



## ***Linux Ubuntu 26.04 Manual Pages on command 'dpkg-maintscript-helper.1'***

### ***\$ man dpkg-maintscript-helper.1***

dpkg-maintscript-helper(1)                      dpkg suite                      dpkg-maintscript-helper(1)

#### NAME

dpkg-maintscript-helper - works around known dpkg limitations in maintainer scripts

#### SYNOPSIS

dpkg-maintscript-helper command [parameter...] -- maint-script-parameter...

#### COMMANDS AND PARAMETERS

supports command

rm\_conffile conffile [prior-version [package]]

mv\_conffile old-conffile new-conffile [prior-version [package]]

symlink\_to\_dir pathname old-target [prior-version [package]]

dir\_to\_symlink pathname new-target [prior-version [package]]

#### DESCRIPTION

This program is designed to be run within maintainer scripts to achieve some tasks that dpkg can't (yet) handle natively either because of design decisions or due to current limitations.

Many of those tasks require coordinated actions from several maintainer scripts (preinst, postinst, prerm, postrm). To avoid mistakes the same call simply needs to be put in all scripts and the program will automatically adapt its behavior based on the environment variable `DPKG_MAINTSCRIPT_NAME` and on the maintainer scripts arguments that you have to forward after a double hyphen.

This program was introduced in dpkg 1.15.7.

#### COMMON PARAMETERS

prior-version

Defines the latest version of the package whose upgrade should trigger the operation.

It is important to calculate prior-version correctly so that the operations are correctly performed even if the user rebuilt the package with a local version. If prior-version is empty or omitted, then the operation is tried on every upgrade (note: it's safer to give the version and have the operation tried only once).

If the conffile has not been shipped for several versions, and you are now modifying the maintainer scripts to clean up the obsolete file, prior-version should be based on the version of the package that you are now preparing, not the first version of the package that lacked the conffile. This applies to all other actions in the same way.

For example, for a conffile removed in version 2.0-1 of a package, prior-version should be set to 2.0-1~. This will cause the conffile to be removed even if the user rebuilt the previous version 1.0-1 as 1.0-1local1. Or a package switching a path from a symlink (shipped in version 1.0-1) to a directory (shipped in version 2.0-1), but only performing the actual switch in the maintainer scripts in version 3.0-1, should set prior-version to 3.0-1~.

#### package

The package name owning the pathname(s). When the package is ?Multi-Arch: same? this parameter must include the architecture qualifier, otherwise it should not usually include the architecture qualifier (as it would disallow cross-grades, or switching from being architecture specific to architecture all or vice versa). If the parameter is empty or omitted, the DPKG\_MAINTSCRIPT\_PACKAGE and DPKG\_MAINTSCRIPT\_ARCH environment variables (as set by dpkg when running the maintainer scripts) will be used to generate an arch-qualified package name.

-- All the parameters of the maintainer scripts have to be forwarded to the program after --.

#### CONFFILE RELATED TASKS

When upgrading a package, dpkg will not automatically remove a conffile (a configuration file for which dpkg should preserve user changes) if it is not present in the newer version.

There are two principal reasons for this; the first is that the conffile could've been dropped by accident and the next version could restore it, users wouldn't want their changes thrown away. The second is to allow packages to transition files from a dpkg-maintained conffile to a file maintained by the package's maintainer scripts, usually with a tool like debconf or ucf.

This means that if a package is intended to rename or remove a conffile, it must explicitly do so and `dpkg-maintscript-helper` can be used to implement graceful deletion and moving of conffiles within maintainer scripts.

### Removing a conffile

Note: This can be replaced in most cases by the "remove-on-upgrade" flag in `DEBIAN/conffiles` (since `dpkg` 1.20.6), see `deb-conffiles(5)`.

If a conffile is completely removed, it should be removed from disk, unless the user has modified it. If there are local modifications, they should be preserved. If the package upgrade aborts, the newly obsolete conffile should not disappear.

All of this is implemented by putting the following shell snippet in the `preinst`, `postinst` and `postrm` maintainer scripts:

```
dpkg-maintscript-helper rm_conffile \  
    conffile prior-version package -- "$@"
```

`conffile` is the filename of the conffile to remove.

Current implementation: in the `preinst`, it checks if the conffile was modified and renames it either to `conffile.dpkg-remove` (if not modified) or to `conffile.dpkg-backup` (if modified). In the `postinst`, the latter file is renamed to `conffile.dpkg-bak` and kept for reference as it contains user modifications but the former will be removed. If the package upgrade aborts, the `postrm` reinstalls the original conffile. During purge, the `postrm` will also delete the `.dpkg-bak` file kept up to now.

### Renaming a conffile

If a conffile is moved from one location to another, you need to make sure you move across any changes the user has made. This may seem a simple change to the `preinst` script at first, however that will result in the user being prompted by `dpkg` to approve the conffile edits even though they are not responsible of them.

