systemd-sysext

- Description
- Uses
- Mutability
- Commands
- Options
- Exit status
- See Also

systemd

systemd-sysext

8

systemd-sysext

systemd-sysext.service

systemd-sysext-initrd.service

systemd-sysext-sysroot.service

systemd-confext

systemd-confext.service

systemd-confext-initrd.service

systemd-confext-sysroot.service

Activates System Extension Images

systemd-sysext

OPTIONS

COMMAND

`systemd-sysext.service`

systemd-confext

OPTIONS

COMMAND

`systemd-confext.service`

Description

`systemd-sysext` activates/deactivates system extension images. System extension images may – dynamically at runtime — extend the `/usr/` and `/opt/` directory hierarchies with additional files. This is particularly useful on immutable system images where a `/usr/` and/or `/opt/` hierarchy residing on a read-only file system shall be extended temporarily at runtime without making any persistent modifications.

System extension images should contain files and directories similar in fashion to regular operating system tree. When one or more system extension images are activated, their `/usr/` and `/opt/` hierarchies are combined via `overlayfs` with the same hierarchies of the host OS, and the host `/usr/` and `/opt/` overmounted with it ("merging"). When they are deactivated, the mount point is disassembled — again revealing the unmodified original host version of the hierarchy ("unmerging"). Merging thus makes the extension's resources suddenly appear below the `/usr/` and `/opt/` hierarchies as if they were included in the base OS image itself. Unmerging makes them disappear again, leaving in place only the files that were shipped with the base OS image itself.

Files and directories contained in the extension images outside of the `/usr/` and `/opt/` hierarchies are *not* merged, and hence have no effect when included in a system extension image. In particular, files in the `/etc/` and `/var/` included in a system extension image will *not* appear in the respective hierarchies after activation.

System extension images are strictly read-only by default. On mutable host file systems, `/usr/` and `/opt/` hierarchies become read-only while extensions are merged, unless mutability is enabled. Mutability may be enabled via the `--mutable=` option and the `Mutable=` option in the configuration file; see "Mutability" below for more information.

Various command options can be configured globally via configuration files. See `sysext.conf(5)` for details.

System extensions are supposed to be purely additive, i.e. they are supposed to include only files that do not exist in the underlying basic OS image. However, the underlying mechanism (overlayfs)

also allows overlaying or removing files, but it is recommended not to make use of this.

System extension images may be provided in the following formats:

1. Plain directories or btrfs subvolumes containing the OS tree
2. Disk images with a GPT disk label, following the UAPI.2 Discoverable Partitions Specification
3. Disk images lacking a partition table, with a naked Linux file system (e.g. erofs, squashfs or ext4)

These image formats are the same ones that `systemd-nspawn(1)` supports via its `--directory=`/`--image=` switches and those that the service manager supports via `RootDirectory=`/`RootImage=`. Similar to them they may optionally carry Verity authentication information. LUKS encrypted extension images carrying a TPM2 token (as enrolled via the `--tpm2-device=` option of `systemd-cryptenroll(1)`) are unlocked via the local TPM2 automatically, without any passphrase. This may be turned off via the `$SYSTEMD_DISSECT_TPM2_UNLOCK` environment variable.

System extensions are searched for in the directories `/etc/extensions/`, `/run/extensions/` and `/var/lib/extensions/`. The first two listed directories are not suitable for carrying large binary images, however are still useful for carrying symlinks to them. The primary place for installing system extensions is `/var/lib/extensions/`. Any directories found in these search directories are considered directory based extension images; any files with the `.raw` suffix are considered disk image based extension images. When invoked in the initrd, the additional directory `/.extra/sysext/` is included in the directories that are searched for extension images. Note however, that by default a tighter image policy applies to images found there, though, see below. This directory is populated by `systemd-stub(7)` with extension images found in the system's EFI System Partition.

During boot, system and configuration extension images are activated automatically if the `systemd-sysext.service` and `systemd-confext.service` services are enabled. Note that these services run only after the underlying file systems where system and configuration extensions may be located have been mounted. To make it possible to ship resources that are processed by subsystems running in the earliest boot stages (for example, system services or `systemd-sysusers(8)` definitions), the `systemd-sysext-sysroot.service` and `systemd-confext-sysroot.service` initrd services are provided. Currently, these services cannot be used to merge system extensions from `/sysroot/var/lib/extensions/` and configuration extensions from `/sysroot/var/lib/confexts/` when the `/var/` partition is split off. These extensions are later merged by the `systemd-sysext.service` and `systemd-confext.service` services during the main OS boot process.

