os-release

- Description
  - /etc/initrd-release
  - /usr/lib/extension-release.d/extension-release.IMAGE
- Options
  - General information identifying the operating system
  - Information about the version of the operating system
  - Presentation information and links
  - Distribution-level defaults and metadata
  - Notes
- Examples
- See Also

systemd

os-release

5

os-release

initrd-release

extension-release

Operating system identification

`/etc/os-release`, `/usr/lib/os-release`, `/etc/initrd-release`, `/usr/lib/extension-release.d/extension-release.IMAGE`

Description

The `/etc/os-release` and `/usr/lib/os-release` files contain operating system identification data.

The format of `os-release` is a newline-separated list of environment-like shell-compatible variable assignments. It is possible to source the configuration from Bourne shell scripts, however, beyond mere variable assignments, no shell features are supported (this means variable expansion is explicitly not supported), allowing applications to read the file without implementing a shell compatible execution engine. Variable assignment values must be enclosed in double or single quotes if they include spaces, semicolons or other special characters outside of A–Z, a–z, 0–9. (Assignments that do not include these special characters may be enclosed in quotes too, but this is optional.) Shell special characters ("$", quotes, backslash, backtick) must be escaped with backslashes, following shell style. All strings should be in UTF-8 encoding, and non-printable characters should not be used. Concatenation of multiple individually quoted strings is not supported. Lines beginning with "#" are treated as comments. Blank lines are permitted and ignored.

The file `/etc/os-release` takes precedence over `/usr/lib/os-release`. Applications should check for the former, and exclusively use its data if it exists, and only fall back to `/usr/lib/os-release` if that is missing. Applications should not combine the data from both files. `/usr/lib/os-release` is the recommended place to store OS release information as part of vendor trees. `/etc/os-release` should be a relative symlink to `/usr/lib/os-release`, to provide compatibility with applications only looking at `/etc/`. A relative symlink instead of an absolute symlink is necessary to avoid breaking the link in a chroot or initrd environment.

`os-release` contains data that is defined by the operating system vendor and should generally not be changed by the administrator.

As this file only encodes names and identifiers it should not be localized.

The `/etc/os-release` and `/usr/lib/os-release` files might be symlinks to other files, but it is important that the file is available from earliest boot on, and hence must be located on the root file system.

`os-release` must not contain repeating keys. Nevertheless, readers should pick the entries later in the file in case of repeats, similarly to how a shell sourcing the file would. A reader may warn about repeating entries.

For a longer rationale for `os-release` please refer to the Announcement of `/etc/os-release`.

`/etc/initrd-release`

In the initrd and exitrd, `/etc/initrd-release` plays the same role as `os-release` in the main system. Additionally, the presence of that file means that the system is in the initrd/exitrd phase. `/etc/os-release` should be symlinked to `/etc/initrd-release` (or vice versa), so programs that only look for `/etc/os-release` (as described above) work correctly.

The rest of this document that talks about `os-release` should be understood to apply to `initrd-release` too.

`/usr/lib/extension-release.d/extension-release.IMAGE`

`/usr/lib/extension-release.d/extension-release.IMAGE` plays the same role for extension images as `os-release` for the main system, and follows the syntax and rules as described in the Portable Services page. The purpose of this file is to identify the extension and to allow the operating system to verify that the extension image matches the base OS. This is typically implemented by checking that the extension `ID=` option either matches the host `ID=` option or is included the host `ID_LIKE=` option, and either `SYSEXT_LEVEL=` exists and matches too, or if it is not present, `VERSION_ID=` exists and matches. If the extension image sets `IMAGE_ID=` and/or `IMAGE_VERSION=`, the host must also declare these fields and the values must match. This ensures ABI/API compatibility between the layers and prevents merging of an incompatible image in an overlay.

In order to identify the extension image itself, the same fields defined below can be added to the `extension-release` file with a `SYSEXT_` prefix (to disambiguate from fields used to match on the base image). E.g.: `SYSEXT_ID=myext`, `SYSEXT_VERSION_ID=1.2.3`.

