



Rocky Enterprise Linux 9.2 Manual Pages on command 'dh_python3.1'

C:\>man dh_python3.1

DH_PYTHON3(1)

DH_PYTHON3(1)

NAME

dh_python3 - calculates Python dependencies, adds maintainer scripts to byte com?
pile files, etc.

SYNOPSIS

dh_python3 -p PACKAGE [-V [X.Y][-[A.B]] DIR [-X REGEXPR]

DESCRIPTION

QUICK GUIDE FOR MAINTAINERS

- ? if necessary, describe supported Python 3 versions via X-Python3-Version field in debian/control,
- ? build depend on dh-python
- ? build-depend on python3 or python3-all or python3-all-dev,
- ? build module/application using its standard build system, remember to build extensions for all supported Python 3 versions (loop over py3versions -vr),
- ? install files to the standard locations, add --install-layout=deb to setup.py's install command if your package is using distutils,
- ? add python3 to dh's --with option, or:
- ? include /usr/share/cdb/1/class/python-distutils.mk in debian/rules and depend on cdb (>= 0.4.90), or:
- ? call dh_python3 in the binary-* target,
- ? add \${python3:Depends} to Depends

NOTES

dependencies

`dh_python3` tries to translate Python dependencies from the `requires.txt` file to Debian dependencies. In many cases, this works without any additional configuration because `dh_python3` comes with a build-in mapping of Python module names to Debian packages that is periodically regenerated from the Debian archive. By default, the version information in the Python dependencies is discarded. If you want `dh_python3` to generate more strict dependencies (e.g. to avoid ABI problems), or if the automatic mapping does not work correctly for your package, you have to provide `dh_python3` with additional rules for the translation of Python module to Debian package dependencies.

For a package `python3-foo` that depends on a package `python3-bar`, there are two files that may provide such rules:

1. If the `python3-foo` source package ships with a `debian/py3dist-overrides` file, this file is used by `dh_python3` during the build of `python3-foo`.
2. If the `python3-bar` source package ships with a `debian/python3-bar.pydist` file (and uses `dh_python3`), this file will be included in the binary package as `/usr/share/dh-python/dist/cpython3/python3-bar`. During the build of `python3-foo`, `dh_python3` will then find and use the file.

Both files have the same format described in `/usr/share/doc/dh-python/README.Py?Dist`. If all you want is to generate versioned dependencies (and assuming that the `python3-bar` package provides the `pybar` Python module), in most cases it will be sufficient to put the line `pybar python3-bar; PEP386` into either of the above files.

private dirs

`/usr/share/foo`, `/usr/share/games/foo`, `/usr/lib/foo` and `/usr/lib/games/foo` private directories are scanned for Python files by default (where `foo` is binary package name). If your package ships Python files in some other directory, add another `dh_python3` call in `debian/rules` with directory name as an argument - you can use different set of options in this call. If you need to change options (f.e. a list of supported Python 3 versions) for a private directory that is checked by default, invoke `dh_python3` with `--skip-private` option and add another call with a path to this directory and new options.

debug packages

In binary packages which name ends with `-dbg`, all files in `/usr/lib/python3/dist-packages/` directory that have extensions different than `so` or `h` are removed by default. Use `--no-dbg-cleaning` option to disable this feature.

pyinstall files

Files listed in `debian/pkg.pyinstall` file will be installed as public modules (i.e. into `.../dist-packages/` directory) for all requested Python versions.

Syntax: `path/to/file [VERSION_RANGE] [NAMESPACE]`

`debian` directory is automatically removed from the path, so you can place your files in `debian/` directory and install them from this location (if you want to install them in "debian" namespace, set `NAMESPACE` to `debian`). If `NAMESPACE` is set, all listed files will be installed in `.../dist-packages/NAMESPACE/` directory.

Examples:

- ? `foo.py` installs `.../dist-packages/foo.py` for all supported Python versions
- ? `foo/bar.py 3.3-` installs `.../dist-packages/foo/bar.py` for versions `>= 3.3`
- ? `foo/bar.py spam` installs `.../dist-packages/spam/bar.py`
- ? `debian/*.py spam.egg 3.2` installs `.../python3.2/dist-packages/spam/egg/*.py` files

pyremove files

If you want to remove some public modules (i.e. files in `.../dist-packages/` directory) installed by build system (from all supported Python versions or only from a subset of these versions), add them to `debian/pkg.pyremove` file.

