Rocky Enterprise Linux 9.2 Manual Pages on command 'dpkg-source.1'

C:\>man dpkg-source.1

dpkg-source(1) dpkg suite dpkg-source(1)

NAME

dpkg-source - Debian source package (.dsc) manipulation tool

SYNOPSIS

dpkg-source [option...] command

DESCRIPTION

dpkg-source packs and unpacks Debian source archives.

None of these commands allow multiple options to be combined into one, and they do not allow the value for an option to be specified in a separate argument.

COMMANDS

-x, --extract filename.dsc [output-directory]

Extract a source package (--extract since dpkg 1.17.14). One non-option argument must be supplied, the name of the Debian source control file (.dsc). An optional second non-option argument may be supplied to specify the directory to extract the source package to, this must not exist. If no output directory is specified, the source package is extracted into a directory named source-version under the current working directory.

dpkg-source will read the names of the other file(s) making up the source package from the control file; they are assumed to be in the same directory as the .dsc.

The files in the extracted package will have their permissions and ownerships set to those which would have been expected if the files and

directories had simply been created - directories and executable files will be 0777 and plain files will be 0666, both modified by the extractors' umask; if the parent directory is setgid then the extracted directories will be too, and all the files and directories will inherit its group ownership.

If the source package uses a non-standard format (currently this means all formats except ?1.0?), its name will be stored in debian/source/format so that the following builds of the source package use the same format by default.

-b, --build directory [format-specific-parameters]

Build a source package (--build since dpkg 1.17.14). The first non-option argument is taken as the name of the directory containing the debianized source tree (i.e. with a debian sub-directory and maybe changes to the original files). Depending on the source package format used to build the package, additional parameters might be accepted.

dpkg-source will build the source package with the first format found in this ordered list: the format indicated with the --format command line option, the format indicated in debian/source/format, ?1.0?. The fallback to ?1.0? is deprecated and will be removed at some point in the future, you should always document the desired source format in debian/source/format.

See section SOURCE PACKAGE FORMATS for an extensive description of the various source package formats.

--print-format directory

Print the source format that would be used to build the source package if dpkg-source --build directory was called (in the same conditions and with the same parameters; since dpkg 1.15.5).

--before-build directory

Run the corresponding hook of the source package format (since dpkg 1.15.8). This hook is called before any build of the package (dpkg-buildpackage calls it very early even before debian/rules clean). This command is idempotent and can be called multiple times. Not all source formats implement something in this hook, and those that do usually prepare the source tree for the build for example by ensuring that the Debian patches are applied.

--after-build directory

Run the corresponding hook of the source package format (since dpkg 1.15.8). This hook is called after any build of the package (dpkg-buildpackage calls it last). This command is idempotent and can be called multiple times. Not all source formats implement something in this hook, and those that do usually use it to undo what --before-build has done.

--commit [directory] ...

Record changes in the source tree unpacked in directory (since dpkg 1.16.1). This command can take supplementary parameters depending on the source format. It will error out for formats where this operation doesn't mean anything.

-?, --help

Show the usage message and exit. The format specific build and extract options can be shown by using the --format option.

--version

Show the version and exit.

OPTIONS

Generic build options

-ccontrol-file

Specifies the main source control file to read information from. The default is debian/control. If given with relative pathname this is interpreted starting at the source tree's top level directory.

-lchangelog-file

Specifies the changelog file to read information from. The default is debian/changelog. If given with relative pathname this is interpreted starting at the source tree's top level directory.

-Fchangelog-format

Specifies the format of the changelog. See dpkg-parsechangelog(1) for information about alternative formats.

--format=value

Use the given format for building the source package (since dpkg 1.14.17). It does override any format given in debian/source/format.

-Vname=value

Set an output substitution variable. See deb-substvars(5) for a discussion

of output substitution.

-Tsubstvars-file

Read substitution variables in substvars-file; the default is to not read any file. This option can be used multiple times to read substitution variables from multiple files (since dpkg 1.15.6).

-Dfield=value

Override or add an output control file field.

-Ufield

Remove an output control file field.

-Zcompression, --compression=compression

Specify the compression to use for created tarballs and diff files (--compression since dpkg 1.15.5). Note that this option will not cause existing tarballs to be recompressed, it only affects new files. Supported values are: gzip, bzip2, lzma and xz. The default is xz for formats 2.0 and newer, and gzip for format 1.0. xz is only supported since dpkg 1.15.5.