In the `extension-release.IMAGE` filename, the <IMAGE> part must exactly match the file name of the containing image with the suffix removed. In case it is not possible to guarantee that an image file name is stable and does not change between the build and the deployment phases, it is possible to relax this check: if exactly one file whose name matches `extension-release.*` is present in this directory, and the file is tagged with a `user.extension-release.strict` `xattr(7)` set to the string `0`, it will be used instead.

The rest of this document that talks about `os-release` should be understood to apply to `extension-release` too.

Options

The following OS identifications parameters may be set using `os-release`:

General information identifying the operating system

`NAME=`
A string identifying the operating system, without a version component, and suitable for presentation to the user. If not set, a default of `NAME=Linux` may be used.

Examples: `NAME=Fedora`, `NAME="Debian GNU/Linux"`.

`ID=`
A lower-case string (no spaces or other characters outside of 0–9, a–z, ".", "_" and "-") identifying the operating system, excluding any version information and suitable for processing by scripts or usage in generated filenames. If not set, a default of `ID=linux` may be used. Note that even though this string may not include characters that require shell quoting, quoting may nevertheless be used.

Examples: `ID=fedora`, `ID=debian`.

`ID_LIKE=`
A space-separated list of operating system identifiers in the same syntax as the `ID=` setting. It should list identifiers of operating systems that are closely related to the local operating system in regards to packaging and programming interfaces, for example listing one or more OS identifiers the local OS is a derivative from. An OS should generally only list other OS identifiers it itself is a derivative of, and not any OSes that are derived from it, though symmetric relationships are possible. Build scripts and similar should check this variable if they need to identify the local operating system and the value of `ID=` is not recognized. Operating systems should be listed in order of how closely the local operating system relates to the listed ones, starting with the closest. This field is optional.

Examples: for an operating system with `ID=centos`, an assignment of `ID_LIKE="rhel fedora"` would be appropriate. For an operating system with `ID=ubuntu`, an assignment of `ID_LIKE=debian` is appropriate.

`PRETTY_NAME=`
A pretty operating system name in a format suitable for presentation to the user. May or may not contain a release code name or OS version of some kind, as suitable. If not set, a default of `PRETTY_NAME="Linux"` may be used.

Example: `PRETTY_NAME="Fedora 17 (Beefy Miracle)"`

`FANCY_NAME=`
Similar to `PRETTY_NAME=`, but may contain ANSI sequences and fancy UTF-8 characters such as emojis. If defined this shall preferably used when displaying the OS name on modern terminal

emulators. If the terminal emulator does not support emojis, `PRETTY_NAME=` shall be shown instead, possibly with `ANSI_COLOR=` coloring. Use `\033` to encode the ESC character and `\\` to encode the backslash character.

Unlike `PRETTY_NAME=` this field must not contain information that is present in other fields, in particular the version (already specified in `VERSION=`) or the codename (already specified in `VERSION_CODENAME=`).

Example: `FANCY_NAME="□□\033[31mTomato\033[0;1mOS\033[0m"`

`CPE_NAME=`
A CPE name for the operating system, in URI binding syntax, following the Common Platform Enumeration Specification as proposed by the NIST. This field is optional.

Example: `CPE_NAME="cpe:/o:fedoraproject:fedora:17"`

`VARIANT=`
A string identifying a specific variant or edition of the operating system suitable for presentation to the user. This field may be used to inform the user that the configuration of this system is subject to a specific divergent set of rules or default configuration settings. This field is optional and may not be implemented on all systems.

Examples: `VARIANT="Server Edition"`, `VARIANT="Smart Refrigerator Edition"`.

Note: this field is for display purposes only. The `VARIANT_ID` field should be used for making programmatic decisions.

`VARIANT_ID=`
A lower-case string (no spaces or other characters outside of 0–9, a–z, ".", "_" and "-"), identifying a specific variant or edition of the operating system. This may be interpreted by other packages in order to determine a divergent default configuration. This field is optional and may not be implemented on all systems.