Also, see the Portable Services page for a simple mechanism for shipping system services in disk images, in a similar fashion to OS extensions. Note the differences in isolation between these two mechanisms: while system extensions directly extend the underlying OS image with additional files that appear as if they were shipped in the OS image itself and thus imply no security isolation, portable services imply service-level sandboxing in one way or another. Binaries shipped in system extensions can only link against libraries from the host if the extension is bound to the version of

the host OS (see further below) or otherwise have to use static linking. Portable services, however, are fully decoupled from the host as they ship their own library dependencies.

The `systemd-sysext.service` and `systemd-confext.service` services are guaranteed to finish start-up before `basic.target` is reached; i.e., by the time regular services initialize (those which do not use `DefaultDependencies=no`), the files and directories provided by system and configuration extensions are available in `/usr/`, `/opt/`, and `/etc/` and may be accessed.

System and configuration extensions can also be used to extend the initrd, and the `systemd-sysext-initrd.service` and `systemd-confext-initrd.service` initrd services are provided. Note that some limitations apply: resources that are used in the earliest boot stages of the initrd (e.g. system services) cannot be updated.

Note that there is no concept of enabling/disabling installed system extension images: all installed extension images are automatically activated at boot. However, you can place an empty directory named like the extension (no `.raw`) in `/etc/extensions/` to "mask" an extension with the same name in a system folder with lower precedence. It is also possible to disable automatic merging altogether using the `rd.systemd.sysext=`, `rd.systemd.confext=`, `systemd.sysext=`, and `systemd.confext=` kernel command line options. Note that `systemd-sysext-sysroot.service` and `systemd-confext-sysroot.service` are controlled by the `systemd.sysext=` and `systemd.confext=` options, as these services merge system and configuration extensions for the main system, not for the initrd.

A simple mechanism for version compatibility is enforced: a system extension image must carry a `/usr/lib/extension-release.d/extension-release.NAME` file, which must match its image name, that is compared with the host `os-release` file: the contained `ID=` fields have to match unless `_any` is set for the extension. If the extension `ID=` is not `_any`, the `SYSEXT_LEVEL=` field (if defined) has to match. If the latter is not defined, the `VERSION_ID=` field has to match instead. If the extension defines the `ARCHITECTURE=` field and the value is not `_any` it has to match the kernel's architecture reported by `uname(2)` but the used architecture identifiers are the same as for `ConditionArchitecture=` described in `systemd.unit(5)`. `EXTENSION_RELOAD_MANAGER=` can be set to 1 if the extension requires a service manager reload after application of the extension. `EXTENSION_RESTART_UNITS=` and `EXTENSION_RELOAD_OR_RESTART_UNITS=` can be set to a whitespace-separated list of unit names (e.g. `EXTENSION_RESTART_UNITS="my.service other.service"`) that `systemd-sysext`/`systemd-confext` should restart resp. reload (or restart, if the unit does not support reloading) after merging or refreshing the extension, and after unmerging it. Use `EXTENSION_RESTART_UNITS=` for units whose binary the extension replaces (a reload would not pick up the new binary), and `EXTENSION_RELOAD_OR_RESTART_UNITS=` for units the extension extends with configuration drop-ins only (so reload-capable services are not needlessly interrupted). If a unit has disappeared, e.g., on unmerge because it was part of the extension, it is stopped instead. If the same unit is listed in both fields, `EXTENSION_RESTART_UNITS=` takes precedence. Listing any unit in either field implies `EXTENSION_RELOAD_MANAGER=1`, as the service manager would otherwise not yet (or no longer) see the unit files shipped by the extension. Note that these settings do not take effect when the extension is merged for the system already from the initrd and extensions should still ship a `.wants/` or `.upholds/` symlink from the unit's install target to ensure the unit starts at boot. System extensions should not ship a `/usr/lib/os-release` file (as that would be merged into the

host `/usr/` tree, overriding the host OS version data, which is not desirable). The `extension-release` file follows the same format and semantics, and carries the same content, as the `os-release` file of the OS, but it describes the resources carried in the extension image.

The `systemd-confext` concept follows the same principle as the `systemd-sysext(8)` functionality but instead of working on `/usr` and `/opt`, `confext` will extend only `/etc`. Files and directories contained in the confext images outside of the `/etc/` hierarchy are *not* merged, and hence have no effect when included in the image. Formats for these images are of the same as sysext images. The merged hierarchy will be mounted with `nosuid` and (if not disabled via `--noexec=false`) `noexec` .

Just like sysexts, confexts are strictly read-only by default. Merging confexts on mutable host file systems will result in `/etc/` becoming read-only. As with sysexts, mutability can be enabled via the `--mutable=` option. Refer to "Mutability" below for more information.