-zlevel, --compression-level=level

Compression level to use (--compression-level since dpkg 1.15.5). As with -Z it only affects newly created files. Supported values are: 1 to 9, best, and fast. The default is 9 for gzip and bzip2, 6 for xz and lzma.

-i[regex], --diff-ignore[=regex]

You may specify a perl regular expression to match files you want filtered out of the list of files for the diff (--diff-ignore since dpkg 1.15.6). (This list is generated by a find command.) (If the source package is being built as a version 3 source package using a VCS, this can be used to ignore uncommitted changes on specific files. Using -i.* will ignore all of them.)

The -i option by itself enables this setting with a default regex (preserving any modification to the default regex done by a previous use of --extend-diff-ignore) that will filter out control files and directories of the most common revision control systems, backup and swap files and Libtool build output directories. There can only be one active regex, of multiple -i options only the last one will take effect.

This is very helpful in cutting out extraneous files that get included in the diff, e.g. if you maintain your source in a revision control system and

want to use a checkout to build a source package without including the additional files and directories that it will usually contain (e.g. CVS/, .cvsignore, .svn/). The default regex is already very exhaustive, but if you need to replace it, please note that by default it can match any part of a path, so if you want to match the begin of a filename or only full filenames, you will need to provide the necessary anchors (e.g. ?(^|/)?, ?($|/)?) yourself.

--extend-diff-ignore=regex

The perl regular expression specified will extend the default value used by --diff-ignore and its current value, if set (since dpkg 1.15.6). It does this by concatenating ?|regex? to the existing value. This option is convenient to use in debian/source/options to exclude some auto-generated files from the automatic patch generation.

-I[file-pattern], --tar-ignore[=file-pattern]

If this option is specified, the pattern will be passed to tar(1)'s --exclude option when it is called to generate a .orig.tar or .tar file (--tar-ignore since dpkg 1.15.6). For example, -ICVS will make tar skip over CVS directories when generating a .tar.gz file. The option may be repeated multiple times to list multiple patterns to exclude.

-I by itself adds default --exclude options that will filter out control files and directories of the most common revision control systems, backup and swap files and Libtool build output directories.

Note: While they have similar purposes, -i and -I have very different syntax and semantics. -i can only be specified once and takes a perl compatible regular expression which is matched against the full relative path of each file. -I can specified multiple times and takes a filename pattern with shell wildcards. The pattern is applied to the full relative path but also to each part of the path individually. The exact semantic of tar's --exclude option is somewhat complicated, see https://www.gnu.org/software/tar/manual/tar.html#wildcards for a full documentation.

The default regex and patterns for both options can be seen in the output of the --help command.

Generic extract options

--no-copy

Do not copy original tarballs near the extracted source package (since dpkg 1.14.17).

--no-check

Do not check signatures and checksums before unpacking (since dpkg 1.14.17).

--no-overwrite-dir

Do not overwrite the extraction directory if it already exists (since dpkg 1.18.8).

--require-valid-signature

Refuse to unpack the source package if it doesn't contain an OpenPGP signature that can be verified (since dpkg 1.15.0) either with the user's trustedkeys.gpg keyring, one of the vendor-specific keyrings, or one of the official Debian keyrings (/usr/share/keyrings/debian-keyring.gpg and /usr/share/keyrings/debian-maintainers.gpg).

--require-strong-checksums

Refuse to unpack the source package if it does not contain any strong checksums (since dpkg 1.18.7). Currently the only known checksum considered strong is SHA-256.

--ignore-bad-version

Turns the bad source package version check into a non-fatal warning (since dpkg 1.17.7). This option should only be necessary when extracting ancient source packages with broken versions, just for backwards compatibility.

SOURCE PACKAGE FORMATS

If you don't know what source format to use, you should probably pick either ?3.0 (quilt)? or ?3.0 (native)?. See https://wiki.debian.org/Projects/DebSrc3.0 for information on the deployment of those formats within Debian.

Format: 1.0

A source package in this format consists either of a .orig.tar.gz associated to a .diff.gz or a single .tar.gz (in that case the package is said to be native). Optionally the original tarball might be accompanied by a detached upstream signature .orig.tar.gz.asc, extraction supported since dpkg 1.18.5.

Extracting

Extracting a native package is a simple extraction of the single tarball in the

target directory. Extracting a non-native package is done by first unpacking the .orig.tar.gz and then applying the patch contained in the .diff.gz file. The timestamp of all patched files is reset to the extraction time of the source package (this avoids timestamp skews leading to problems when autogenerated files are patched). The diff can create new files (the whole debian directory is created that way) but can't remove files (empty files will be left over).

