SUSE Linux Package Lifecycles and Strategy

SUSE Linux component package lifecycles define the update and support strategies for software packages across SUSE Linux 16 product lines. By categorizing software into stable, balanced or agile lifecycle models, SUSE balances long-term system stability with timely upstream feature updates and security patches. This framework allows system administrators and developers to anticipate update impacts, maintain application compatibility and plan enterprise deployments across general support and long-term service support phases.

Publication Date: 05 Nov 2026

Contents

1 Types of package lifecycles 3

2 Ansible update strategy 5

3 Apache HTTP server update strategy 6

4 Cockpit update strategy 7

5 Container components update strategy 8

6 Desktop components update strategy 9

7 Go update strategy 10

8 High Availability update strategy 11

9 Java update strategy 13

10 `libpng` update strategy 14

11 `man-pages` update strategy 14

12 MariaDB update strategy 15

13 Network UPS Tools update strategy 15

14 `node.js` update strategy 16

15 Perl update strategy 17

16 PHP update strategy 17

17 PostgreSQL update strategy 18

18 Python update strategy 18

19 Ruby delivery strategy 19

20 Rust update strategy 20

21 Security components update strategy 20

22 `smartmontools` update strategy 23

23 **`sudo`** update strategy 23

24 Time zone update strategy 24

25 Toolchain components update strategy 24

26 `util-linux` update strategy 28

27 Valkey update strategy 29

28 Vim update strategy 30

29 Virtualization components update strategy 30

30 Wireshark update strategy 31

31 Legal Notice 32

A GNU Free Documentation License 32

1 Types of package lifecycles

On SUSE Linux 16, component packages are sorted into different lifecycle categories. This section describes the criteria for the sorting.

When categorizing a package, the impact of its changes on the system is considered. To estimate this impact, the interfaces of the component must first be identified. For a shared library, changes to its API or ABI (application binary interface) may disrupt the system. For a compiler or interpreter, disruptive changes may involve supported languages, command-line options or the performance of the compiled code. By contrast, a minor backward-compatible change has little to no impact.

In general, each package belongs to exactly one category: **stable**, **balanced** or **agile**. However, some technologies may have packages sorted into several categories, for example, Python. The following sections describe the package categories in detail.

1.1 Stable

Stable (also called conservative) packages do not deliver a disruptive change while a customer stays on any 16.x minor release. During the upgrade to another minor version, the package version may change but the newer version does **not** introduce incompatible behavior. Customers expect to have LTS (Long Term Support) on these packages.

The packages belonging to this category can change, but the following criteria apply:

- Changes in functionality are backward compatible: functionality can be added, but not removed.
- Changes to interfaces are backward compatible.
- The default behavior of applications does not change unexpectedly.

Under exceptional circumstances, such as serious security issues, a package can be updated even at the cost of bringing disruptive changes. Alternatively, a version with disruptive changes can be delivered as an alternative to the previous one.

A typical example of a stable package is `util-linux`. Another is `glibc`, which remains backward compatible except for symbols deprecated upstream.

1.2 Balanced

Balanced packages change, driven by upstream evolution and customer demands, but must not cause disruptive changes within one minor release. A few incompatible changes are possible, but they are always documented in the release notes for the respective minor release.

Customers expect a moderate number of changes during the upgrade from a minor release. The transition must be smooth, either by restoring the original behavior or by providing the older version in parallel with the new one.

When change is being introduced, it can be in one of the following ways:

- A single version is provided with the minor release. The new version replaces the previous one and allows a smooth transition. New versions are released only with a new minor release.
- Two versions are provided simultaneously. The new version is introduced in addition to the existing version, which then becomes obsolete with the next minor release.

Versions are supported at least until the end of the LTS phase of the minor release that introduced them.

To help with incorporating changes in a conservative environment, the *tick-tock* model can be used. For example, even-numbered minor releases are *tick* releases and odd-numbered minor releases are *tock* releases. Tock releases can still introduce version updates for balanced packages, but these updates remain fully backward compatible in all relevant aspects.

For ISVs (independent software vendors), a stable runtime environment is critical. Therefore, to avoid breaking third-party applications, the older `.so` version is not deprecated immediately when SUSE provides the corresponding `-devel` package. For example, for a package called `foo`, there are packages: `libfoo-0_1`, `foo-devel` and `foo-utils`. If the package is updated and the shared library version changes to `libfoo-0_2`, `libfoo-0_1` is not removed.

Typical examples of balanced packages include the following:

- `systemd`: Changes are backward compatible. Incompatible changes are documented in the release notes.
- kernel: Updated with each minor release.
- MariaDB: A new minor version with each minor release.
- PostgreSQL: Upstream releases roughly annually. The new version is introduced with a minor release or as part of a maintenance update for an older minor release.
- System Python: The primary system interpreter and its associated modules remain stable and supported throughout the product lifecycle.

