

**NAME**

`dh_python2` – calculates Python dependencies, adds maintainer scripts to byte compile files, etc.

**SYNOPSIS**

`dh_python2 -p PACKAGE [-V [X.Y][-[A.B]] DIR_OR_FILE [-X REGEXPR]`

**DESCRIPTION****QUICK GUIDE FOR MAINTAINERS**

- if necessary, describe supported Python versions via `X-Python-Version` field in `debian/control`,
- `build-depend` on `python` or `python-all` or `python-all-dev` ( $\geq 2.6.6-3\sim$ ),
- build module/application using its standard build system, remember to build extensions for all supported Python versions (loop over **pyversions** `-vr`),
- install files to the *standard* locations, add `--install-layout=deb` to `setup.py`'s `install` command if your package is using `distutils`,
- add `python2` to `dh`'s `---with` option, or:
- `include /usr/share/cdb/1/class/python-distutils.mk` in `debian/rules` and depend on `cdb` ( $\geq 0.4.90$ ), or:
- call **`dh_python2`** in the `binary-*` target,
- add `${python:Depends}` to `Depends`

**NOTES**

In order to support more than one Python version in the same binary package, `dh_python2` (unlike `dh_pycentral` and `dh_pysupport`) creates symlinks to all supported Python versions at build time. It means `binNMU` (or sourceful upload in case of architecture independent packages) is required once a list of supported Python version is changed. It's faster and more robust than its competitors, though.

**dependencies**

`dh_python2` tries to translate Python dependencies from `requires.txt` file to Debian dependencies. Use `debian/pydist-overrides` or `---no-guessing-deps` option to override it if the guess is incorrect. If you want `dh_python2` to generate more strict dependencies (f.e. to avoid ABI problems) create `debian/python-foo.pydist` file. See `/usr/share/doc/python-doc/README.PyDist` (provided by `python-doc` package) for more information. If the `pydist` file contains PEP386 flag or set of (uscan like) rules, `dh_python2` will make the dependency versioned (version requirements are ignored by default).

**namespace feature**

`dh_python2` parses Egg's `namespace_packages.txt` files (in addition to `---namespace` command line argument(s)) and drops empty `__init__.py` files from binary package. `pycompile` will regenerate them at install time and `pyclean` will remove them at uninstall time (if they're no longer used in installed packages). It's still a good idea to provide `__init__.py` file in one of binary packages (even if all other packages use this feature).

**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 is shipping Python files in some other directory, add another `dh_python2` 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 versions) for a private directory that is checked by default, invoke `dh_python2` 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/python2.X/{site,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 for all requested Python versions (`dh_install` doesn't know about python's site- vs. dist-packages issue).

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 2.6-** installs `.../dist-packages/foo/bar.py` for versions `>= 2.6`
- **foo/bar.py spam** installs `.../dist-packages/spam/bar.py`
- **debian/\*.py spam.egg 2.5** installs `.../python2.5/site-packages/spam/egg/*.py` files

**pyremove files**

If you want to remove some files 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 2.5** removes `.../python2.5/site-packages/bar/baz.py`

**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 `DEBPYTHON_SUPPORTED` and/or `DEBPYTHON_DEFAULT` env. variables.

Example: **2.5,2.7** limits the list of supported Python versions to Python 2.5 and Python 2.7.

**OPTIONS**

- version**  
show program's version number and exit
- h, --help**  
show help message and exit
- no-guessing-versions**  
disable guessing other supported Python versions
- no-guessing-deps**  
disable guessing dependencies
- no-dbg-cleaning**  
do not remove any files from debug packages
- 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 VRANGE**  
specify list of supported Python versions. See pycompile(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.
- 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 pycompile/pyclean)
- depends=DEPENDS**  
translate given requirements into Debian dependencies and add them to \${python:Depends}. Use it for missing items in requires.txt
- recommends=RECOMMENDS**  
translate given requirements into Debian dependencies and add them to \${python:Recommends}
- suggests=SUGGESTS**  
translate given requirements into Debian dependencies and add them to \${python:Suggests}
- namespace**  
use this option (multiple time if necessary) if namespace\_packages.txt is not complete
- ignore-namespace**  
ignore Egg's namespace declaration and --namespace option. This option will disable removing (and recreating at install time) empty \_\_init\_\_.py files. Removing namespace\_packages.txt from egg-info directory has the same effect.
- clean-pycentral**  
generate maintainer script that will remove byte code generated by python-central helper
- 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/python-doc/README.PyDist (python-doc package)
- pycompile(1), pyclean(1)
- dh\_python3(1), py3compile(1), py3clean(1)
- Wiki page about converting package to dh\_python2:  
<http://wiki.debian.org/Python/TransitionToDHPython2>

## AUTHOR

Piotr Oarowski, 2012-2013