
Debian Hamradio Maintainers Guide

Release 0.7

Debian Hamradio Maintainers

Nov 21, 2021

CONTENTS

| | | |
|----------|---|-----------|
| 1 | Introduction | 3 |
| 1.1 | Joining the team as a packager | 3 |
| 1.2 | Assisting the team with translations | 3 |
| 2 | Packaging | 5 |
| 2.1 | Newcomer guidelines | 5 |
| 2.2 | Announcing intention to package | 5 |
| 2.3 | Team Policy | 5 |
| 3 | Version Control System | 9 |
| 3.1 | Anonymous access to git repositories | 9 |
| 3.2 | Repository layout | 9 |
| 3.3 | Importing an existing source package | 9 |
| 3.4 | Importing an existing git repository | 10 |
| 3.5 | Creating a new git repository for a new package | 10 |
| 3.6 | Updating a package already in the team git | 11 |
| 3.7 | Testing package builds with git-buildpackage | 11 |
| 3.8 | Using git-buildpackage and git-pbuilder for building packages | 11 |
| 3.9 | Final build and upload | 12 |
| 3.10 | Backports and updates to stable distributions | 12 |
| 3.11 | Tracking commit notifications | 12 |
| 3.12 | Further Reading | 13 |
| 4 | QA Tools | 15 |
| 4.1 | Debian Developer's Package Overview | 15 |
| 4.2 | Debian Maintainer Dashboard | 15 |
| 4.3 | Debian Package Tracker | 15 |
| 4.4 | Lintian | 15 |
| 5 | Changelog | 17 |
| 5.1 | Version 0.7 | 17 |
| 5.2 | Version 0.6 | 17 |
| 5.3 | Version 0.1 | 17 |
| 6 | Copyright | 19 |

Contents:

INTRODUCTION

The Debian Hamradio Maintainers team collaborates on maintenance of amateur-radio related packages for Debian. Team members may also maintain radio-related packages independently using the team's infrastructure; there is no obligation to share all radio-related package development.

Anyone is welcome to join; please contact the mailing list. New team members may help with the existing packages or create new packages as they wish. You do not need to be a Debian Maintainer or Debian Developer in order to contribute to packages. See below for more information on getting involved.

1.1 Joining the team as a packager

To join the team you should join our mailing list at a bare minimum. In order to use our Git repositories, you will need an Alioth account and this is very much encouraged. If possible, you are also encouraged to join our IRC channel.

To join the mailing list, visit [the subscription page](#) and enter your email address to subscribe. You will have to reply to an email in order to confirm your subscription.

To sign up for an Salsa account, visit [the Salsa registration page](#) and register. You can then visit the [project page for the Hamradio Maintainers team](#) and request to be added to the team.

Our IRC channel is [#debian-hams](#) on [irc.debian.org](#).

The following should be considered essential reading for anyone wishing to participate in packaging within the team:

- [Debian Policy](#)
- [Developers Reference](#)
- [New Maintainers Guide](#)
- [Debian Hamradio Maintainers Guide \(this document\)](#)

1.2 Assisting the team with translations

If you speak a language other than English, you can contribute right away with translations of package descriptions at the [Debian Description Translation Project](#).

PACKAGING

2.1 Newcomer guidelines

Newcomers may try to start off with a `debian/` directory and move files around until they have a working binary `.deb`. This sort of package will never enter the official Debian archives however. The only way to get your package into the Debian archives is through a proper source package that complies with [Debian Policy](#). See the [Debian New Maintainers Guide](#) for help on getting started with packaging. You may also find our [salsa script](#) useful for configuring new repositories if you have an account on salsa.debian.org.

2.2 Announcing intention to package

If the package you would like to work on is new to Debian, you should announce your intention to create the package. You do this by filing a WNPP bug report. Please add an X-Debbugs-CC header pointing to the Debian Hamradio Maintainers mailing list when you submit the bug.

2.3 Team Policy

It will be expected that you have read the [Debian Policy](#), [Developers Reference](#) and the [New Maintainers Guide](#).

