bin/awk/`.
`cronjobs/`	yes	The cronjob parts `runcronjobs.php` runs (`basket_cleanup.php`, `linkcheck.php`, `audit.php`, ...).
`design/`	yes	The designs: `standard` and `base` (fallbacks), `admin`, `admin3`, `admin4`, `admin4l`, `editor`, and the examples `mysite` and `plain`. Theme extensions add more.
`doc/`	yes	The documentation, this book included.
`extension/`	partly	Extensions. The repository ships `ezjscore`, `ezoe`, `ezformtoken` and `expservices`; Composer adds the others.
`kernel/`	yes	The kernel: content model, modules, datatypes, the setup wizard (`kernel/setup/`), the console commands' classes (`kernel/private/classes/commands/`).

Directory	Tracked	What it holds
`lib/`	yes	The general libraries: database drivers (`lib/ezdb/`), templates (`lib/eztemplate/`), files, locales, images, sessions; `lib/version.php` holds the version.
`schemas/`	yes	XML schemas (the translation file schema).
`settings/`	yes	The shipped `.ini` files. `settings/siteaccess/<name>/` and `settings/override/` hold your installation's settings and are written by the installers.
`share/`	yes	`db_schema.dba` and `db_data.dba` (the database the installer creates), `locale/` and `translations/`, `codepages/` and `transformations/` (character sets), `icons/`, `maintenance.html`, and `filelist.md5`, the file manifest the admin's upgrade check compares against.
`support/`	yes	The sources of `ezlupdate`, the translation extraction tool.
`templates/`	yes	Code templates for the class generator (`.ctpl`).
`tests/`	yes	The PHPUnit test suite (`tests/runtests.php`, `phpunit.xml`).
`update/`	yes	Database update files per engine and version (`update/database/<engine>/`) and data repair scripts (`update/common/scripts/`); see chapter 11.
`var/`	no (only `var/webdav/`)	Everything written at run time: `var/cache/`, `var/log/`, `var/storage/` (uploaded files, packages, the SQLite database in `var/storage/sqlite3/`), `var/autoload/` (the extensions' class maps), `var/vc/` (Velocity's run files and logs). Must be writable by the web server.
`vendor/`	no	The Composer libraries: the expcomponents (`vendor/se7enxweb/expcomponents-*`, the fork of the Zeta Components), the installer plugin, Velocity when installed.

Files

File	What it is
`index.php`	The front controller: every page request goes through it.
`index_rest.php`	The front controller of the REST API (`/api/`).
`index_cluster.php`	Serves files from the database cluster.
`index_treemenu.php`	The fast endpoint of the admin's content tree menu.
`index_soap.php`, `index_webdav.php`	The SOAP and WebDAV entry points.
`soap.php`	Forwards to `index_soap.php`, its name until 6.0.14; removed in a later major version (what to change).
`webdav.php`	Forwards to `index_webdav.php`, its name until 6.0.14; removed in a later major version (what to change).
`runcronjobs.php`	Runs cronjob parts: `php runcronjobs.php --list`.
`autoload.php`	Loads Composer's autoloader and the kernel's; every entry point starts with it.
`ezpm.php`	The package manager on the command line.

File	What it is
`composer.json`	The requirements: PHP, extensions, the expcomponents libraries. `composer.dev.json` holds the development tools of `make devtools` (installed into `.devtools/`); `composer.json.dist` is an older upstream manifest kept for reference.
`.htaccess_root`	Apache rewrite rules; copy to `.htaccess` with `./console exp:setup:htaccess --apply` (or `cp .htaccess_root .htaccess`). `.htaccess_root_static` is the same with the rules that serve the static cache ahead of the application (`exp:setup:htaccess --static --apply`).
`kickstart.ini-dist`	The commented template of `kickstart.ini`; `kickstart.ini--example` is an example.
`config.php-RECOMMENDED`	A documented list of the settings `config.php` can make. Copy it to `config.php` with `./console exp:setup:config --apply` (or `cp config.php-RECOMMENDED config.php`), list the settings with `./console exp:config:list` and set one with `./console exp:config:set <NAME> <value> --apply` instead of uncommenting it in an editor. See Setting up .htaccess and config.php from the console.
`config.env.php.example`	Copy to `config.env.php` to name the machine's environment (`EXP_ENV`); see Settings per environment.
`exponential.cron`, `ezpublish.cron`	Example crontabs for the cronjobs (`crontab exponential.cron` after setting its two variables, `EXPONENTIALROOT` and `PHP`).
`sw.js`, `index.js`	The site's service worker.
`robots.txt`, `favicon.ico`	As usual for a web site.
`phpunit.xml`, `phpcs.xml.dist`, `phpstan.neon.dist`, `phpstan-baseline.neon`, `Makefile`	Development and quality tools.
`LICENSE`, `COPYRIGHT.md`, `README.md`, `CONTRIBUTING.md`, `SECURITY.md`	Licence, copyright, the project README, how to contribute, how to report a vulnerability.

Files that appear after an install and are yours, never to be committed or shared: `settings/override/*` and `settings/siteaccess/*` (database credentials), `kickstart.ini` (the database password), `config.env.php`, and `var/log/initial-admin-password`.

3.7 Permissions after getting the code

The user the web server or Velocity's workers run as must be able to write `design`, `extension`, `settings`, `settings/override`, `settings/siteaccess` and `var` with everything below it (the list the setup wizard checks is `[directory_permissions] CheckList` in `settings/setup.ini`). If you ran Composer as a different user, hand the files over before the install, in this form (the setup wizard prints the exact commands for your machine):

Shell

```
sudo chmod -R ug+rwx design extension settings var
sudo chown -R <web user>:<web group> design extension settings var
```

When the command line (cronjobs, the console, Composer) and the web server run as different users, give both a common group that can write the same files, rather than opening the

directories to everyone. Serving and permissions are covered in depth in chapter 8.

References

In this repository:

- Installation overview, section 3.
- Upgrading: moving an existing site to new code.
- PHP 8.0 support: installing with Composer on PHP 8.0.
- Continuous integration and Quality checks.
- Velocity engines, Velocity packages and binaries, Velocity engine upgrade notes.
- Extensions guide, Extensions, Extension metadata.
- Repairing an installation: when `vendor/` is missing.
- Console.
- Changelog 6.0.15 and the extensions' release notes.
- Code: `composer.json`, `lib/version.php`, `.github/workflows/quality.yml` (job `php80`), `vendor/se7enxweb/exponential-legacy-installer/src/eZ/Publish/Composer/` (`LegacyKernelInstaller`, `LegacyExtensionInstaller`).

External:

- Composer: create-project, install, require, remove, version constraints, allow-plugins, scripts .
- Packagist: se7enxweb/exponential, se7enxweb/exponential-velocity, all se7enxweb packages.
- GitHub: se7enxweb/exponential, releases, se7enxweb/exponential-velocity, all se7enxweb repositories.
- PHPUnit: supported versions.
- Git: git clone, git tag.

Contents · Previous: 2. Requirements · Next: 4. Choosing an install method

> **Become an Exponential Partner**
>
> Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.
>
> Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.