Examples: `VARIANT_ID=server`, `VARIANT_ID=embedded`.

Information about the version of the operating system

`VERSION=`
A string identifying the operating system version, excluding any OS name information, possibly including a release code name, and suitable for presentation to the user. This field is optional.

Examples: `VERSION=17`, `VERSION="17 (Beefy Miracle)"`.

`VERSION_ID=`
A string (lower-case recommended, no spaces or other characters outside of 0–9, a–z, A-Z, ".", "-", "~", "^", "+" and "_") identifying the operating system version, excluding any OS name information or release code name, and suitable for processing by scripts or usage in generated filenames. This field is optional.

It is recommended to follow the UAPI.10 Version Format Specification for this version string, but this is generally not enforced.

Examples: `VERSION_ID=17`, `VERSION_ID=11.04`.

`VERSION_CODENAME=`
A string (lower-case recommended, no spaces or other characters outside of 0–9, a–z, A-Z, ".", "-", "~", "^", "+", and "_") identifying the operating system release code name, excluding any OS name information or release version, and suitable for processing by scripts or usage in generated filenames. This field is optional and may not be implemented on all systems.

Examples: `VERSION_CODENAME=buster`, `VERSION_CODENAME=xenial`.

`BUILD_ID=`
A string uniquely identifying the system image originally used as the installation base. In most cases, `VERSION_ID` or `IMAGE_ID`+`IMAGE_VERSION` are updated when the entire system image is replaced during an update. `BUILD_ID` may be used in distributions where the original installation image version is important: `VERSION_ID` would change during incremental system updates, but `BUILD_ID` would not. This field is optional.

Examples: `BUILD_ID="2013-03-20.3"`, `BUILD_ID=201303203`.

`IMAGE_ID=`
A lower-case string (no spaces or other characters outside of 0–9, a–z, ".", "_" and "-"), identifying a specific image of the operating system. This is supposed to be used for environments where OS images are prepared, built, shipped and updated as comprehensive, consistent OS images. This field is optional and may not be implemented on all systems, in particularly not on those that are not managed via images but put together and updated from individual packages and on the local system.

When used in an `extension-release` file, this field is matched against the host `os-release`. If the extension declares `IMAGE_ID=`, the host must also set `IMAGE_ID=` with the same value, or the extension is rejected. To describe the extension image itself rather than constrain the host, use the prefixed variant (e.g. `SYSEXT_IMAGE_ID=`) instead.

Examples: `IMAGE_ID=vendorx-cashier-system`, `IMAGE_ID=netbook-image`.

`IMAGE_VERSION=`
A string (lower-case recommended, no spaces or other characters outside of 0–9, a–z, A-Z, ".", "-",

"~", "^", "+" and "_") identifying the OS image version. This is supposed to be used together with `IMAGE_ID` described above, to discern different versions of the same image.

It is recommended to follow the UAPI.10 Version Format Specification for this version string, but this is generally not enforced.

When used in an `extension-release` file, this field is matched against the host `os-release`, analogous to `IMAGE_ID=` described above.

Examples: `IMAGE_VERSION=33`, `IMAGE_VERSION=47.1rc1`.

`RELEASE_TYPE=`
A lower-case string (no spaces or other characters outside of 0-9, a-z, ".", "_", and "-"), describing what kind of release this version of the OS is. Known values follow:

- `stable` is for normal releases of the system, suitable for production use. Generally, stable releases become end-of-life soon after the next major stable release is out, although this might not be the case if, for example, a distribution adopts a rolling release model and still be production ready. Examples include Fedora 40, Ubuntu 23.10, OpenSUSE Tumbleweed, and Arch Linux.
- `lts` is for long term support releases of the system, suitable for production use and supported for an extended period of time. Generally, LTS releases continue to receive support even if newer major releases of the distribution are available. Examples include Ubuntu 24.04, Debian 12 Bookworm, and RHEL 9.4.
- `development` is for unstable versions of the system, unsuitable for production use, such as alpha, beta, or rolling unstable releases. Examples include Fedora Rawhide, Debian Testing, Fedora 40 Beta, and GNOME OS Nightly.
- `experiment` is for experimental builds of the system, created specifically to test some work-in-progress feature. This is meant to be used in combination with `EXPERIMENT=`.

