Debian Hamradio Maintainers Guide

Release 0.6

Debian Hamradio Maintainers

Aug 05, 2019

CONTENTS

Contents:

CHAPTER

ONE

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 Alioth account, visit the Alioth homepage 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.

CHAPTER

TWO

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 package template useful.

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 `hamradio` for the source package in most cases. In some cases `science` will be more appropriate.
Priority	Should be `optional` 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 Alioth:
	`Vcs-Browser: https://salsa.debian.org/debian-hamradio-team/<pkg>`
	`Vcs-Git: https://salsa.debian.org/debian-hamradio-team/<pkg>.git`

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 chose 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 cdbs
licensecheck --copyright -r `find -type f` | /usr/lib/cdbs/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 sould 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 9 which is available in both stable (jessie) and oldstable (wheezy) and has been backported to old-old-stable (squeeze) 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 hierachy 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.

CHAPTER

THREE

VERSION CONTROL SYSTEM

The Debian Hamradio Maintainers team has a project on Alioth. 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 Alioth 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 Alioth, you will need to add an SSH key to your Alioth 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 Alioth and creating the new repository:

```
cd /git/pkg-hamradio/
./setup-repository <pkg-name> "Packaging for <pkg-name> in Debian"
```

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://alioth.debian.org/git/pkg-hamradio/<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 Alioth and creating the new repository:

```
cd /git/pkg-hamradio/
./setup-repository <pkg-name> "Packaging for <pkg-name> in Debian"
```

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://alioth.debian.org/git/pkg-hamradio/<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.

Even if your existing repository is on Alioth, do **NOT** just copy the repository into `/git/pkg-hamradio/`. The `setup-repository` script adds commit hooks to the repository to aid in tracking commits.

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 Alioth and creating the new repository:

```
cd /git/pkg-hamradio/
./setup-repository <pkg-name> "Packaging for <pkg-name> in Debian"
```

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.

You may optionally use the Debian Hamradio Maintainers template `debian/` directory:

```
svn export svn://anonscm.debian.org/svn/pkg-hamradio/trunk/package_template debian
```

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

```
git remote add origin git+ssh://alioth.debian.org/git/pkg-hamradio/<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 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 seperate 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 "jessie". 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 "jessie-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=jessie
```

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 pkg-hamradio-commits@lists.alioth.debian.org mailing list (archives).

3.12 Futher Reading

For more information on using Git for packaging, see the Git packages pages on the wiki.

CHAPTER

FOUR

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 Package Entropy Tracker

PET is a collection of scripts that gather information about the teams packages. It allows you to see in a bird's eye view the health of hundreds of packages, instantly realizing where work is needed. It will also show where work is in progress in the git repositories.

- Link: PET 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

CHAPTER

FIVE

CHANGELOG

5.1 Version 0.1

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

CHAPTER

SIX

COPYRIGHT