Building

Building a native package is just creating a single tarball with the source directory. Building a non-native package involves extracting the original tarball in a separate ?.orig? directory and regenerating the .diff.gz by comparing the source package directory with the .orig directory.

Build options (with --build):

If a second non-option argument is supplied it should be the name of the original source directory or tarfile or the empty string if the package is a Debian-specific one and so has no debianization diffs. If no second argument is supplied then dpkg-source will look for the original source tarfile package_upstream-version.orig.tar.gz or the original source directory directory.orig depending on the -sX arguments.

-sa, -sp, -sk, -su and -sr will not overwrite existing tarfiles or directories. If this is desired then -sA, -sP, -sK, -sU and -sR should be used instead.

-sk Specifies to expect the original source as a tarfile, by default package_upstream-version.orig.tar.extension. It will leave this original source in place as a tarfile, or copy it to the current directory if it isn't already there. The tarball will be unpacked into directory.orig for the generation of the diff.

-sp Like -sk but will remove the directory again afterwards.

-su Specifies that the original source is expected as a directory, by default package-upstream-version.orig and dpkg-source will create a new original source archive from it.

-sr Like -su but will remove that directory after it has been used.

-ss Specifies that the original source is available both as a directory and as a tarfile. dpkg-source will use the directory to create the diff, but the tarfile to create the .dsc. This option must be used with care - if the

directory and tarfile do not match a bad source archive will be generated.

-sn Specifies to not look for any original source, and to not generate a diff. The second argument, if supplied, must be the empty string. This is used for Debian-specific packages which do not have a separate upstream source and therefore have no debianization diffs.

-sa or -sA

Specifies to look for the original source archive as a tarfile or as a directory - the second argument, if any, may be either, or the empty string (this is equivalent to using -sn). If a tarfile is found it will unpack it to create the diff and remove it afterwards (this is equivalent to -sp); if a directory is found it will pack it to create the original source and remove it afterwards (this is equivalent to -sr); if neither is found it will assume that the package has no debianization diffs, only a straightforward source archive (this is equivalent to -sn). If both are found then dpkg-source will ignore the directory, overwriting it, if -sA was specified (this is equivalent to -sP) or raise an error if -sa was specified. -sa is the default.

--abort-on-upstream-changes

The process fails if the generated diff contains changes to files outside of the debian sub-directory (since dpkg 1.15.8). This option is not allowed in debian/source/options but can be used in debian/source/local-options.

Extract options (with --extract):

In all cases any existing original source tree will be removed.

-sp Used when extracting then the original source (if any) will be left as a tarfile. If it is not already located in the current directory or if an existing but different file is there it will be copied there. (This is the default).

-su Unpacks the original source tree.

-sn Ensures that the original source is neither copied to the current directory nor unpacked. Any original source tree that was in the current directory is still removed.

All the -sX options are mutually exclusive. If you specify more than one only the last one will be used.

--skip-debianization

Skips application of the debian diff on top of the upstream sources (since dpkg 1.15.1).

Format: 2.0

Extraction supported since dpkg 1.13.9, building supported since dpkg 1.14.8. Also known as wig&pen. This format is not recommended for wide-spread usage, the format ?3.0 (quilt)? replaces it. Wig&pen was the first specification of a new-generation source package format.

The behaviour of this format is the same as the ?3.0 (quilt)? format except that it doesn't use an explicit list of patches. All files in debian/patches/ matching the perl regular expression [\w-]+ must be valid patches: they are applied at extraction time.

When building a new source package, any change to the upstream source is stored in a patch named zz_debian-diff-auto.

Format: 3.0 (native)

Supported since dpkg 1.14.17. This format is an extension of the native package format as defined in the 1.0 format. It supports all compression methods and will ignore by default any VCS specific files and directories as well as many temporary files (see default value associated to -I option in the --help output).

Format: 3.0 (quilt)

Supported since dpkg 1.14.17. A source package in this format contains at least an original tarball (.orig.tar.ext where ext can be gz, bz2, lzma and xz) and a debian tarball (.debian.tar.ext). It can also contain additional original tarballs (.orig-component.tar.ext). component can only contain alphanumeric (?a-zA-Z0-9?) characters and hyphens (?-?). Optionally each original tarball can be accompanied by a detached upstream signature (.orig.tar.ext.asc and .orig-component.tar.ext.asc), extraction supported since dpkg 1.17.20, building supported since dpkg 1.18.5.