1.3 Agile

Agile packages are kept up to date with upstream, even if this brings incompatible changes to the system.

Updates to these packages are done in two ways:

- With the release of a new package version, one or more older versions remain supported for customers who cannot switch easily to the new version. There is a sliding window in which different versions are supported concurrently (for example, for version N and version N - 1). These concurrent versions are supported for a certain period that can differ from the lifetime of the minor release.
- The package is updated without support for the older version.

All new package versions are released simultaneously to all minor releases under General Support and, generally, also in LTS.

Typical examples of agile packages include the following:

- Go: A new version roughly every six months, with versions N and N-1 supported.
- Rust: A new version roughly every six weeks, with versions N and N-1 supported.
- GCC: A new version with each minor release. Versions introduced in odd minor releases are not the default and have a shorter support period. See *Section 25.2, "Compiler for user space applications and libraries"* for details. The libraries `libgcc` and `libstdc++` are described in *Section 25.5, "GCC and C++ runtime libraries"*.
- CLI and SDK for Public Cloud: A new version every quarter. The new version replaces the previous one.
- Python interpreter, library and pip (Developer Python): A new version annually.
- Time zone data files: A new version whenever new definitions become available.

2 Ansible update strategy

Ansible is an open source IT automation engine that simplifies provisioning, configuration management, application deployment and orchestration by using a human-readable, agentless language.

Lifecycle category

Agile

Release cycle

Per component, see *Table 1, "Ansible components and release frequency"*

Support scope

On SUSE Linux, the Ansible interpreter and the Linux system roles are supported. On SLES for SAP, the SAP roles and playbooks are supported as well. Support covers only the roles and playbooks delivered by SUSE.

TABLE 1: ANSIBLE COMPONENTS AND RELEASE FREQUENCY

Component	Packages	Release frequency
Ansible interpreter	`ansible`, `ansible-core`	Twice a year (typically May and November)
Linux System Roles	`ansible-linux-system-roles`	Frequently
SAP roles and playbooks	`ansible-sap-install`, `ansible-sap-infrastructure`, `ansible-sap-operations`, `ansible-sap-playbooks`	Irregularly
Unsupported SAP roles	`ansible-sap-launchpad`	Irregularly

3 Apache HTTP server update strategy

The Apache HTTP Server (Apache) is a feature-rich, open source Web server. In SUSE Linux 16.1, Apache follows the stable lifecycle model. This model balances the delivery of security fixes and other relevant upstream changes with the compatibility and long-term stability expected from SUSE Linux systems.

3.1 Upstream compatibility policy

The Apache HTTP Server Project maintains stable release branches, identified by even-numbered minor versions. Within a stable branch, upstream designs revisions to remain compatible to the maximum possible extent. This applies to the module API, the runtime configuration and the compile-time configura-

tion (the `configure` options supported by one revision remain supported by later ones). Features may be added, and features may be deprecated with appropriate documentation. This compatibility-oriented policy is what allows SUSE to update Apache regularly during General Availability (GA) without introducing disruptive changes.



Note

Upstream compatibility does not constitute an unconditional guarantee for every third-party module, local customization or deployment configuration. Verify new Apache package versions with your particular configuration and module set before deploying updates in production.

3.2 Update strategy and lifecycle model

For Apache in SUSE Linux 16.1, SUSE maintains the stable update strategy, which combines a rolling-style update approach during GA with a more conservative approach during LTS.

Lifecycle model

Stable

GA phase

For SUSE Linux minor releases in GA, the Apache package is upgraded when a new upstream Apache version is released. New upstream versions are typically released several times per year.

LTS phase

When a SUSE Linux minor release enters LTS, the Apache version is frozen. Version upgrades are avoided during LTS unless they are considered necessary. Potential version upgrades are evaluated on a case-by-case basis, taking into account security, maintenance and compatibility requirements.

The LTS freeze does not prevent an exceptional version update when one is required. In particular, SUSE retains the option of performing an emergency version update when necessary to address a critical security vulnerability or other exceptional requirement.

4 Cockpit update strategy

Cockpit is a Web-based graphical interface that enables you to manage most administration tasks from one place.

Cockpit follows a balanced lifecycle strategy and is updated and supported according to the following rules:

- Each SUSE Linux minor release comes with the latest Cockpit version.
- Support for this version continues throughout the entire lifecycle of the minor release, with any necessary fixes provided through backports.



Note

SUSE may upgrade to a newer version when reasonable, for example, for bug fixes, security updates or usability improvements, especially during LTS.