Confexts are looked for in the directories `/run/confexts/`, `/var/lib/confexts/`, `/usr/lib/confexts/` and `/usr/local/lib/confexts/`. The first listed directory is not suitable for carrying large binary images, however is still useful for carrying symlinks to them. The primary place for installing configuration extensions is `/var/lib/confexts/`. Any directories found in these search directories are considered directory based confext images; any files with the `.raw` suffix are considered disk image based confext images.

Again, just like sysext images, the confext images will contain a `/etc/extension-release.d/extension-release.NAME` file, which must match the image name (with the usual escape hatch of the `user.extension-release.strict` `xattr(7)`), and again with content being one or more of `ID=`, `VERSION_ID=`, and `CONFEXT_LEVEL`. Confext images will then be checked and matched against the base OS layer.

Uses

The primary use case for system images are immutable environments where debugging and development tools shall optionally be made available, but not included in the immutable base OS image itself (e.g. `strace(1)` and `gdb(1)` shall be an optionally installable addition in order to make debugging/development easier). System extension images should not be misunderstood as a generic software packaging framework, as no dependency scheme is available: system extensions should carry all files they need themselves, except for those already shipped in the underlying host system image. Typically, system extension images are built at the same time as the base OS image — within the same build system.

Another use case for the system extension concept is temporarily overriding OS supplied resources with newer ones, for example to install a locally compiled development version of some low-level component over the immutable OS image without doing a full OS rebuild or modifying the nominally immutable image. (e.g. "install" a locally built package with `DESTDIR=/var/lib/extensions/mytest make install && systemd-sysext refresh`, making it available in

`/usr/` as if it was installed in the OS image itself.) This case works regardless of whether the underlying host `/usr/` is managed as immutable disk image or is a traditional package manager controlled (i.e. writable) tree.

With `systemd-confext` one can perform runtime reconfiguration of OS services. Sometimes, there is a need to swap certain configuration parameter values or restart only a specific service without deployment of new code or a complete OS deployment. In other words, we want to be able to tie the most frequently configured options to runtime updateable flags that can be changed without a system reboot. This will help reduce servicing times when there is a need for changing the OS configuration. It also provides a reliable tool for managing configuration because all old configuration files disappear when the `systemd-confext` image is removed.

Mutability

By default, merging system extensions on mutable host file systems will render `/usr/` and `/opt/` hierarchies read-only. Merging configuration extensions will have the same effect on `/etc/`. Mutable mode allows writes to these locations when extensions are merged.

The following modes are supported:

1. `disabled`: Force immutable mode even if write routing directories exist below `/var/lib/extensions.mutable/`. This is the default.
2. `auto`: Automatic mode. Mutability is disabled by default and only enabled if a corresponding write routing directory exists below `/var/lib/extensions.mutable/`.
3. `enabled`: Force mutable mode and automatically create write routing directories below `/var/lib/extensions.mutable/` when required.
4. `import`: Force immutable mode like `disabled` above, but merge the contents of directories below `/var/lib/extensions.mutable/` into the host file system.
5. `ephemeral`: Force mutable mode like `enabled` above, but instead of using write routing directory below `/var/lib/extensions.mutable/`, `systemd-sysext` will use empty ephemeral directories. This means that the modifications made in the merged hierarchies will be gone when the hierarchies are unmerged.
6. `ephemeral-import`: Force mutable mode like `ephemeral` above, but instead of ignoring the contents of write routing directories under `/var/lib/extensions.mutable/`, merge them into the host file system, like `import` does.

See "Options" below on specifying modes using the `--mutable=` command line option.

With exception of the ephemeral mode, the mutable mode routes writes to subdirectories in `/var/lib/extensions.mutable/`. Writes to `/usr/` are directed to `/var/lib/extensions.mutable/usr/`, writes to `/opt/` are directed to `/var/lib/extensions.mutable/opt/`, and, writes to `/etc/` land in `/var/lib/extensions.mutable/etc/`.

If `usr/`, `opt/`, or `etc/` in `/var/lib/extensions.mutable/` are symlinks, then writes are directed to the symlinks' targets. Consequently, to retain mutability of a host file system, create symlinks `/var/lib/extensions.mutable/etc/` → `/etc/`, `/var/lib/extensions.mutable/usr/` → `/usr/`, `/var/lib/extensions.mutable/opt/` → `/opt/` to route writes back to the original base directory hierarchy.

Alternatively, a temporary file system may be mounted to `/var/lib/extensions.mutable/`, or symlinks in `/var/lib/extensions.mutable/` may point to sub-directories on a temporary file system (e.g. below `/tmp/`) to only allow ephemeral changes. Note that this is not the same as ephemeral mode, because the temporary file system will still exist after unmerging.