If unset, or an unknown value, assume that the release is `stable`.

Examples: `RELEASE_TYPE=development`, `RELEASE_TYPE=lts`.

To summarize: if the image updates are built and shipped as comprehensive units, `IMAGE_ID`+ `IMAGE_VERSION` is the best fit. Otherwise, if updates eventually completely replace previously installed contents, as in a typical binary distribution, `VERSION_ID` should be used to identify major releases of the operating system. `BUILD_ID` may be used instead or in addition to `VERSION_ID` when the original system image version is important.

Presentation information and links

`HOME_URL=`; `DOCUMENTATION_URL=`; `SUPPORT_URL=`; `BUG_REPORT_URL=`; `PRIVACY_POLICY_URL=`
Links to resources on the Internet related to the operating system. `HOME_URL=` should refer to the

homepage of the operating system, or alternatively some homepage of the specific version of the operating system. `DOCUMENTATION_URL=` should refer to the main documentation page for this operating system. `SUPPORT_URL=` should refer to the main support page for the operating system, if there is any. This is primarily intended for operating systems which vendors provide support for. `BUG_REPORT_URL=` should refer to the main bug reporting page for the operating system, if there is any. This is primarily intended for operating systems that rely on community QA. `PRIVACY_POLICY_URL=` should refer to the main privacy policy page for the operating system, if there is any. These settings are optional, and providing only some of these settings is common. These URLs are intended to be exposed in "About this system" UIs behind links with captions such as "About this Operating System", "Obtain Support", "Report a Bug", or "Privacy Policy". The values should be in RFC3986 format, and should be `http:` or `https:` URLs, and possibly `mailto:` or `tel:`. Only one URL shall be listed in each setting. If multiple resources need to be referenced, it is recommended to provide an online landing page linking all available resources.

Examples: `HOME_URL="https://fedoraproject.org/"`, `BUG_REPORT_URL="https://bugzilla.redhat.com/"`.

`SUPPORT_END=`
The date at which support for this version of the OS ends. (What exactly "lack of support" means varies between vendors, but generally users should assume that updates, including security fixes, will not be provided.) The value is a date in the ISO 8601 format `YYYY-MM-DD`, and specifies the first day on which support *is not* provided.

For example, `SUPPORT_END=2001-01-01` means that the system was supported until the end of the last day of the previous millennium.

`LOGO=`
A string, specifying the name of an icon as defined by freedesktop.org Icon Theme Specification. This can be used by graphical applications to display an operating system's or distributor's logo. This field is optional and may not necessarily be implemented on all systems.

Examples: `LOGO=fedora-logo`, `LOGO=distributor-logo-opensuse`

`ANSI_COLOR=`
A suggested presentation (foreground) text color when showing the OS name on the console. This should be specified as string suitable for inclusion in the ESC [m ANSI/ECMA-48 escape code for setting graphical rendition. This field is optional.

Examples: `ANSI_COLOR="0;31"` for red, `ANSI_COLOR="1;34"` for light blue, or `ANSI_COLOR="0;38;2;60;110;180"` for Fedora blue.

`ANSI_COLOR_REVERSE=`
Similar to `ANSI_COLOR=`, but should encode the desired presentation color as background color, along with a suitable foreground color. This is may be used by console applications to set off "chrome" UI elements from the main terminal contents. This field is optional.

`VENDOR_NAME=`
The name of the OS vendor. This is the name of the organization or company which produces the OS. This field is optional.

This name is intended to be exposed in "About this system" UIs or software update UIs when needed to distinguish the OS vendor from the OS itself. It is intended to be human-readable.

Examples: `VENDOR_NAME="Fedora Project"` for Fedora Linux, `VENDOR_NAME="Canonical"` for Ubuntu.