5 Container components update strategy

Container technologies evolve rapidly upstream. To provide timely enhancements, security fixes and ecosystem compatibility while maintaining enterprise stability, container components use two strategies: an agile approach for standard components and a long-term stable model for one Docker option.

5.1 Agile lifecycle model for container ecosystems

Most container components follow an agile lifecycle model, aligning closely with upstream project releases. This applies to general container infrastructure packages, including `runc`, `docker`, `podman` and their respective ecosystems, as well as `helm`.

Packages trail corresponding upstream project releases with an intentional buffer window reserved for enterprise hardening, security auditing and integration testing.

Updated versions are released across all supported SUSE Linux 16 codestreams, including versions under LTSS (Long Term Service Pack Support). If a new upstream major version introduces severe backward-incompatible changes, SUSE defers its introduction until the next minor release.

5.2 Long-term stable Docker option (`docker-stable`)

For environments that need extended release predictability without feature version bumps, SUSE provides a dedicated `docker-stable` package.

Lifecycle model

Stable (pinned major and minor release)

Version alignment

Pinned at a specific major and minor release. The same version is delivered across all codestreams.

Support phases

Supported and maintained for three years from its release date.

Lifecycle transition

A new version of `docker-stable` is released at the end of the three-year support period.

6 Desktop components update strategy

In general, desktop components follow the balanced lifecycle. The individual components differ, as shown below.

TABLE 2: DESKTOP COMPONENTS

Component	Version	Lifecycle category	Update rule
GNOME desktop	48.0	Balanced	Updated with bug fixes and compatible minor point releases within the GNOME 48 stream.
GStreamer, PipeWire, Flatpak	Latest stable release	Balanced	Updated via maintenance updates for bug fixes and security patches within the shipped stable branch.
Firefox	Minimum 140.3	Balanced (ESR)	Follows the Mozilla Firefox ESR (Extended Support Release) update lifecycle.
WebKit	Minimum 2.46	Agile	Periodic update cycle driven by critical CVEs; updated to newer upstream releases rather than backporting fixes.
BRLTTY	Latest upstream stable in 16.1	Stable	Maintained with bug fixes only during this minor release lifecycle.

Component	Version	Lifecycle category	Update rule
Qt	Qt 6, initial version 6.9	Balanced	Updated with bug fixes and backward-compatible minor updates aligned with minor releases.
KDE	—	N/A	Not a standard SUSE Linux delivery. Available only in PackageHub 16.

7 Go update strategy

Go is an open source, statically typed, compiled programming language designed for building reliable and efficient software.

7.1 Lifecycle and support policy

On SUSE Linux 16, Go and its components are updated and supported according to the following rules:

Lifecycle category
: Agile

Release cycle
: In sync with upstream, which nominally releases every six months, in February and August. Updates are integrated within a two-month sliding window following each upstream release.

Supported versions
: Versions N and N-1, mirroring the upstream support policy. When a new major version is released, the oldest supported version enters a two-month sliding grace period for transition before retirement.

Minor releases
: Go minor releases are made available shortly after the upstream release.

Parallel installation
: Supported. Versions are parallel installable and undergo automated testing. The `go` metapackage always points to the latest go1.x package.

Support phases
: General Support and LTS for all relevant SUSE Linux minor releases

7.2 Building Go applications

When building applications, developers have the following options regarding versions:

Latest recommended version

Depend on a minimum version, for example, `BuildRequires: golang(API) >= go1.23`.

Controlled timing of build changes

Pin a specific `go1.x` package, for example, `BuildRequires: golang(API) == go1.23`.

Minimum version

In both cases, use the `go1.x` version given in the `go` directive of the application's `go.mod` file, because the Go compiler now requires it as a minimum version.

7.3 FIPS compliance

Alongside standard `go1.xx` releases, SUSE provides a parallel `go1.xx-openssl` variant. This version is specifically patched to use OpenSSL for FIPS-compliant cryptographic operations.

Starting with `go1.24`, Go upstream introduced an integrated FIPS 140-3 cryptographic implementation. As this implementation achieves certification, the parallel `go1.xx-openssl` variant will be deprecated in favor of the upstream-supported FIPS mechanism. SUSE packaging follows the upstream deprecation schedule and support lifecycle.

8 High Availability update strategy

The High Availability (HA) extension stack in SUSE Linux 16 provides mission-critical clustering, resource management, automated fencing, Web-based management and cluster storage services.

8.1 HA stack components

The HA stack is divided into core clustering components and cluster storage components.

TABLE 3: HA CORE STACK

Package	Description
pacemaker	High-availability cluster resource manager

Package	Description
corosync	Cluster engine providing infrastructure messaging and membership services
sbd	STONITH Block Device for node fencing using shared storage
resource-agents	Open Cluster Framework (OCF) resource agents for managing cluster services
fence-agents	Fencing agents for power control and node isolation
crmsh	Cluster management command-line interface
hawk	High Availability Web Konsole (Hawk) for cluster management and monitoring. Requires Ruby, see *Section 19, "Ruby delivery strategy"*.