Commands

The following commands are understood by both the sysext and confext concepts:

`status`
When invoked without any command verb, or when `status` is specified the current merge status is shown, separately (for both `/usr/` and `/opt/` of sysext and for `/etc/` of confext).

`merge`
Merges all currently installed system extension images into `/usr/` and `/opt/`, by overmounting these hierarchies with an `overlayfs` file system combining the underlying hierarchies with those included in the extension images. This command will fail if the hierarchies are already merged. For confext, the merge happens into the `/etc/` directory instead.

`unmerge`
Unmerges all currently installed system extension images from `/usr/` and `/opt/` for sysext and `/etc/`, for confext, by unmounting the `overlayfs` file systems created by `merge` prior.

`refresh`
A combination of `unmerge` and `merge`: if already mounted the existing `overlayfs` instance is unmounted temporarily, and then replaced by a new version. This command is useful after installing/removing system extension images, in order to update the `overlayfs` file system accordingly. If no system extensions are installed when this command is executed, the equivalent of `unmerge` is executed, without establishing any new `overlayfs` instance. Note that currently there's a brief moment where neither the old nor the new `overlayfs` file system is mounted. This implies that all resources supplied by a system extension will briefly disappear — even if it exists continuously during the refresh operation.

`list`
A brief list of installed extension images is shown.

Options

`--root=`
Operate relative to the specified root directory, i.e. establish the `overlayfs` mount not on the top-level host `/usr/` and `/opt/` hierarchies for sysext or `/etc/` for confext, but below some specified root directory.

`--force`
When merging system extensions into `/usr/` and `/opt/` for sysext and `/etc/` for confext, ignore version incompatibilities, i.e. force merging regardless of whether the version information included in the images matches the host or not.

`--always-refresh=yes|no`
When refreshing system extensions on `/usr/` and `/opt/` for sysext and `/etc/` for confext, ignore when the existing merged extensions already match what would be merged. By default the refresh is skipped when no changes are found. Note that changes done to an extension directory while it's merged are ignored without this flag (unless another extension got changed). Note that changing the contents while merged is also undefined behavior in overlayfs.

`--image-policy=policy`
Takes an image policy string as argument, as per `systemd.image-policy(7)`. The policy is enforced when operating on system extension disk images. If not specified, defaults to `root=verity+signed+encrypted+unprotected+absent:usr=verity+signed+encrypted+unprotected+absent` for system extensions, i.e. only the root and `/usr/` file systems in the image are used. For configuration extensions defaults to `root=verity+signed+encrypted+unprotected+absent`. When run in the initrd and operating on a system extension image stored in the `/.extra/sysext/` directory a slightly stricter policy is used by default: `root=signed+absent:usr=signed+absent`, see above for details.

`--mutable=BOOL|auto|import|ephemeral|ephemeral-import|help`
Set mutable mode. The special value `help` will list the known values.

`no`
force immutable mode even with write routing directories present. This is the default.

`auto`
enable mutable mode individually for `/usr/`, `/opt/`, and `/etc/` if write routing sub-directories or symlinks are present in `/var/lib/extensions.mutable/`; disable otherwise. See "Mutability" above for more information on write routing.

`yes`
force mutable mode. Write routing directories will be created in `/var/lib/extensions.mutable/` if not present.

`import`
immutable mode, but with contents of write routing directories in `/var/lib/extensions.mutable/` also merged into the host file system.

`ephemeral`
force mutable mode, but with contents of write routing directories in `/var/lib/extensions.mutable/` being ignored, and modifications of the host file system being discarded after unmerge.

`ephemeral-import`
force mutable mode, with contents of write routing directories in `/var/lib/extensions.mutable/` being merged into the host file system, but with the modifications made to the host file system being discarded after unmerge.

`help`
list known values and exit immediately.

`--noexec=BOOL`
When merging configuration extensions into `/etc/` the `MS_NOEXEC` mount flag is used by default. This option can be used to disable it.

`--no-reload`
When used with `merge`, `unmerge` or `refresh`, do not reload daemon after executing the changes even if an extension that is applied requires a reload via the `EXTENSION_RELOAD_MANAGER=` set to 1. This also skips processing any units listed via `EXTENSION_RESTART_UNITS=` or `EXTENSION_RELOAD_OR_RESTART_UNITS=`.

Exit status

On success, 0 is returned.

See Also

`systemd(1)`, `sysext.conf(5)`, `systemd-nspawn(1)`, `systemd-stub(7)`, `importctl(1)`

□□□ #1
□ □□□ □□□ 2026-10-31 15:02:55 UTC
□ □□□ □□□ 2026-10-31 15:02:55 UTC