`VENDOR_URL=`
The homepage of the OS vendor. This field is optional. The `VENDOR_NAME=` field should be set if this one is, although clients must be robust against either field not being set.

The value should be in RFC3986 format, and should be `http:` or `https:` URLs. Only one URL shall be listed in the setting.

Examples: `VENDOR_URL="https://fedoraproject.org/"`, `VENDOR_URL="https://canonical.com/"`.

`EXPERIMENT=`
A human-presentable description of what makes this build of the OS experimental. This field is optional. The `RELEASE_TYPE` field should be set to `experiment` if this field is set, otherwise clients should ignore this field.

This description is intended to be exposed at system installation time, or in "About this system" UIs, to warn the user that they're installing/running an experimental build of the OS. If `RELEASE_TYPE` is `experiment` but this field is unset, the UI should still warn the user, but it will be unable to explain what exactly is experimental about the current build of the OS.

Examples: `EXPERIMENT="Switch to DNF5"` for an experimental build of Fedora Linux made to test DNF5, `EXPERIMENT="Port to Apple M3 chip"` for experimental builds of Asahi Linux ported to the Apple M3 SoC, `EXPERIMENT="Mutter !1441: Dynamic triple/double buffering (v4)"` for builds of GNOME OS created by Mutter's CI for merge request !1441.

`EXPERIMENT_URL=`
The main informational page about what makes the current OS build experimental, where users can learn more about the experiment's status and potentially leave feedback. This field is optional. The `EXPERIMENT=` field should be set if this one is, although clients must be robust against either field not being set.

The value should be in RFC3986 format, and should be `http:` or `https:` URLs. Only one URL shall be listed in the setting.

Examples, corresponding to the examples above in `EXPERIMENT=`:
`EXPERIMENT_URL="https://fedoraproject.org/wiki/Changes/SwitchToDnf5"`,
`EXPERIMENT_URL="https://github.com/AsahiLinux/docs/wiki/M3-Series-Feature-Support"`,
`EXPERIMENT_URL="https://gitlab.gnome.org/GNOME/mutter/-/merge_requests/1441"`

.

Distribution-level defaults and metadata

`DEFAULT_HOSTNAME=`
A string specifying the hostname if `hostname(5)` is not present and no other configuration source specifies the hostname. Must be either a single DNS label (a string composed of 7-bit ASCII lower-case characters and no spaces or dots, limited to the format allowed for DNS domain name labels), or a sequence of such labels separated by single dots that forms a valid DNS FQDN. The hostname must be at most 64 characters, which is a Linux limitation (DNS allows longer names).

See `org.freedesktop.hostname1(5)` for a description of how `systemd-hostnamed.service(8)` determines the fallback hostname.

`ARCHITECTURE=`
A string that specifies which CPU architecture the userspace binaries require. The architecture identifiers are the same as for `ConditionArchitecture=` described in `systemd.unit(5)`. The field is optional and should only be used when just single architecture is supported. It may provide redundant information when used in a GPT partition with a GUID type that already encodes the architecture. If this is not the case, the architecture should be specified in e.g., an extension image, to prevent an incompatible host from loading it.

`SYSEXT_LEVEL=`
A lower-case string (mostly numeric, no spaces or other characters outside of 0–9, a–z, ".", "_" and "-") identifying the operating system extensions support level, to indicate which extension images are supported. See `/usr/lib/extension-release.d/extension-release.IMAGE`, initrd and `systemd-sysext(8)`) for more information.

Examples: `SYSEXT_LEVEL=2`, `SYSEXT_LEVEL=15.14`.

`CONFEXT_LEVEL=`
Semantically the same as `SYSEXT_LEVEL=` but for confext images. See `/etc/extension-release.d/extension-release.IMAGE` for more information.

Examples: `CONFEXT_LEVEL=2`, `CONFEXT_LEVEL=15.14`.