TABLE 4: HA STORAGE STACK

Package	Kernel integration	Description
drbd-kmp	Out of tree	Distributed Replicated Block Device (DRBD) kernel module, managed outside of the kernel tree
drbd-utils	Userland	Userland management utilities for DRBD
lvmlockd	Userland	LVM lock daemon for clustered LVM volumes (sub-package of lvm2)
gfs2-kmp	In tree	Global File System 2 (GFS2) kernel module, part of the kernel tree
gfs2-utils	Userland	Userland utilities for managing GFS2 file systems
cluster-md-kmp	In tree	Clustered Multi-Device RAID kernel module, part of the kernel tree

8.2 Update policy and lifecycle rules

Lifecycle category

Balanced

Supported versions

Only one active package version is maintained per SUSE Linux 16 minor release.

Release cycle

Package versions are updated across minor releases to synchronize with the latest stable upstream releases.

In-tree kernel modules

The `kernel-module` packages integrated into the kernel tree (`gfs2-kmp` and `cluster-md-kmp`) follow the kernel tick-tock version update model.

Out-of-tree and userland storage components

The out-of-tree kernel module (`drbd-kmp`) and the userland storage tools (`drbd-utils`, `lvmlockd` and `gfs2-utils`) are updated in alignment with stable upstream releases and kernel compatibility updates.

Support phases

Supported throughout General Support and LTS as part of the SUSE Linux Enterprise High Availability extension

9 Java update strategy

On SUSE Linux 16, Java is delivered in several versions that are maintained independently of the minor release cycle.

Lifecycle category

Agile

Supported versions

See *Table 5, "Java versions and support end dates"*

Parallel installation

Supported. Different Java versions can be installed concurrently.

Release cycle

Java versions are maintained independently of the SUSE Linux minor release cycle, and Java packages can be updated in an agile manner.

Support phases

Each version is supported throughout General Support and LTS.

TABLE 5: JAVA VERSIONS AND SUPPORT END DATES

Version	Supported until
Java 17	December 2027
Java 21	October 2031
Java 25	October 2033

10 `libpng` update strategy

`libpng` is the official reference library for supporting the Portable Network Graphics (PNG) image format. It is a fundamental graphical component used across desktop environments, image processing utilities and multimedia frameworks in SUSE Linux.

The `libpng` package follows the balanced lifecycle model across all SUSE Linux 16 products. The update strategy is governed by the following rules:

Lifecycle category

Balanced, across all SUSE Linux 16 products. This permits updating to new upstream patch releases to deliver security fixes, bug fixes and performance improvements, while safeguarding application compatibility and soname stability.

Compatibility

Upstream `libpng` maintains strict ABI compatibility within active release series. Updates are patch-level version bumps that keep the same soname, so downstream applications and graphical libraries do not break or need rebuilds.

Supported versions

For every `libpng` series that remains in upstream maintenance, the latest patch version is delivered to all supported SUSE Linux 16 minor releases as it becomes available.

11 `man-pages` update strategy

The `man-pages` package provides the primary reference documentation for Linux system calls, C library functions, device files, file formats and system administration conventions.

Lifecycle category

Balanced, across all SUSE Linux 16 minor versions

Release cycle

With each new SUSE Linux 16 minor release, `man-pages` is updated to the latest upstream version.

Kernel and API alignment

The primary objective of these periodic version updates is to keep documentation in sync with new kernel updates, system call modifications and C library (`glibc`) enhancements introduced in each minor release.

12 MariaDB update strategy

MariaDB is a high-performance, open source relational database management system (RDBMS) that was created as a community-developed fork of MySQL to ensure the software remains free and accessible under the GPL license.

Lifecycle category

Balanced. Only upstream LTS MariaDB releases are supported.

Versions

Each minor release gets a new MariaDB version. The starting version is 11.4.5.

Support phases

Each version is supported for the General Support and LTS periods of its minor release.

Minor version updates

Aligned with upstream releases and provided exclusively for the `mariadb` and `mariadb-connector-c` packages.

13 Network UPS Tools update strategy

Network UPS Tools (NUT) is an open source software suite that supports power devices, such as uninterruptible power supplies (UPS), power distribution units and controllers. It simplifies monitoring of power hardware and ensures orderly system shutdowns during power outages.

Lifecycle category

Stable, with on-demand updates. Newer upstream versions maintain strict backward compatibility, so upgrades do not introduce disruptive changes for customers on any 16.x minor release.

Update drivers

Because upstream versions are backward compatible, updates are driven by new features that are difficult or impractical to backport, mainly support for new hardware such as new UPS models.