Examples:

- ? `*.pth` removes `.pth` files from `.../dist-packages/`
- ? `bar/baz.py 3.2` removes `.../python3.2/dist-packages/bar/baz.py`

bcep files

Byte-compilation exception patterns can be described in these files. Use it if you want `py3compile` to skip specific files. This is the only way to skip `.py` files in `.../dist-packages/` directory (as `--exclude` passed to `py3compile` in `postinst` is not used in `rtupdate` scripts and thus this option cannot be used for non-private modules).

`rel-3.6|usr/lib/python3/dist-packages/jinja2|*/async(foo|bar).py` will skip byte-compilation of `asyncfoo.py` and `asyncbar.py` in `usr/lib/python3/dist-packages/jinja2/` directory for each interpreter that doesn't support `async` keyword (in?

troduced in Python 3.6).

If you want to skip byte-compilation in a subdirectory for all interpreters, use:

`dir|-4.0|usr/lib/python3/dist-packages/foo/tests/`. `VERSION_RANGE` (-4.0 in the example) is described in `README.PyDist` file.

`debian/python3-foo.bcep` file from source package will be included in the binary package as `/usr/share/python3/bcep/python3-foo.bcep`

overriding supported / default Python versions

If you want to override system's list of supported Python versions or the default one (f.e. to build a package that includes symlinks for older version of Python or compile `.py` files only for given interpreter version), you can do that via `DEBPYTHON3_SUPPORTED` and/or `DEBPYTHON3_DEFAULT` env. variables.

Example: `3.2,3.3` limits the list of supported Python versions to Python 3.2 and Python 3.3.

OPTIONS

`--version`

show program's version number and exit

`-h, --help`

show help message and exit

`--no-guessing-deps`

disable guessing dependencies

`--no-dbg-cleaning`

do not remove any files from debug packages

`--no-ext-rename`

do not add magic tags nor multiarch tuples to extension file names

`--no-shebang-rewrite`

do not rewrite shebangs

`--skip-private`

don't check private directories

`-v, --verbose`

turn verbose mode on

`-i, --indep`

act on architecture independent packages

`-a, --arch`

act on architecture dependent packages

`-q, --quiet`

be quiet

`-p PACKAGE, --package=PACKAGE`

act on the package named PACKAGE

`-N NO_PACKAGE, --no-package=NO_PACKAGE`

do not act on the specified package

`-V VERSION_RANGE`

specify list of supported Python 3 versions. See `py3compile(1)` for examples

`-X REGEXPR, --exclude=REGEXPR`

exclude items that match given REGEXPR. You may use this option multiple

times to build up a list of things to exclude from byte-compilation in pri?

vate dirs. See also `bcep` files.

`--compile-all`

compile all files from given private directory in `postinst/rtupdate` not just

the ones provided by the package (i.e. do not pass the `--package` parameter

to `py3compile/py3clean`)

`--accept-upstream-versions`

accept upstream versions while translating Python dependencies into Debian

ones

`--depends=DEPENDS`

translate given requirements into Debian dependencies and add them to

`${python3:Depends}`. Use it for missing items in `requires.txt`

`--depends-section=SECTION`

translate requirements from given sections of `requires.txt` file into Debian

dependencies and add them to `${python3:Depends}`.

`--recommends=RECOMMENDS`

translate given requirements into Debian dependencies and add them to

`${python3:Recommends}`

`--recommends-section=SECTION`

translate requirements from given sections of `requires.txt` file into Debian

dependencies and add them to `${python3:Recommends}`.

`--suggests=SUGGESTS`

translate given requirements into Debian dependencies and add them to
\${python3:Suggests}

--suggests-section=SECTION

translate requirements from given sections of requires.txt file into Debian
dependencies and add them to \${python3:Suggests}.

--requires=FILENAME

translate requirements from given file(s) into Debian dependencies and add
them to \${python3:Depends}

--shebang=COMMAND

use given command as shebang in scripts

--ignore-shebangs

do not translate shebangs into Debian dependencies

SEE ALSO

- ? /usr/share/doc/python/python-policy.txt.gz
- ? /usr/share/doc/dh-python/README.PyDist
- ? pybuild(1)
- ? py3compile(1), py3clean(1)
- ? dh_python2(1), pycompile(1), pyclean(1)
- ? <http://deb.li/dhp3> - most recent version of this document

AUTHOR

Piotr O?arowski, 2012-2013

DH_PYTHON3(1)