Extracting

The main original tarball is extracted first, then all additional original tarballs are extracted in subdirectories named after the component part of their filename (any pre-existing directory is replaced). The debian tarball is extracted on top of the source directory after prior removal of any pre-existing debian directory. Note

that the debian tarball must contain a debian sub-directory but it can also contain binary files outside of that directory (see --include-binaries option).

All patches listed in debian/patches/vendor.series or debian/patches/series are then applied, where vendor will be the lowercase name of the current vendor, or debian if there is no vendor defined. If the former file is used and the latter one doesn't exist (or is a symlink), then the latter is replaced with a symlink to the former. This is meant to simplify usage of quilt to manage the set of patches. Vendor-specific series files are intended to make it possible to serialize multiple development branches based on the vendor, in a declarative way, in preference to open-coding this handling in debian/rules. This is particularly useful when the source would need to be patched conditionally because the affected files do not have built-in conditional occlusion support. Note however that while dpkg-source parses correctly series files with explicit options used for patch application (stored on each line after the patch filename and one or more spaces), it does ignore those options and always expect patches that can be applied with the -p1 option of patch. It will thus emit a warning when it encounters such options, and the build is likely to fail.

Note that lintian(1) will emit unconditional warnings when using vendor series due to a controversial Debian specific ruling, which should not affect any external usage; to silence these, the dpkg lintian profile can be used by passing ?--profile dpkg? to lintian(1).

The timestamp of all patched files is reset to the extraction time of the source package (this avoids timestamp skews leading to problems when autogenerated files are patched).

Contrary to quilt's default behaviour, patches are expected to apply without any fuzz. When that is not the case, you should refresh such patches with quilt, or dpkg-source will error out while trying to apply them.

Similarly to quilt's default behaviour, the patches can remove files too.

The file .pc/applied-patches is created if some patches have been applied during the extraction.

Building

All original tarballs found in the current directory are extracted in a temporary directory by following the same logic as for the unpack, the debian directory is

copied over in the temporary directory, and all patches except the automatic patch (debian-changes-version or debian-changes, depending on --single-debian-patch) are applied. The temporary directory is compared to the source package directory. When the diff is non-empty, the build fails unless --single-debian-patch or --auto-commit has been used, in which case the diff is stored in the automatic patch. If the automatic patch is created/deleted, it's added/removed from the series file and from the quilt metadata.

Any change on a binary file is not representable in a diff and will thus lead to a failure unless the maintainer deliberately decided to include that modified binary file in the debian tarball (by listing it in debian/source/include-binaries). The build will also fail if it finds binary files in the debian sub-directory unless they have been whitelisted through debian/source/include-binaries.

The updated debian directory and the list of modified binaries is then used to generate the debian tarball.

The automatically generated diff doesn't include changes on VCS specific files as well as many temporary files (see default value associated to -i option in the --help output). In particular, the .pc directory used by quilt is ignored during generation of the automatic patch.

Note: dpkg-source --before-build (and --build) will ensure that all patches listed in the series file are applied so that a package build always has all patches applied. It does this by finding unapplied patches (they are listed in the series file but not in .pc/applied-patches), and if the first patch in that set can be applied without errors, it will apply them all. The option --no-preparation can be used to disable this behavior.

Recording changes

--commit [directory] [patch-name] [patch-file]

Generates a patch corresponding to the local changes that are not managed by the quilt patch system and integrates it in the patch system under the name patch-name. If the name is missing, it will be asked interactively. If patch-file is given, it is used as the patch corresponding to the local changes to integrate. Once integrated, an editor is launched so that you can edit the meta-information in the patch header.

Passing patch-file is mainly useful after a build failure that pre-generated

this file, and on this ground the given file is removed after integration.

Note also that the changes contained in the patch file must already be applied on the tree and that the files modified by the patch must not have supplementary unrecorded changes.

If the patch generation detects modified binary files, they will be automatically added to debian/source/include-binaries so that they end up in the debian tarball (exactly like dpkg-source --include-binaries --build would do).

Build options

--allow-version-of-quilt-db=version

Allow dpkg-source to build the source package if the version of the quilt metadata is the one specified, even if dpkg-source doesn't know about it (since dpkg 1.15.5.4). Effectively this says that the given version of the quilt metadata is compatible with the version 2 that dpkg-source currently supports. The version of the quilt metadata is stored in .pc/.version.