Scope of updates

When a new version with valuable features is available upstream, the packages are updated across all supported codestreams in General Support and maintenance.

Flexible maintenance delivery

The upstream release schedule is irregular, so new versions are submitted as they are released. They are delivered within a new minor release or as maintenance updates for already released codestreams.

14 `node.js` update strategy

`Node.js` is an open source, cross-platform JavaScript runtime environment for building server-side applications outside of a Web browser. It is built on Google Chrome's V8 engine and uses an event-driven, non-blocking I/O model, making it highly efficient for creating scalable network applications.

Lifecycle category

Agile

Versions

The version is kept in sync with the upstream LTS version. Each minor release provides the latest Active LTS version (see Node.js previous releases (https://nodejs.org/en/about/previous-releases)), starting with `nodejs 22`, and ships it as the default.

Minor releases

Upstream minor and patch updates for active Node.js streams are delivered via maintenance updates, primarily triggered by upstream security releases or upon request.

Parallel installation

Supported. Different `node.js` versions can be installed in parallel.

Support phases

Versions in a minor release are supported until the end of LTS.

Building applications

To use the most recent recommended version in the repository, use `BuildRequires: nodejs >= 22`.

15 Perl update strategy

Perl is a high-level, general-purpose programming language, originally developed for text manipulation, that is now used for system administration, Web development and network programming.

Lifecycle category

Stable

Versions

The version shipped with the initial release remains fixed throughout the SUSE Linux 16 lifecycle. Maintenance updates deliver bug and security fixes only, with no major version upgrades planned.

16 PHP update strategy

PHP is an open source server-side scripting language designed for Web development. It is used to create dynamic, interactive content that works with databases.

Lifecycle category

Balanced

Versions

The starting version is 8.4. A new PHP version is shipped with each SUSE Linux 16 minor release.

Release cycle

Patch-level updates are delivered throughout the support period of each minor release.

17 PostgreSQL update strategy

PostgreSQL is an open source object-relational database management system (ORDBMS) known for its reliability and its support for advanced data types and complex queries.

Lifecycle category

Balanced

Supported versions

All versions maintained by the upstream community.

Release cycle

When upstream releases a new version, it is added to each minor release under General Support or LTS.

Parallel installation

Supported. Multiple PostgreSQL major versions can be installed side by side on the same system.

End of life

SUSE support for a PostgreSQL version ends concurrently with upstream end of life. Once a version reaches upstream EOL, it remains in the repositories for migration purposes but receives no further security patches or bug fixes.

18 Python update strategy

SUSE Linux 16.1 includes several Python interpreter versions with distinct support lifecycles. Rather than applying a single blanket classification, each Python interpreter stream is categorized individually based on its update model.

18.1 Python interpreter lifecycles

TABLE 6: PYTHON INTERPRETERS

Interpreter	Category	Supported during	Description
System Python	Stable	General Support and LTS of 16.1, and of	The version for which SUSE provides a large set of Python modules needed for applications

Interpreter	Category	Supported during	Description
		each future 16.x minor release	and solutions. The version and its modules remain stable throughout the lifecycle.
Developer Python	Agile	General Support only, excluding LTS (two years)	A more recent Python version, intended for use with Python wheels from PyPI, installed and maintained through `pip` and `pipx`.
Legacy Python	Stable	General Support and LTS of the minor release in which it is provided	Provided in future 16.x releases when System Python moves to a newer version. The previous System Python and all its modules remain supported for compatibility.

18.2 Python release cycle

System Python

Python 3.13 in SUSE Linux 16.1. The current plan, which is subject to change, is to provide newer System Python versions in future 16.x releases.

Developer Python

A new Developer Python version is released with each minor release and is supported only for the General Support period of that release (two years).

19 Ruby delivery strategy

The Ruby stack is not delivered as a standard component of SUSE Linux 16.

However, specific components that depend on Ruby include it as a supported dependency.

Lifecycle category

Not applicable. Ruby is supported only as a dependency of the components below.

Hawk

Part of SUSE Linux Enterprise High Availability. It is built on Ruby on Rails and requires a Ruby runtime.

RMT (Repository Mirroring Tool)

Requires Ruby. Ruby is provided as a supported exception.

20 Rust update strategy

Rust is a multi-paradigm, general-purpose programming language designed for performance and safety, especially safe concurrency.

On SUSE Linux 16, Rust and its components are updated and supported according to the following rules:

Lifecycle category

Agile

Release cycle

In sync with upstream, which releases every six weeks

Supported versions

Versions N and N-1

Upgrade window

Six weeks, to perform upgrades and find potential issues

Parallel installation

Supported. Parallel installable versions are available and undergo automated testing.

