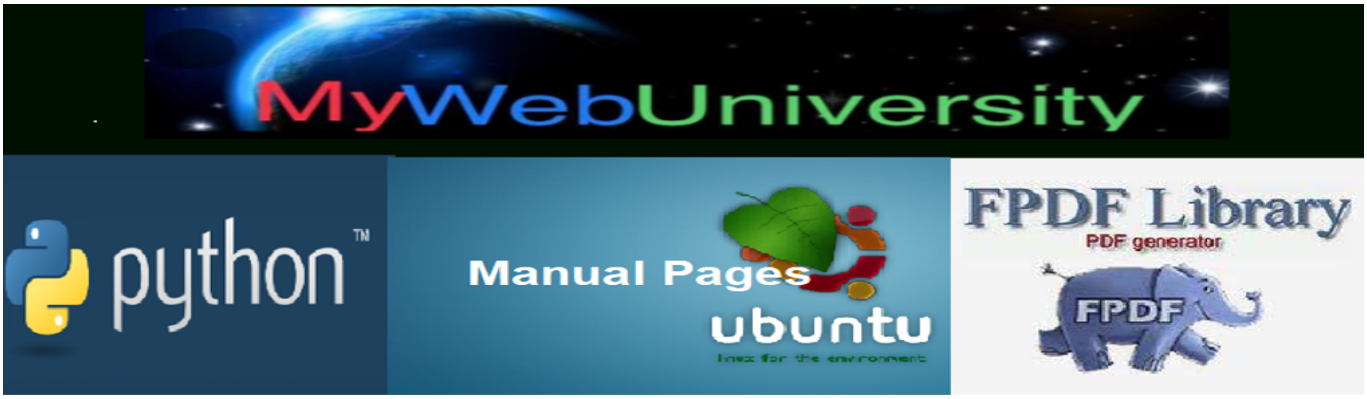




## *Rocky Enterprise Linux 9.2 Manual Pages on command 'systemd.generator.7'*

***$ man systemd.generator.7***

SYSTEMD.GENERATOR(7) systemd.generator SYSTEMD.GENERATOR(7)

NAME

systemd.generator - systemd unit generators

SYNOPSIS

/path/to/generator normal-dir early-dir late-dir

/run/systemd/system-generators/*

/etc/systemd/system-generators/*

/usr/local/lib/systemd/system-generators/*

/lib/systemd/system-generators/*

/run/systemd/user-generators/*

/etc/systemd/user-generators/*

/usr/local/lib/systemd/user-generators/*

/usr/lib/systemd/user-generators/*

DESCRIPTION

Generators are small executables placed in /lib/systemd/system-generators/ and other directories listed above. systemd(1) will execute these binaries very early at bootup and at configuration reload time ? before unit files are loaded. Their main purpose is to convert configuration that is not native to the service manager into dynamically generated unit files, symlinks or unit file drop-ins, so that they can extend the unit file hierarchy the service manager subsequently loads and operates on.

Each generator is called with three directory paths that are to be used for generator output. In these three directories, generators may dynamically generate unit files (regular ones, instances, as well as templates), unit file .d/ drop-ins, and create

symbolic links to unit files to add additional dependencies, create aliases, or instantiate existing templates. Those directories are included in the unit load path of systemd(1), allowing generated configuration to extend or override existing definitions. Directory paths for generator output differ by priority: .../generator.early has priority higher than the admin configuration in /etc/, while .../generator has lower priority than /etc/ but higher than vendor configuration in /usr/, and .../generator.late has priority lower than all other configuration. See the next section and the discussion of unit load paths and unit overriding in systemd.unit(5).

Generators are loaded from a set of paths determined during compilation, as listed above. System and user generators are loaded from directories with names ending in system-generators/ and user-generators/, respectively. Generators found in directories listed earlier override the ones with the same name in directories lower in the list. A symlink to /dev/null or an empty file can be used to mask a generator, thereby preventing it from running. Please note that the order of the two directories with the highest priority is reversed with respect to the unit load path, and generators in /run/ overwrite those in /etc/.

After installing new generators or updating the configuration, systemctl daemon-reload may be executed. This will delete the previous configuration created by generators, re-run all generators, and cause systemd to reload units from disk. See systemctl(1) for more information.

OUTPUT DIRECTORIES

Generators are invoked with three arguments: paths to directories where generators can place their generated unit files or symlinks. By default those paths are runtime directories that are included in the search path of systemd, but a generator may be called with different paths for debugging purposes.

1. normal-dir

   In normal use this is /run/systemd/generator in case of the system generators and $XDG_RUNTIME_DIR/generator in case of the user generators. Unit files placed in this directory take precedence over vendor unit configuration but not over native user/administrator unit configuration.

