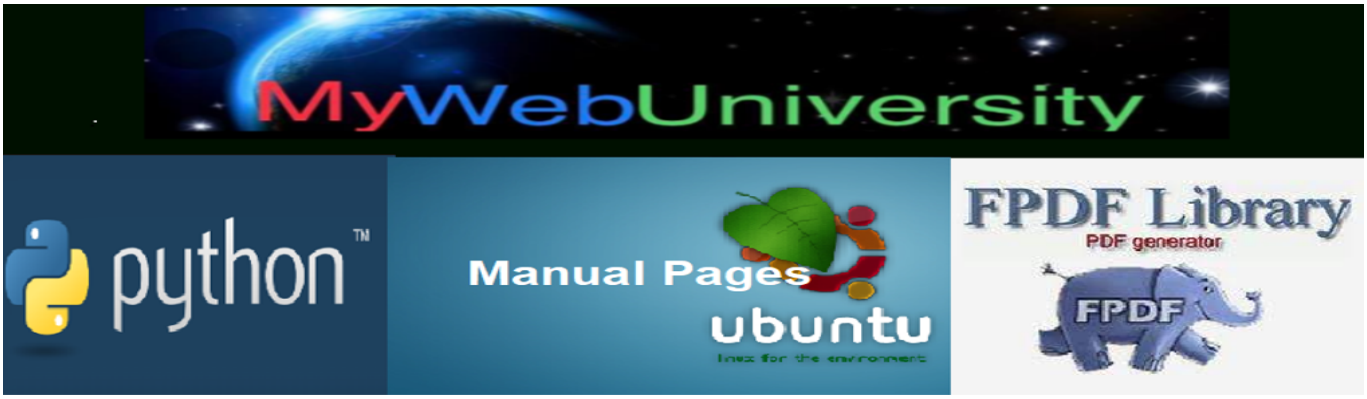

Rocky Enterprise Linux 9.2 Manual Pages on command 'deb-src-control.5'

$ man deb-src-control.5

deb-src-control(5) dpkg suite deb-src-control(5)

NAME

deb-src-control - Debian source packages' master control file format

SYNOPSIS

debian/control

DESCRIPTION

Each Debian source package contains the master ?debian/control? file, and its deb822(5) format is a superset of the control file shipped in Debian binary packages, see deb-control(5).

This file contains at least 2 paragraphs, separated by a blank line. The first paragraph lists all information about the source package in general, while each following paragraph describes exactly one binary package. Each paragraph consists of at least one field. A field starts with a fieldname, such as Package or Section (case insensitive), followed by a colon, the body of the field (case sensitive unless stated otherwise) and a newline. Multi-line fields are also allowed, but each supplementary line, without a fieldname, should start with at least one space. The content of the multi-line fields is generally joined to a single line by the tools (except in the case of the Description field, see below). To insert empty lines into a multi-line field, insert a dot after the space.

Lines starting with a ?#? are treated as comments.

SOURCE FIELDS

Source: source-package-name (required)

The value of this field is the name of the source package, and should match the name of the source package in the debian/changelog file. A package name must consist only

of lowercase letters (a-z), digits (0-9), plus (+) and minus (-) signs, and periods (.). Package names must be at least two characters long and must start with a lowercase alphanumeric character (a-z0-9).

Maintainer: fullname-email (recommended)

Should be in the format ?Joe Bloggs <jbloggs@foo.com>?, and references the person who currently maintains the package, as opposed to the author of the software or the original packager.

Uploaders: fullname-email

Lists all the names and email addresses of co-maintainers of the package, in the same format as the Maintainer field. Multiple co-maintainers should be separated by a comma.

Standards-Version: version-string

This documents the most recent version of the distribution policy standards this package complies with.

Description short-description

long-description

The format for the source package description is a short brief summary on the first line (after the Description field). The following lines should be used as a longer, more detailed description. Each line of the long description must be preceded by a space, and blank lines in the long description must contain a single ?.? following the preceding space.

Homepage: url

The upstream project home page URL.

Bugs: url

The url of the bug tracking system for this package. The current used format is bts-type://bts-address, like debbugs://bugs.debian.org. This field is usually not needed.

Rules-Requires-Root: no|binary-targets|impl-keywords

This field is used to indicate whether the debian/rules file requires (fake)root privileges to run some of its targets, and if so when.

no The binary targets will not require (fake)root at all.

binary-targets

The binary targets must always be run under (fake)root. This value is the default when the field is omitted; adding the field with an explicit binary-targets while

not strictly needed, marks it as having been analyzed for this requirement.

impl-keywords

This is a space-separated list of keywords which define when (fake)root is required.

Keywords consist of namespace/cases. The namespace part cannot contain "/" or whitespace. The cases part cannot contain whitespace. Furthermore, both parts must consist entirely of printable ASCII characters.

Each tool/package will define a namespace named after itself and provide a number of cases where (fake)root is required. (See "Implementation provided keywords" in rootless-builds.txt).