`SYSEXT_SCOPE=`
Takes a space-separated list of one or more of the strings `system`, `initrd` and `portable`. This field is only supported in `extension-release.d/` files and indicates what environments the system extension is applicable to: i.e. to regular systems, to initrds and exitrds, or to portable service images. If not specified, `SYSEXT_SCOPE=system portable` is implied, i.e. any system extension without

this field is applicable to regular systems and to portable service environments, but not to initrd/exitrd environments.

`CONFEXT_SCOPE=`
Semantically the same as `SYSEXT_SCOPE=` but for confext images.

`PORTABLE_PREFIXES=`

Takes a space-separated list of one or more valid prefix match strings for the Portable Services logic. This field serves two purposes: it is informational, identifying portable service images as such (and thus allowing them to be distinguished from other OS images, such as bootable system images). It is also used when a portable service image is attached: the specified or implied portable service prefix is checked against the list specified here, to enforce restrictions how images may be attached to a system.

`PORTABLE_SCOPE=`
Specifies the scope of the portable service. Takes one of `system`, `user`, or `any`. When set to `system`, the portable service can only be attached to the system instance of `systemd-portabled`. When set to `user`, the portable service can only be attached to the user instance of `systemd-portabled`. When set to `any`, the portable service can be attached to both the system and user instances of `systemd-portabled`. If not set, `PORTABLE_SCOPE=system` is implied.

Notes

If you are using this file to determine the OS or a specific version of it, use the `ID` and `VERSION_ID` fields, possibly with `ID_LIKE` as fallback for `ID`. When looking for an OS identification string for presentation to the user use the `PRETTY_NAME` field.

Note that operating system vendors may choose not to provide version information, for example to accommodate for rolling releases. In this case, `VERSION` and `VERSION_ID` may be unset. Applications should not rely on these fields to be set.

Operating system vendors may extend the file format and introduce new fields. It is highly recommended to prefix new fields with an OS-specific name in order to avoid name clashes. Applications reading this file must ignore unknown fields.

Example: `DEBIAN_BTS="debbugs://bugs.debian.org/"`.

Container and sandbox runtime managers may make the host's identification data available to applications by providing the host's `/etc/os-release` (if available, otherwise `/usr/lib/os-release` as a fallback) as `/run/host/os-release`.

Examples

`os-release` file for Fedora Workstation

```
NAME=Fedora
VERSION="32 (Workstation Edition)"
ID=fedora
VERSION_ID=32
PRETTY_NAME="Fedora 32 (Workstation Edition)"
ANSI_COLOR="0;38;2;60;110;180"
LOGO=fedora-logo-icon
CPE_NAME="cpe:/o:fedoraproject:fedora:32"
HOME_URL="https://fedoraproject.org/"
DOCUMENTATION_URL="https://docs.fedoraproject.org/en-US/fedora/f32/system-administrators-guide/"
SUPPORT_URL="https://fedoraproject.org/wiki/Communicating_and_getting_help"
BUG_REPORT_URL="https://bugzilla.redhat.com/"
REDHAT_BUGZILLA_PRODUCT="Fedora"
REDHAT_BUGZILLA_PRODUCT_VERSION=32
REDHAT_SUPPORT_PRODUCT="Fedora"
REDHAT_SUPPORT_PRODUCT_VERSION=32
PRIVACY_POLICY_URL="https://fedoraproject.org/wiki/Legal:PrivacyPolicy"
VARIANT="Workstation Edition"
VARIANT_ID=workstation
```

`extension-release` file for an extension for Fedora Workstation 32

```
ID=fedora
VERSION_ID=32
```

Reading `os-release` in `sh(1)`

Reading `os-release` in `python(1)` (versions >= 3.10)

See docs for `platform.freedesktop_os_release` for more details.

Reading `os-release` in `python(1)` (any version)

Note that the above version that uses the built-in implementation is preferred in most cases, and the open-coded version here is provided for reference.

See Also

`systemd(1)`, `lsb_release(1)`, `hostname(5)`, `machine-id(5)`, `machine-info(5)`

□□□ #1
□ □□□ □□□ 2026-10-31 14:47:56 UTC
□ □□□ □□□ 2026-10-31 14:47:56 UTC