Support phases

General Support and LTS for all relevant SUSE Linux minor releases

21 Security components update strategy

This section defines the update policy and lifecycle strategy for core security components in the SUSE Linux 16 codestream.

21.1 Stable security components

These components remain the same within a minor release. New versions are typically introduced only with the next minor release.

OpenSSL

The version is fixed for FIPS 140-3 certification within the minor release.

GnuTLS, Nettle

The version is fixed for FIPS 140-3 certification.

libgcrypt

The version is fixed for FIPS 140-3 certification.

OpenSSH

Remote login and communication protocol toolset; maintained with bug and security fixes within the minor release.

liboqs, oqs-provider

Post-quantum cryptography library and OpenSSL provider; maintained with bug and security fixes within the minor release.

GnuPG

OpenPGP encryption and signing toolset; maintained with bug and security fixes within the minor release.

krb5

Kerberos authentication protocol implementation; maintained with bug and security fixes within the minor release.

Smartcards and U2F tokens

Hardware authentication middleware and libraries (such as OpenSC, pcsc-lite, and libfido2); maintained with bug and security fixes within the minor release.

IMA tools

Integrity Measurement Architecture management utilities; maintained with bug and security fixes within the minor release.

openCryptoki

PKCS#11 implementation for hardware security modules; maintained with bug and security fixes within the minor release.

rage

Modern file encryption tool and Rust implementation of the age format; maintained with bug and security fixes within the minor release.

rsign2

Digital signature utility and Rust implementation of minisign; maintained with bug and security fixes within the minor release.

PAM stack, polkit

Pluggable Authentication Modules and PolicyKit authorization framework; maintained with bug and security fixes within the minor release.

21.2 Balanced security components

These components can be updated with a maintenance update or receive a version update with a new minor release.

firewalld

Updated with compatible minor and patch releases during maintenance updates or new minor releases.

SELinux

Policies are updated with monthly snapshots.

Network Security Services (mozilla-nss)

Initial versions establish the baseline for FIPS 140-3 evaluation. Maintenance updates align with Firefox ESR releases to deliver compatible security and protocol updates.

strongSwan

New versions are typically introduced with a new minor release.

Stunnel

Updated with compatible point releases during maintenance updates or new minor releases.

TPM libraries and tools

Trusted Platform Module software stack and utilities; updated with compatible point releases during maintenance updates or new minor releases.

21.3 Agile security components

These components are kept up to date with upstream.

Cosign, Rekor

Kept up to date with upstream.

ClamAV

Kept up to date with upstream.

scap-security-guide

Kept up to date with upstream.

ca-certificates-mozilla

Updated when a new NSS certificate bundle is released.

22 `smartmontools` update strategy

`smartmontools` (S.M.A.R.T. Monitoring Tools) is a software suite for controlling and monitoring storage systems using the Self-Monitoring, Analysis and Reporting Technology (S.M.A.R.T.) built into ATA, SATA, SCSI, SAS and NVMe hard drives and solid-state drives. It plays a critical role in early disk failure detection and hardware monitoring.

Lifecycle category

Stable, with on-demand updates. Upstream versions maintain strict backward compatibility, so updates do not introduce disruptive changes to system configurations, script integrations or reliability.

Update drivers

New hardware enablement and support for new drive models. Updates are driven by features and hardware support requested by customers that are difficult or impractical to backport.

Scope of updates

When a new version with useful features or new hardware support is available upstream, the packages are updated across all supported codestreams in active maintenance.

Delivery

The upstream release schedule is irregular, so new versions are submitted as they are released. They are delivered within a new minor release or as maintenance updates for already released codestreams.

23 **`sudo`** update strategy

`sudo` (Superuser Do) is a critical system administration utility that allows permitted users to execute commands as a superuser or another user according to security policy. It is a foundation for privilege management, access control and audit logging in SUSE Linux 16.

Lifecycle category

Balanced. This balances system stability with timely maintenance fixes, security hardening and upstream bug fixes.

Release cycle

With every new minor release, **`sudo`** is updated to a newer patch release. This delivers cumulative stability and security updates while preserving configuration and workflow compatibility.

24 Time zone update strategy

Data files for time zones follow the agile lifecycle.

Lifecycle category

Agile

Release cycle

When a new set of time zone definitions becomes available, the package is updated across all supported SUSE Linux systems.

Supported versions

Only the latest version. The package is updated without ongoing support for older versions, the same approach as in SLE 15.

25 Toolchain components update strategy

The toolchain components include the following tools: the GNU C library, the GCC and G++ compilers, `binutils`, GDB and LLVM. Each component has its own update strategy, described in the corresponding sections.