--include-removal

Do not ignore removed files and include them in the automatically generated patch.

--include-timestamp

Include timestamp in the automatically generated patch.

--include-binaries

Add all modified binaries in the debian tarball. Also add them to debian/source/include-binaries: they will be added by default in subsequent builds and this option is thus no more needed.

--no-preparation

Do not try to prepare the build tree by applying patches which are apparently unapplied (since dpkg 1.14.18).

--single-debian-patch

Use debian/patches/debian-changes instead of debian/patches/debian-changes-version for the name of the automatic patch generated during build (since dpkg 1.15.5.4). This option is particularly useful when the package is maintained in a VCS and a patch set can't reliably be generated. Instead the current diff with upstream should be

stored in a single patch. The option would be put in debian/source/local-options and would be accompanied by a debian/source/local-patch-header file explaining how the Debian changes can be best reviewed, for example in the VCS that is used.

--create-empty-orig

Automatically create the main original tarball as empty if it's missing and if there are supplementary original tarballs (since dpkg 1.15.6). This option is meant to be used when the source package is just a bundle of multiple upstream software and where there's no ?main? software.

--no-unapply-patches, --unapply-patches

By default, dpkg-source will automatically unapply the patches in the --after-build hook if it did apply them during --before-build (--unapply-patches since dpkg 1.15.8, --no-unapply-patches since dpkg 1.16.5). Those options allow you to forcefully disable or enable the patch unapplication process. Those options are only allowed in debian/source/local-options so that all generated source packages have the same behavior by default.

--abort-on-upstream-changes

The process fails if an automatic patch has been generated (since dpkg 1.15.8). This option can be used to ensure that all changes were properly recorded in separate quilt patches prior to the source package build. This option is not allowed in debian/source/options but can be used in debian/source/local-options.

--auto-commit

The process doesn't fail if an automatic patch has been generated, instead it's immediately recorded in the quilt series.

Extract options

--skip-debianization

Skips extraction of the debian tarball on top of the upstream sources (since dpkg 1.15.1).

--skip-patches

Do not apply patches at the end of the extraction (since dpkg 1.14.18).

Supported since dpkg 1.14.17. This format is special. It doesn't represent a real source package format but can be used to create source packages with arbitrary files.

Build options

All non-option arguments are taken as files to integrate in the generated source package. They must exist and are preferably in the current directory. At least one file must be given.

--target-format=value

Required. Defines the real format of the generated source package. The generated .dsc file will contain this value in its Format field and not ?3.0 (custom)?.

Format: 3.0 (git)

Supported since dpkg 1.14.17. This format is experimental.

A source package in this format consists of a single bundle of a git repository .git to hold the source of a package. There may also be a .gitshallow file listing revisions for a shallow git clone.

Extracting

The bundle is cloned as a git repository to the target directory. If there is a gitshallow file, it is installed as .git/shallow inside the cloned git repository.

Note that by default the new repository will have the same branch checked out that was checked out in the original source. (Typically ?master?, but it could be anything.) Any other branches will be available under remotes/origin/.

Building

Before going any further, some checks are done to ensure that we don't have any non-ignored uncommitted changes.

git-bundle(1) is used to generate a bundle of the git repository. By default, all branches and tags in the repository are included in the bundle.

Build options

--git-ref=ref

Allows specifying a git ref to include in the git bundle. Use disables the default behavior of including all branches and tags. May be specified multiple times. The ref can be the name of a branch or tag to include. It may also be any parameter that can be passed to git-rev-list(1). For

example, to include only the master branch, use --git-ref=master. To include all tags and branches, except for the private branch, use --git-ref=--all --git-ref=^private

--git-depth=number

Creates a shallow clone with a history truncated to the specified number of revisions.

Format: 3.0 (bzr)

Supported since dpkg 1.14.17. This format is experimental. It generates a single tarball containing the bzr repository.

Extracting

The tarball is unpacked and then bzr is used to checkout the current branch.

Building

Before going any further, some checks are done to ensure that we don't have any non-ignored uncommitted changes.

Then the VCS specific part of the source directory is copied over to a temporary directory. Before this temporary directory is packed in a tarball, various cleanup are done to save space.

DIAGNOSTICS

no source format specified in debian/source/format

The file debian/source/format should always exist and indicate the desired source format. For backwards compatibility, format ?1.0? is assumed when the file doesn't exist but you should not rely on this: at some point in the future dpkg-source will be modified to fail when that file doesn't exist.