Graceful renaming can be implemented by putting the following shell snippet in the `preinst`, `postinst` and `postrm` maintainer scripts:

```
dpkg-maintscript-helper mv_conffile \  
    old-conffile new-conffile prior-version package -- "$@"
```

`old-conffile` and `new-conffile` are the old and new name of the conffile to rename.

Current implementation: the `preinst` checks if the conffile has been modified, if yes it's left on place otherwise it's renamed to `old-conffile.dpkg-remove`. On configuration, the `postinst` removes `old-conffile.dpkg-remove` and renames `old-conffile` to `new-conffile` if old-

conffile is still available. On abort-upgrade/abort-install, the postrm renames old-conffile.dpkg-remove back to old-conffile if required.

## SYMLINK AND DIRECTORY SWITCHES

When upgrading a package, dpkg will not automatically switch a symlink to a directory or vice-versa. Downgrades are not supported and the path will be left as is.

Note: The symlinks and directories created during these switches need to be shipped in the new packages, or dpkg will not be able to remove them on purge.

### Switching a symlink to directory

If a symlink is switched to a real directory, you need to make sure before unpacking that the symlink is removed. This may seem a simple change to the preinst script at first, however that will result in some problems in case of admin local customization of the symlink or when downgrading the package.

Graceful renaming can be implemented by putting the following shell snippet in the preinst, postinst and postrm maintainer scripts:

```
dpkg-maintscript-helper symlink_to_dir \  
    pathname old-target prior-version package -- "$@"
```

pathname is the absolute name of the old symlink (the path will be a directory at the end of the installation) and old-target is the target name of the former symlink at pathname. It can either be absolute or relative to the directory containing pathname.

Current implementation: the preinst checks if the symlink exists and points to old-target, if not then it's left in place, otherwise it's renamed to pathname.dpkg-backup. On configuration, the postinst removes pathname.dpkg-backup if pathname.dpkg-backup is still a symlink. On abort-upgrade/abort-install, the postrm renames pathname.dpkg-backup back to pathname if required.

### Switching a directory to symlink

If a real directory is switched to a symlink, you need to make sure before unpacking that the directory is removed. This may seem a simple change to the preinst script at first, however that will result in some problems in case the directory contains conffiles, pathnames owned by other packages, locally created pathnames, or when downgrading the package.

Graceful switching can be implemented by putting the following shell snippet in the preinst, postinst and postrm maintainer scripts:

```
dpkg-maintscript-helper dir_to_symlink \  
    pathname old-target prior-version package -- "$@"
```

```
pathname new-target prior-version package -- "$@"
```

pathname is the absolute name of the old directory (the path will be a symlink at the end of the installation) and new-target is the target of the new symlink at pathname. It can either be absolute or relative to the directory containing pathname.

Current implementation: the preinst checks if the directory exists, does not contain conffiles, pathnames owned by other packages, or locally created pathnames, if not then it's left in place, otherwise it's renamed to pathname.dpkg-backup, and an empty staging directory named pathname is created, marked with a file so that dpkg can track it. On configuration, the postinst finishes the switch if pathname.dpkg-backup is still a directory and pathname is the staging directory; it removes the staging directory mark file, moves the newly created files inside the staging directory to the symlink target new-target/, replaces the now empty staging directory pathname with a symlink to new-target, and removes pathname.dpkg-backup. On abort-upgrade/abort-install, the postrm renames pathname.dpkg-backup back to pathname if required.

## INTEGRATION IN PACKAGES

When using a packaging helper, please check if it has native dpkg-maintscript-helper integration, which might make your life easier. See for example `dh_installdeb(1)`.

Given that dpkg-maintscript-helper is used in the preinst, using it unconditionally requires a versioned pre-dependency to ensure that the required version of dpkg has been unpacked before, unless we know we are upgrading from a release where the required dpkg version is going to be satisfied. The required version depends on the command used, for `rm_conffile` and `mv_conffile` it is 1.15.7.2, for `symlink_to_dir` and `dir_to_symlink` it is 1.17.14. For example, if we required support for `symlink_to_dir` when upgrading from a release that had a dpkg version earlier than 1.17.14, we would need to add:

```
Pre-Depends: dpkg (>= 1.17.14)
```

But in many cases the operation done by the program is not critical for the package, and instead of using a pre-dependency we can call the program only if we know that the required command is supported by the currently installed dpkg:

```
if dpkg-maintscript-helper supports command; then
```

```
    dpkg-maintscript-helper command ...
```

```
fi
```

The command supports will return 0 on success, 1 otherwise. The supports command will check if the environment variables as set by dpkg and required by the script are present, and will

consider it a failure in case the environment is not sufficient.

## ENVIRONMENT

### DPKG\_ROOT

If set, it will be used as the filesystem root directory.

### DPKG\_ADMDIR

If set, it will be used as the dpkg data directory.

### DPKG\_COLORS

Sets the color mode (since dpkg 1.19.1). The currently accepted values are: auto (default), always and never.

## SEE ALSO

[dh\\_installdeb\(1\)](#).

1.23.7

2026-03-31

[dpkg-maintscript-helper\(1\)](#)