In order to harmonise the packages within the Debian Hamradio Maintainers team, please adhere to the following additional team policy:

2.3.1 `debian/control`

The following considerations should be made when creating or modifying a package's control file.

| | |
|-------------------|---|
| Section | Should be <code>hamradio</code> for the source package in most cases. In some cases <code>science</code> will be more appropriate. |
| Priority | Should be <code>optional</code> unless forbidden by the Debian Policy (see §2.5). |
| Maintainer | Maintainer should be Debian Hamradio Maintainers <debian-hams@lists.debian.org>. Please subscribe to this list if you list yourself in the Uploaders: field of one of the team's packages. |
| Uploaders | Please add yourself as an uploader when you have a significant interest in a package. Being Uploader means that you are expected to be responsive to bug reports and act as a primary contact for the package. For more occasional works, you can do a team upload. There must be at least one named person in the Uploaders field for each team package. |
| Standards-Version | Please always use the latest unless there are concerns for backporting. If no changes are needed, please indicate this fact in the changelog, and increment the value of the field. |
| Homepage | Should be documented whenever possible. |
| Vcs-* | Please use the following template when using the team's Git repositories on Salsa: Vcs-Browser: <code>https://salsa.debian.org/debian-hamradio-team/<pkg></code> Vcs-Git: <code>https://salsa.debian.org/debian-hamradio-team/<pkg>.git</code> |

It is a good idea to check the syntax of the control file with `Config::Model`. This tool is also capable of reformatting the control file to a unified format (although the reformatting currently appears to only work if there are errors in the file to be fixed).

```
sudo apt-get install cme libconfig-model-dpkg-perl libconfig-model-itself-perl
cme check dpkg-control # checks the control file syntax
cme fix dpkg-control # fixes errors if possible
```

2.3.2 debian/copyright

We use the machine-readable format for the `debian/copyright` file. The `Source` field does not need to contain the full URL to the particular version that is being packaged, since this can be determined by the `uscan` program with the `debian/watch` file. Please list yourself in the `Files: debian/*` section if you think that your contributions are not trivial and therefore subjected to copyright. Please choose a license that is compatible with the program you package. You can also use “same as if it were in the public domain” or “same as the packaged program itself”.

To create some reasonable skeleton for a `debian/copyright` file you can try the following:

```
sudo apt-get install devscripts cdb
licensecheck --copyright -r `find -type f | /usr/lib/cdb/licensecheck2dep5 > debian/
↳ copyright
```

To verify the correct syntax of the `debian/copyright` file you can use `Config::Model` again:

```
cme check dpkg-copyright
```

2.3.3 debian/changelog

Packages hosted in our Git repositories on Alioth that have been modified but not uploaded must use UNRELEASED as a distribution name. QA tools will use this to identify packages that are being worked on but for which uploads are not yet ready.

2.3.4 debian/README.source

This file is recommended by the Policy (§4.14) from version 3.8.0 for documenting source package handling. Please follow the recommendation.

2.3.5 debian/README.test

This file was recommended by the Security team for describing to others than the regular maintainer how the package's functionality can properly be tested. If it is not obvious how to test the functionality of your package, please consider including this file.

2.3.6 debian/source/format

This file should contain “3.0 (quilt)” in order to use this source format. Other formats should be avoided unless they bring a specific advantage.

2.3.7 Debhelper

Debhelper uses compatibility levels to control the behaviour of its commands. We currently recommend to use the level 13 which is available in both stable (bullseye) and as a backport in oldstable (buster). However, there is no urgent need to touch packages only because it has an older Debhelper version.

It is strongly recommended to use the short dh notation in debian/rules files which makes code factorisation very simple and easy to understand the packaging for other members of the team. Even complex packaging becomes quite transparent this way.

2.3.8 Team Uploads

If you have contacted the uploaders listed for a package and they are unresponsive, or if you have been asked by the uploaders to perform a team upload, you may perform an upload without adding yourself to the uploaders field. Team uploads are distinguished from NMUs by adding “* Team upload” as the first changelog entry, and this is supported by debchange (-team).

For more information about team uploads, you can see the page about *team uploads* <<https://wiki.debian.org/TeamUpload>> on the Wiki.

2.3.9 /var/ax25

Many packet radio applications use `/var/ax25` as a location for storing operational state. This directory is not listed in the filesystem hierarchy standard but this has been the directory used in Debian for decades and so has become a de-facto standard. Unless the Debian technical committee decide otherwise (no one has asked at the time of writing) use `/var/ax25` in preference to `/var/lib/ax25`. The team will consider use of `/var/lib/ax25` as a bug in Debian packages.

VERSION CONTROL SYSTEM

The Debian Hamradio Maintainers team has a project on [Salsa](#). You are encouraged to use this for managing git repositories for the source of your team packages. If you do not currently have an account on Salsa or have not requested to be added to the project, you will need to do that before you can use this facility. To request to be added to the project, send an email to the [mailing list](#). In order to be able to push to the repositories on Salsa, you will need to add an SSH key to your Salsa account.