2. early-dir

   In normal use this is /run/systemd/generator.early in case of the system generators and $XDG_RUNTIME_DIR/generator.early in case of the user generators. Unit files placed

in this directory override unit files in /usr/, /run/ and /etc/. This means that unit files placed in this directory take precedence over all normal configuration, both vendor and user/administrator.

3. late-dir

In normal use this is /run/systemd/generator.late in case of the system generators and $XDG_RUNTIME_DIR/generator.late in case of the user generators. This directory may be used to extend the unit file tree without overriding any other unit files. Any native configuration files supplied by the vendor or user/administrator take precedence.

NOTES ABOUT WRITING GENERATORS

- All generators are executed in parallel. That means all executables are started at the very same time and need to be able to cope with this parallelism.
- Generators are run very early at boot and cannot rely on any external services. They may not talk to any other process. That includes simple things such as logging to syslog(3), or systemd itself (this means: no systemctl(1))! Non-essential file systems like /var/ and /home/ are mounted after generators have run. Generators can however rely on the most basic kernel functionality to be available, as well as mounted /sys/, /proc/, /dev/, /usr/ and /run/ file systems.
- Units written by generators are removed when the configuration is reloaded. That means the lifetime of the generated units is closely bound to the reload cycles of systemd itself.
- Generators should only be used to generate unit files, .d/*.conf drop-ins for them and symlinks to them, not any other kind of non-unit related configuration. Due to the lifecycle logic mentioned above, generators are not a good fit to generate dynamic configuration for other services. If you need to generate dynamic configuration for other services, do so in normal services you order before the service in question. Note that using the StandardInputData=/StandardInputText= settings of service unit files (see systemd.exec(5)), it is possible to make arbitrary input data (including daemon-specific configuration) part of the unit definitions, which often might be sufficient to embed data or configuration for other programs into unit files in a native fashion.
- Since syslog(3) is not available (see above), log messages have to be written to /dev/kmsg instead.
- The generator should always include its own name in a comment at the top of the

generated file, so that the user can easily figure out which component created or amended a particular unit.

The SourcePath= directive should be used in generated files to specify the source configuration file they are generated from. This makes things more easily understood by the user and also has the benefit that systemd can warn the user about configuration files that changed on disk but have not been read yet by systemd. The SourcePath= value does not have to be a file in a physical filesystem. For example, in the common case of the generator looking at the kernel command line, SourcePath=/proc/cmdline should be used.

? Generators may write out dynamic unit files or just hook unit files into other units with the usual .wants/ or .requires/ symlinks. Often, it is nicer to simply instantiate a template unit file from /usr/ with a generator instead of writing out entirely dynamic unit files. Of course, this works only if a single parameter is to be used.

? If you are careful, you can implement generators in shell scripts. We do recommend C code however, since generators are executed synchronously and hence delay the entire boot if they are slow.

? Regarding overriding semantics: there are two rules we try to follow when thinking about the overriding semantics:

1. User configuration should override vendor configuration. This (mostly) means that stuff from /etc/ should override stuff from /usr/.
2. Native configuration should override non-native configuration. This (mostly) means that stuff you generate should never override native unit files for the same purpose.

Of these two rules the first rule is probably the more important one and breaks the second one sometimes. Hence, when deciding whether to use argv[1], argv[2], or argv[3], your default choice should probably be argv[1].

? Instead of heading off now and writing all kind of generators for legacy configuration file formats, please think twice! It is often a better idea to just deprecate old stuff instead of keeping it artificially alive.

EXAMPLES

Example 1. systemd-fstab-generator

systemd-fstab-generator(8) converts /etc/fstab into native mount units. It uses argv[1] as

location to place the generated unit files in order to allow the user to override /etc/fstab with their own native unit files, but also to ensure that /etc/fstab overrides any vendor default from /usr/.

After editing /etc/fstab, the user should invoke systemctl daemon-reload. This will re-run all generators and cause systemd to reload units from disk. To actually mount new directories added to fstab, systemctl start /path/to/mountpoint or systemctl start local-fs.target may be used.

Example 2. systemd-system-update-generator

systemd-system-update-generator(8) temporarily redirects default.target to system-update.target, if a system update is scheduled. Since this needs to override the default user configuration for default.target, it uses argv[2]. For details about this logic, see systemd.offline-updates(7).

Example 3. Debugging a generator

```
dir=$(mktemp -d)
SYSTEMD_LOG_LEVEL=debug /lib/systemd/system-generators/systemd-fstab-generator \
        "$dir" "$dir" "$dir"
find $dir
```

## SEE ALSO

systemd(1), systemd-cryptsetup-generator(8), systemd-debug-generator(8), systemd-fstab-generator(8), fstab(5), systemd-getty-generator(8), systemd-gpt-auto-generator(8), systemd-hibernate-resume-generator(8), systemd-rc-local-generator(8), systemd-system-update-generator(8), systemd-sysv-generator(8), systemd-xdg-autostart-generator(8), systemd.unit(5), systemctl(1), systemd.environment-generator(7)