25.1 GNU C library (`glibc`)

Lifecycle category

Stable

Versions

The initial version is 2.40. The package is updated with each minor release if there are reasons for changes, for example feature requests or performance tuning.

Compatibility

Updates are backward compatible for dynamic linking, so programs built on earlier SUSE Linux 16 releases keep running.

Deprecated symbols

Symbols deprecated in upstream `glibc` are not declared for the compiler and are not available for link editing (static linking).

25.2 Compiler for user space applications and libraries

Developers of user-space applications can use the supported GNU Compiler Collection (GCC) C and C++ built-in compilers. Compilers for other languages, cross-compilers and accelerator offloading compilers are not available on SUSE Linux from standard repositories. Developers can install them from PackageHub with community support.

Lifecycle category

Agile, with the tick-tock model described below.

Initial version

GCC 15 in SUSE Linux 16.1. The tick-tock model applies to later releases.

Tick releases (even minor releases)

Introduce a new major GCC version as the default compiler. It is supported during LTS of the introducing minor release and of the next minor release. Example: GCC 17, introduced in 16.2, is supported until the end of LTS of 16.3.

Tock releases (odd minor releases)

Include a new non-default major GCC version. To use it, invoke the binaries `gcc-x`, `g++-x` and `gfortran-x` explicitly. Non-default versions are supported for 24 months. In 16.1, this is GCC 16, invoked with `gcc-16`, `g++-16` and `gfortran-16`.

25.2.1 Supported compiler flags

Any combination of the following compiler flags is supported:

- `-O0`, `-O2` and `-O3`
- `-ffast-math`
- `-flto`
- `-fpie` and `-fno-pie`
- `-fPIC`
- `-g`

The following options are also supported on AMD64/Intel 64:

- `-march=x86-64-v2` (the default one)
- `-march=x86-64-v3`
- `-march=x86-64-v4`

Other compiler flags are not supported by SUSE. SUSE can assist in reporting issues to the upstream GCC project.

25.2.2 Supported language versions

C

ISO/IEC 9899:2024 (C23) with GNU extensions

C++

With the default GCC 15: ISO/IEC 14882:2017 (C++17) with GNU extensions. With the non-default GCC 16: ISO/IEC 14882:2020 (C++20) with GNU extensions.

Fortran

Up to Fortran 95 (ISO/IEC 1539:1997), with specific features from later standards as described in the manual.

25.3 Kernel module compiler

Purpose

Building a kernel module requires the same compiler version that was used to build the kernel. SUSE therefore provides that GCC version, initially GCC 13.N.

Restrictions

The compiler is not intended for general use and may be dropped in a future minor release.

25.4 Build compiler

Version

SUSE Linux 16 packages are built internally with GCC 13. This build compiler is provided as an unsupported package in PackageHub.

Notes

Package maintainers can use the newer GCC in the internal build service, but must be aware of possible ABI issues, for example, by avoiding linking code written in different C++ standards.

25.5 GCC and C++ runtime libraries

Lifecycle category

Stable (libgcc, libstdc++)

Release cycle

Updated yearly to the versions of a new GCC major version, in all minor releases under LTS. Updates are delivered as maintenance updates.

Support phases

Fully supported during General Support and LTS of each minor release.

Security

These runtime libraries receive maintenance and security updates because they are dynamically linked by running user-space applications. They are hardened for processing untrusted input and are available for use when addressing security incidents.

25.6 GNU Binutils

Release cycle

Upgraded to the latest upstream version in all SUSE Linux 16 minor releases under General Support or LTS.

25.7 GNU project debugger

Release cycle

Updated to the newest major version in all SUSE Linux 16 minor releases under General Support or LTS.

Compatibility

Certain functionality may be removed when GDB is updated to a newer major version.

25.8 LLVM

Scope

LLVM is available exclusively for use with Mesa. Any other use is not supported.

Front-ends

Front-ends such as Clang are not provided. They are available only from a community-supported repository.

25.9 Compatibility and deprecation policy

Dynamic-linking compatibility

SUSE maintains backward dynamic-linking compatibility for `glibc` and the C++ compiler runtime library. A binary built on an earlier SUSE Linux 16 minor release runs correctly on a later minor release.

Deprecated features

Features deprecated by upstream are removed either in newer major versions of compilers or from all toolchain components in later minor releases.

`glibc` symbols

Deprecated symbols are removed from header files and are not available for link editing. Code that uses them no longer compiles or links statically.

25.10 Security considerations

Development tools

Development tools such as the compiler are not hardened to process untrusted input.

Runtime libraries

The GCC and C++ runtime libraries are the only parts of the toolchain hardened for this purpose.

26 `util-linux` update strategy

`util-linux` is a standard suite of essential system utilities central to Linux system administration, storage management, hardware detection and core operating system operations.