3.1 Anonymous access to git repositories

Anonymous access to git repositories is possible at the following URLs:

- <https://salsa.debian.org/debian-hamradio-team/<pkg-name>.git> (Preferred for the Vcs-Git field)

Anonymous access to a web VCS browser is also available at the following URL:

- <https://salsa.debian.org/debian-hamradio-team/<pkg-name>.git>

You can also view an [index of all team git repositories](#).

3.2 Repository layout

The layout of the Git repository is important as we have QA tools that scan our Git repositories for information. The recommended layout for repositories is described by [DEP-14](#), which is implemented by the `git-buildpackage` tool. Examples of this tool in use are given below.

3.3 Importing an existing source package

You will need `git-buildpackage` installed on your local machine before following these steps.

If your package is not currently tracked with git, you can import the sources from a source package.

Begin by logging into Salsa and creating the new repository. To configure the Salsa repository you've created you can use the script at <https://salsa.debian.org/debian-hamradio-team/hamradio-maintguide/tree/master/salsa>.

If you have the source package locally, you can skip this next step. If not, you will need to download the source package from the Debian archives:

```
cd /tmp
apt-get source <pkg-name>
```

Then you need to create a local git repository on your machine and import the source package:

```
mkdir <pkg-name>
cd <pkg-name>
git init
gbp import-dsc --pristine-tar /tmp/<pkg-name>*.dsc
```

Finally, in your local git repository, add the remote and push all branches and tags:

```
git remote add origin git+ssh://salsa.debian.org/debian-hamradio-team/<pkg-name>.git
git push origin --all
git push origin --tags
```

Please remember to update your `Vcs-Git` and `Vcs-Browser` fields in `debian/control` to match the URLs shown above for anonymous access.

3.4 Importing an existing git repository

You will need `git-buildpackage` installed on your local machine before following these steps.

If you've previously had a git repository elsewhere for your package, you can import it quite easily.

Begin by logging into Salsa and creating the new repository. To configure the Salsa repository you've created you can use the script at <https://salsa.debian.org/debian-hamradio-team/hamradio-maintguide/tree/master/salsa>.

Then in your local git repository, add the remote and push all branches and tags:

```
git remote remove origin
git remote add origin git+ssh://salsa.debian.org/debian-hamradio-team/<pkg-name>.git
git push origin --all
git push origin --tags
```

This repository can now be cloned using `git-buildpackage` or you can continue to work in your local repository pushing changes to the new origin.

Please remember to update your `Vcs-Git` and `Vcs-Browser` fields in `debian/control` to match the URLs shown above for anonymous access.

3.5 Creating a new git repository for a new package

You will need `git-buildpackage` installed on your local machine before following these steps.

Begin by logging into Salsa and creating the new repository.

On your machine, download the upstream source tarball and keep it somewhere safe. You will need a tarball to import, so if upstream distributes source as a zip or other type of archive, you will need to repack it as a tarball.

Create a local git repository on your machine and import the tarball like so:

```
mkdir <pkg-name>
cd <pkg-name>
git init
gbp import-orig --pristine-tar /path/to/upstream/source.tar.gz
```

You'll now find that the upstream source has been imported into your git repository.

To configure the Salsa repository you've created you can use the script at <https://salsa.debian.org/debian-hamradio-team/hamradio-maintguide/tree/master/salsa>.

When you are ready to push your changes to Salsa, add the remote and push all branches and tags:

```
git remote add origin git+ssh://salsa.debian.org/debian-hamradio-team/<pkg-name>.git
git push --all
git push --tags
```

3.6 Updating a package already in the team git

You can update an existing Debian package in git to the latest upstream with one of the following commands:

```
gbp import-orig --pristine-tar --uscan # will use the watch file to locate
                                        # the latest upstream
gbp import-orig --pristine-tar --download <URL> # will download a tarball
                                                # from the URL before
                                                # importing
gbp import-orig --pristine-tar /path/to/orig.tar.gz # import from local
                                                # tarball
```

3.7 Testing package builds with git-buildpackage

While building your package with git-pbuilder as described below will help you to test building your package on a clean system and check your build dependencies, it can take a long time to install the build dependencies every time. You can build your package in the directory on your development system while debugging build problems.

```
gbp buildpackage -us -uc
```

Note: The `-us -uc` options will be passed to `dpkg-buildpackage` and disable the automatic signing of the package. There is no need to sign the built packages when debugging.