When the field is set to one of the impl-keywords, the builder will expose an interface that is used to run a command under (fake)root. (See "Gain Root API" in rootless-builds.txt.)

Testsuite: name-list

Testsuite-Triggers: package-list

These fields are described in the dsc(5) manual page, as they are generated from information inferred from debian/tests/control or copied literally to the source control file.

Vcs-Arch: url

Vcs-Bzr: url

Vcs-Cvs: url

Vcs-Darcs: url

Vcs-Git: url

Vcs-Hg: url

Vcs-Mtn: url

Vcs-Svn: url

The url of the Version Control System repository used to maintain this package. Currently supported are Arch, Bzr (Bazaar), Cvs, Darcs, Git, Hg (Mercurial), Mtn (Monotone) and Svn (Subversion). Usually this field points to the latest version of the package, such as the main branch or the trunk.

Vcs-Browser: url

The url of a webinterface to browse the Version Control System repository.

Origin: name

The name of the distribution this package is originating from. This field is usually not needed.

Section: section

This is a general field that gives the package a category based on the software that it installs. Some common sections are utils, net, mail, text, x11, etc.

Priority: priority

Sets the importance of this package in relation to the system as a whole. Common priorities are required, standard, optional, extra, etc.

The Section and Priority fields usually have a defined set of accepted values based on the specific distribution policy.

Build-Depends: package-list

A list of packages that need to be installed and configured to be able to build from source package. These dependencies need to be satisfied when building binary architecture dependent or independent packages and source packages. Including a dependency in this field does not have the exact same effect as including it in both Build-Depends-Arch and Build-Depends-Indep, because the dependency also needs to be satisfied when building the source package.

Build-Depends-Arch: package-list

Same as Build-Depends, but they are only needed when building the architecture dependent packages. The Build-Depends are also installed in this case. This field is supported since dpkg 1.16.4; in order to build with older dpkg versions, Build-Depends should be used instead.

Build-Depends-Indep: package-list

Same as Build-Depends, but they are only needed when building the architecture independent packages. The Build-Depends are also installed in this case.

Build-Conflicts: package-list

A list of packages that should not be installed when the package is built, for example because they interfere with the build system used. Including a dependency in this list has the same effect as including it in both Build-Conflicts-Arch and Build-Conflicts-Indep, with the additional effect of being used for source-only builds.

Build-Conflicts-Arch: package-list

Same as Build-Conflicts, but only when building the architecture dependent packages. This field is supported since dpkg 1.16.4; in order to build with older dpkg versions,

Build-Conflicts should be used instead.

Build-Conflicts-Indep: package-list

Same as Build-Conflicts, but only when building the architecture independent packages.

The syntax of the Build-Depends, Build-Depends-Arch and Build-Depends-Indep fields is a list of groups of alternative packages. Each group is a list of packages separated by vertical bar (or ?pipe?) symbols, ?|?. The groups are separated by commas ?,?, and can end with a trailing comma that will be eliminated when generating the fields for deb-control(5) (since dpkg 1.10.14). Commas are to be read as ?AND?, and pipes as ?OR?, with pipes binding more tightly. Each package name is optionally followed by an architecture qualifier appended after a colon ?:?, optionally followed by a version number specification in parentheses ?(? and ?)?, an architecture specification in square brackets ?[? and ?]?, and a restriction formula consisting of one or more lists of profile names in angle brackets ?<? and ?>?.

The syntax of the Build-Conflicts, Build-Conflicts-Arch and Build-Conflicts-Indep fields is a list of comma-separated package names, where the comma is read as an ?AND?, and where the list can end with a trailing comma that will be eliminated when generating the fields for deb-control(5) (since dpkg 1.10.14). Specifying alternative packages using a ?pipe? is not supported. Each package name is optionally followed by a version number specification in parentheses, an architecture specification in square brackets, and a restriction formula consisting of one or more lists of profile names in angle brackets.

An architecture qualifier name can be a real Debian architecture name (since dpkg 1.16.5), any (since dpkg 1.16.2) or native (since dpkg 1.16.5). If omitted, the default for Build-Depends fields is the current host architecture, the default for Build-Conflicts fields is any. A real Debian architecture name will match exactly that architecture for that package name, any will match any architecture for that package name if the package is marked with Multi-Arch: allowed, and native will match the current build architecture if the package is not marked with Multi-Arch: foreign.

A version number may start with a ?>>?, in which case any later version will match, and may specify or omit the Debian packaging revision (separated by a hyphen). Accepted version relationships are ?>>? for greater than, ?<<? for less than, ?>=? for greater than or equal to, ?<=? for less than or equal to, and ?=? for equal to.

An architecture specification consists of one or more architecture names, separated by whitespace. Exclamation marks may be prepended to each of the names, meaning ?NOT?.