Lifecycle category

Stable

Release cycle

Updated to the latest upstream release with each new SUSE Linux 16 minor release. This continues the maintenance practice established in SLE 15.

Compatibility

Upstream releases maintain strict backward compatibility (approximately 99%), so updates do not disrupt administrative workflows, scripts or core operations.

Rationale

Security patches and bug fixes can be backported, but customers frequently need new utility features, flags and hardware capabilities that are difficult or impractical to backport. Delivering updated upstream versions provides them.

27 Valkey update strategy

Valkey is a high-performance data structure server for key/value workloads. It supports a wide range of native data structures and an extensible plug-in system for adding new data structures and access patterns.

Versions

Kept in sync with upstream (see Valkey releases (https://valkey.io/topics/releases/)). Each minor release has the latest available version.

Release cycle

Backward-compatible Valkey minor releases are published promptly after the upstream release. Patch updates are delivered continuously to minor releases as soon as they are released.

Support phases

General Support and LTS of all SUSE Linux 16 minor releases.

Exceptions

Security fixes are backported if upstream does not issue a new release that includes the fix.

28 Vim update strategy

Vim (Vi IMproved) is an advanced text editor providing comprehensive text editing capabilities and extensive plug-in support.

Lifecycle category

Agile, with a flat version model. This continues the strategy established in SLE 15.

Release cycle

When a new upstream release is available, `vim` is updated to the latest version across all supported SUSE Linux 16 minor releases. active minor releases and those in LTS.

Supported versions

Only the latest version. Support for the older version ends when the update is released.

Rationale

Vim has an active upstream codebase, so deploying the latest release keeps patches easy to apply and issues straightforward to resolve.

29 Virtualization components update strategy

This section documents the lifecycle management and update strategy for virtualization components in SUSE Linux 16.

29.1 Included components and target versions

SUSE Linux 16 provides a modern virtualization stack centered around KVM and QEMU. The primary shipped components and their initial baseline versions in SUSE Linux 16 include:

- `KVM`: Inherited directly from the Linux kernel lifecycle
- `QEMU`: Version 11.0.4 (updated in subsequent minor releases)
- `libvirt`: Version 12.4.0, with VM confinement enabled by default using SELinux
- `open-vm-tools`: Version 13.1.0 (updated in subsequent minor releases)
- `snpguest`: Version 0.10.0, with remote and local attestation as a technology preview

- `virt-top`: Version 1.1.1
- `virt-manager`: Version 5.1.0, with direct XML editing via `virsh` and VNC viewing recommended
- `Hyper-V`: Version 9 guest enlightenments and drivers (kernel)

29.2 Lifecycle model and update policy

Lifecycle category

Balanced

Release cycle

Every new minor release is updated to the latest upstream versions of all supported virtualization packages.

Support phases

All SUSE Linux 16 minor releases, throughout General Support and LTS.

Maintenance updates

In exceptional cases where packages maintain strict backward compatibility, version updates can be delivered within an already released minor release through maintenance updates.

Security

Security fixes are backported as necessary when upstream does not promptly issue a point release that contains the fix.

Quality assurance

Automated testing covers all key enterprise use cases across critical combinations of host and guest operating systems.

30 Wireshark update strategy

Wireshark is an open source network protocol analyzer used for network troubleshooting, analysis, software and communications protocol development and education. Because Wireshark parses a wide array of complex network protocols, it frequently receives updates for security vulnerabilities (CVEs) and protocol support.

Lifecycle category

Balanced. This balances strict stability with timely security patching and protocol analysis capabilities.

Maintenance releases (X.Y.Z)

Delivered continuously within SUSE Linux 16 minor releases and for previously released minor releases in maintenance. They contain security fixes and bug repairs only, with no new or changed features.

Minor versions (X.Y.0)

Updated with a SUSE Linux 16 minor release when a new upstream stable branch is selected for SUSE Linux. Such updates can introduce new protocol dissectors and features.

Major versions (X.0.0)

When upstream releases a new major version, SUSE evaluates case by case whether to transition. Any transition is documented in the release notes.

31 Legal Notice

Copyright© 2006– 2026 SUSE LLC and contributors. All rights reserved.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or (at your option) version 1.3; with the Invariant Section being this copyright notice and license. A copy of the license version 1.2 is included in the section entitled “GNU Free Documentation License”.

For SUSE trademarks, see https://www.suse.com/company/legal/. All other third-party trademarks are the property of their respective owners. Trademark symbols (®, ™ etc.) denote trademarks of SUSE and its affiliates. Asterisks (*) denote third-party trademarks.

All information found in this book has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Neither SUSE LLC, its affiliates, the authors, nor the translators shall be held liable for possible errors or the consequences thereof.

A GNU Free Documentation License

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. 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 non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.
```

```
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.