The `.dsc`, `.changes`, `.orig.tar.gz`, `.debian.tar.xz` and `.deb` files will be placed in the parent directory to your git repository.

3.8 Using git-buildpackage and git-pbuilder for building packages

`git-pbuilder` builds your package in a chroot which ensures that your package has the correct package dependencies and does not depend on any files outside of the repository. It is included in the `git-buildpackage` package.

The first time you use `git-pbuilder`, you'll need to create a local unstable chroot for the builds to be performed in. You will also need to make sure `cowbuilder` is available on your system.

```
sudo apt-get install cowbuilder
sudo git-pbuilder create
```

In order to build your package from the git repository, run:

```
cd /path/to/repository
gbp buildpackage --git-pbuilder
```

3.9 Final build and upload

If your package is ready for upload, you can update the changelog, perform a final build with the updated changelog, sign your package using `debsign` and then upload:

```
dch -r # this will set the distribution to unstable
gbp buildpackage --git-pbuilder --git-tag
cd ../
debsign <pkg>*.changes
dput <target> <pkg>*.changes
```

Note: You may choose to do the final build without `git-pbuilder`, in which case you can omit the `--git-pbuilder` argument from the call to `gbp buildpackage`. The `.dsc` and `.changes` files will also be signed during the build so you do not have to use `debsign` if you are not using `git-pbuilder` for the final build.

If you're not a Debian Maintainer or Debian Developer, you will not be able to upload directly into the Debian archives. See the "Publishing your packages" section of the [mentors.debian.org introduction](https://mentors.debian.org/introduction) to learn how to upload your package there and then send a mail to the mailing list requesting sponsorship for your package.

3.10 Backports and updates to stable distributions

Backports and updates to stable distributions should be managed in a separate branch in the Git repositories. Backports should be managed in the branch with the name of the stable distribution, e.g. the branch containing backports for the current stable distribution should be named "bullseye". Proposed updates for the stable distributions should be managed in the branch with the name of the stable distribution with "-pu" appended, e.g. the branch containing proposed updates for the current stable distribution should be named "bullseye-pu".

In order to use `git-buildpackage` to build the alternative branch, use the `-git-debian-branch` argument:

```
gbp buildpackage --git-pbuilder --git-debian-branch=bullseye
```

3.11 Tracking commit notifications

Commit notifications are displayed in the `#debian-hams` IRC channel on `irc.debian.org` and can also be found on the `debian-hams@lists.debian.org` mailing list.

3.12 Further Reading

For more information on using Git for packaging, see the [Git packages](#) pages on the wiki.

QA TOOLS

A number of QA tools are available to the team to help track our packages. You can also use most of these tools to track the packages that you are an uploader on.

4.1 Debian Developer's Package Overview

Debian Developer's Package Overview (DDPO) is a web application offering a customizable overview of all packages of a given maintainer.

- [Link: DDPO for the Debian Hamradio Maintainers](#)

4.2 Debian Maintainer Dashboard

The maintainer dashboard exposes information about teams or maintainers' packages. It intends to help answering the question "I have a few hours for Debian, what should I do now?"

- [Link: DMD for the Debian Hamradio Maintainers](#)

4.3 Debian Package Tracker

The Debian Package Tracker service is a collection of scripts that gather information about the team's packages. It allows you to see a bird's eye view the health of many packages, instantly realizing where work is needed. It also allows you view the details of an individual package, including links to open bugs, the version in different Debian releases, news about recent uploads, links to build logs, the packaging repo, and more.

- [Link: DPT for the Debian Hamradio Maintainers](#)

4.4 Lintian

Lintian dissects Debian packages and tries to find bugs and policy violations. It contains automated checks for many aspects of Debian policy as well as some checks for common errors.

- [Link: Lintian reports for the Debian Hamradio Maintainers](#)

CHANGELOG

5.1 Version 0.7

- Refer to current stable as “bullseye”
- Replace QA reference to PET with Debian Package Tracker

5.2 Version 0.6

- Add subsection regarding `/var/ax25`

5.3 Version 0.1

- Initial version by Iain R. Learmonth <irl@debian.org>

COPYRIGHT

Copyright 2015 (C) Iain R. Learmonth <irl@debian.org>.

Redistribution **and** use **in** source **and** binary forms, **with or** without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this **list** of conditions **and** the following disclaimer.
2. Redistributions **in** binary form must reproduce the above copyright notice, this **list** of conditions **and** the following disclaimer **in** the documentation **and/or** other materials provided **with** the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.