The rationale is that format ?1.0? is no longer the recommended format, you should usually pick one of the newer formats (?3.0 (quilt)?, ?3.0 (native)?) but dpkg-source will not do this automatically for you. If you want to continue using the old format, you should be explicit about it and put ?1.0? in debian/source/format.

the diff modifies the following upstream files

When using source format ?1.0? it is usually a bad idea to modify upstream files directly as the changes end up hidden and mostly undocumented in the .diff.gz file. Instead you should store your changes as patches in the debian directory and apply them at build-time. To avoid this complexity you can also use the format ?3.0

(quilt)? that offers this natively.

cannot represent change to file

Changes to upstream sources are usually stored with patch files, but not all changes can be represented with patches: they can only alter the content of plain text files. If you try replacing a file with something of a different type (for example replacing a plain file with a symlink or a directory), you will get this error message.

newly created empty file file will not be represented in diff

Empty files can't be created with patch files. Thus this change is not recorded in the source package and you are warned about it.

executable mode perms of file will not be represented in diff

Patch files do not record permissions of files and thus executable permissions are not stored in the source package. This warning reminds you of that fact.

special mode perms of file will not be represented in diff

Patch files do not record permissions of files and thus modified permissions are not stored in the source package. This warning reminds you of that fact.

ENVIRONMENT

DPKG_COLORS

Sets the color mode (since dpkg 1.18.5). The currently accepted values are: auto (default), always and never.

DPKG_NLS

If set, it will be used to decide whether to activate Native Language Support, also known as internationalization (or i18n) support (since dpkg 1.19.0). The accepted values are: 0 and 1 (default).

SOURCE_DATE_EPOCH

If set, it will be used as the timestamp (as seconds since the epoch) to clamp the mtime in the tar(5) file entries.

VISUAL

EDITOR Used by the ?2.0? and ?3.0 (quilt)? source format modules.

GIT_DIR

GIT_INDEX_FILE

GIT_OBJECT_DIRECTORY

GIT_ALTERNATE_OBJECT_DIRECTORIES

GIT_WORK_TREE

Used by the ?3.0 (git)? source format modules.

FILES

debian/source/format

This file contains on a single line the format that should be used to build the source package (possible formats are described above). No leading or trailing spaces are allowed.

debian/source/include-binaries

This file contains a list of binary files (one per line) that should be included in the debian tarball. Leading and trailing spaces are stripped. Lines starting with ?#? are comments and are skipped. Empty lines are ignored.

debian/source/options

This file contains a list of long options that should be automatically prepended to the set of command line options of a dpkg-source --build or dpkg-source --print-format call. Options like --compression and --compression-level are well suited for this file.

Each option should be put on a separate line. Empty lines and lines starting with ?#? are ignored. The leading ?--? should be stripped and short options are not allowed. Optional spaces are allowed around the ?=? symbol and optional quotes are allowed around the value. Here's an example of such a file:

```
# let dpkg-source create a debian.tar.bz2 with maximal compression
compression = "bzip2"
compression-level = 9
# use debian/patches/debian-changes as automatic patch
single-debian-patch
# ignore changes on config.{sub,guess}
extend-diff-ignore = "(^|/)(config.sub|config.guess)$"
```

Note: format options are not accepted in this file, you should use debian/source/format instead.

debian/source/local-options

Exactly like debian/source/options except that the file is not included in the generated source package. It can be useful to store a preference tied to the maintainer or to the VCS repository where the source package is maintained.

debian/source/local-patch-header and debian/source/patch-header

Free form text that is put on top of the automatic patch generated in formats ?2.0? or ?3.0 (quilt)?. local-patch-header is not included in the generated source package while patch-header is.

debian/patches/vendor.series

debian/patches/series

This file lists all patches that have to be applied (in the given order) on top of the upstream source package. Leading and trailing spaces are stripped. The vendor will be the lowercase name of the current vendor, or debian if there is no vendor defined. If the vendor-specific series file does not exist, the vendor-less series file will be used. Lines starting with ?#? are comments and are skipped. Empty lines are ignored. Remaining lines start with a patch filename (relative to the debian/patches/ directory) up to the first space character or the end of line. Optional quilt options can follow up to the end of line or the first ?#? preceded by one or more spaces (which marks the start of a comment up to the end of line).

BUGS

The point at which field overriding occurs compared to certain standard output field settings is rather confused.

SEE ALSO

deb-src-control(5), deb-changelog(5), dsc(5).

1.19.7 2022-05-25 dpkg-source(1)