A restriction formula consists of one or more restriction lists, separated by whitespace. Each restriction list is enclosed in angle brackets. Items in the restriction list are build profile names, separated by whitespace and can be prefixed with an exclamation mark, meaning ?NOT?. A restriction formula represents a disjunctive normal form expression. Note that dependencies on packages in the build-essential set can be omitted and that declaring build conflicts against them is impossible. A list of these packages is in the build-essential package.

BINARY FIELDS

Note that the Priority, Section and Homepage fields can also be in a binary paragraph to override the global value from the source package.

Package: binary-package-name (required)

This field is used to name the binary package name. The same restrictions as to a source package name apply.

Package-Type: deb|udeb|type

This field defines the type of the package. udeb is for size-constrained packages used by the debian installer. deb is the default value, it is assumed if the field is absent. More types might be added in the future.

Architecture: arch|all|any (required)

The architecture specifies on which type of hardware this package runs. For packages that run on all architectures, use the any value. For packages that are architecture independent, such as shell and Perl scripts or documentation, use the all value. To restrict the packages to a certain set of architectures, specify the architecture names, separated by a space. It's also possible to put architecture wildcards in that list (see dpkg-architecture(1) for more information about them).

Build-Profiles: restriction-formula

This field specifies the conditions for which this binary package does or does not build. To express that condition, the same restriction formula syntax from the Build-Depends field is used.

If a binary package paragraph does not contain this field, then it implicitly means that it builds with all build profiles (including none at all).

In other words, if a binary package paragraph is annotated with a non-empty Build-Profiles field, then this binary package is generated if and only if the condition expressed by the conjunctive normal form expression evaluates to true.

Protected: Byes|no

Essential: yes|no

Build-Essential: yes|no

Multi-Arch: same|foreign|allowed|no

Tag: tag-list

Description: short-description (recommended)

These fields are described in the deb-control(5) manual page, as they are copied literally to the control file of the binary package.

Depends: package-list

Pre-Depends: package-list

Recommends: package-list

Suggests: package-list

Breaks: package-list

Enhances: package-list

Replaces: package-list

Conflicts: package-list

Provides: package-list

Built-Using: package-list

These fields declare relationships between packages. They are discussed in the deb-control(5) manpage. When these fields are found in debian/control they can also end with a trailing comma (since dpkg 1.10.14), have architecture specifications and restriction formulas which will all get reduced when generating the fields for deb-control(5).

Subarchitecture: value

Kernel-Version: value

Installer-Menu-Item: value

These fields are used by the debian-installer in udebs and are usually not needed. See /usr/share/doc/debian-installer/devel/modules.txt from the debian-installer package for more details about them.

USER-DEFINED FIELDS

It is allowed to add additional user-defined fields to the control file. The tools will ignore these fields. If you want the fields to be copied over to the output files, such as the binary packages, you need to use a custom naming scheme: the fields should start with

an X, followed by zero or more of the letters SBC and a hyphen.

S The field will appear in the source package control file, see dsc(5).

B The field will appear in the control file in the binary package, see deb-control(5).

C The field will appear in the upload control (.changes) file, see deb-changes(5).

Note that the X[SBC]- prefixes are stripped when the fields are copied over to the output files. A field XC-Approved-By will appear as Approved-By in the changes file and will not appear in the binary or source package control files.

Take into account that these user-defined fields will be using the global namespace, which might at some point in the future collide with officially recognized fields. To avoid such potential situation you can prefix those fields with Private-, such as XB-Private-New-Field.

EXAMPLE

```
# Comment
Source: dpkg
Section: admin
Priority: required
Maintainer: Dpkg Developers <debian-dpkg@lists.debian.org>
# this field is copied to the binary and source packages
XBS-Upstream-Release-Status: stable
Homepage: https://wiki.debian.org/Teams/Dpkg
Vcs-Browser: https://git.dpkg.org/cgit/dpkg/dpkg.git
Vcs-Git: https://git.dpkg.org/git/dpkg/dpkg.git
Standards-Version: 3.7.3
Build-Depends: pkg-config, debhelper (>= 4.1.81),
 libselinux1-dev (>= 1.28-4) [!linux-any]
Package: dpkg-dev
Section: utils
Priority: optional
Architecture: all
# this is a custom field in the binary package
XB-Mentoring-Contact: Raphael Hertzog <hertzog@debian.org>
Depends: dpkg (>= 1.14.6), perl5, perl-modules, cpio (>= 2.4.2-2),
 bzip2, lzma, patch (>= 2.2-1), make, binutils, libtimedate-perl
```

Recommends: gcc | c-compiler, build-essential

Suggests: gnupg, debian-keyring

Conflicts: dpkg-cross (<< 2.0.0), devscripts (<< 2.10.26)

Replaces: manpages-pl (<= 20051117-1)

Description: Debian package development tools

This package provides the development tools (including dpkg-source)

required to unpack, build and upload Debian source packages.

.

Most Debian source packages will require additional tools to build;

for example, most packages need make and the C compiler gcc.

SEE ALSO

deb822(5), deb-control(5), deb-version(7), dpkg-source(1)

1.21.1 2024-02-23 deb-src-control(5)