Rocky Enterprise Linux 9.2 Manual Pages on command 'systemd-sysext.service.8'

$ man systemd-sysext.service.8

SYSTEMD-SYSEXT(8) systemd-sysext SYSTEMD-SYSEXT(8)

NAME

systemd-sysext, systemd-sysext.service - Activates System Extension Images

SYNOPSIS

systemd-sysext [OPTIONS...] COMMAND

systemd-sysext.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, and the host /usr/ and /opt/ hierarchies become read-only too while they are activated.

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 Discoverable Partitions Specification[1]
3. Disk images lacking a partition table, with a naked Linux file system (e.g. 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.

System extensions are automatically looked for in the directories /etc/extensions/, /run/extensions/, /var/lib/extensions/, /usr/lib/extensions/ and /usr/local/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.

During boot OS extension images are activated automatically, if the systemd-sysext.service is enabled. Note that this service runs only after the underlying file systems where system extensions may be located have been mounted. This means they are not suitable for shipping resources that are processed by subsystems running in earliest boot. Specifically, OS extension images are not suitable for shipping system services or systemd-sysusers(8) definitions. See Portable Services[2] for a simple mechanism for shipping system services in disk images, in a similar fashion to OS extensions. Note the different isolation on these two mechanisms: while system extension directly extend the underlying OS image with additional files that appear in a way very similar to 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. The systemd-sysext.service service is guaranteed to finish start-up before basic.target is reached; i.e. at the time regular services initialize (those which do not use DefaultDependencies=no), the files and directories system extensions provide are available in /usr/ and /opt/ and may be accessed.

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.

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). 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.

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 if the underlying host /usr/ is managed as immutable disk image or is a

traditional package manager controlled (i.e. writable) tree.

COMMANDS

The following commands are understood:

status

When invoked without any command verb, or when status is specified the current merge status is shown, separately for both /usr/ and /opt/.

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.

unmerge

Unmerges all currently installed system extension images from /usr/ and /opt/, 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.

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

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, but below some specified root directory.

--force

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

--no-pager

Do not pipe output into a pager.

--no-legend

Do not print the legend, i.e. column headers and the footer with hints.

--json=MODE

Shows output formatted as JSON. Expects one of "short" (for the shortest possible output without any redundant whitespace or line breaks), "pretty" (for a pretty version of the same, with indentation and line breaks) or "off" (to turn off JSON output, the default).

EXIT STATUS

On success, 0 is returned.

SEE ALSO

systemd(1), systemd-nspawn(1)

NOTES

1. Discoverable Partitions Specification

   https://systemd.io/DISCOVERABLE_PARTITIONS

2. Portable Services

   https://systemd.io/PORTABLE_SERVICES