Debian Tcl/Tk Policy

Francesco Paolo Lovergine <frankie@debian.org>
Sergei Golovan <sgolovan@debian.org>

version 0.3.0

Abstract

This document describes the packaging of Tcl/Tk within the Debian distribution and the policy requirements for Tcl/Tk extensions and packages. This policy has been defined during Jessie release cycle, so pre-Jessie releases can violate this policy in one or more aspects. Backporters are warned.

Copyright Notice

Copyright © 2007-2013 Software in the Public Interest

This manual is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

A copy of the GNU General Public License is available as `/usr/share/common-licenses/GPL` in the Debian GNU/Linux distribution or on the World Wide Web at The GNU Public Licence (http://www.gnu.org/copyleft/gpl.html).

You can also obtain it by writing to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Contents

1 Tcl/Tk Packaging **1**

1.1 Versions . . . 1

1.2 Main packages . . . 1

1.3 Tcl and Tk Interpreters . . . 2

1.3.1 Interpreters Names . . . 2

1.3.2 Interpreters Locations . . . 2

1.4 Tcl/Tk libraries . . . 2

1.5 Tools/files for Development of Tcl/Tk modules and extensions . . . 2

1.6 Multiarch support . . . 2

1.7 Auto_load Path . . . 2

1.8 Documentation . . . 3

2 Packaged Modules **5**

2.1 Module Names . . . 5

2.2 Dependencies . . . 5

3 Tcl/Tk Programs **7**

3.1 Version Independent Programs . . . 7

3.2 Version Dependent Programs . . . 7

A Build Dependencies **9**

B Tcl/Tk modules loading **11**

C Possible issues building Tcl/Tk extensions **13**

D Maintainer's Checklist **15**

Chapter 1

Tcl/Tk Packaging

1.1 Versions

At any given time, the binary packages `tcl` and `tk` represent the current default Debian Tcl/Tk version. They contain Tcl and Tk shells `/usr/bin/tclsh` and `/usr/bin/wish` which are symlinks to the current default version binaries. For backward compatibility they provide the virtual packages `tclsh` and `wish` but it isn't recommended to reference them in packages dependencies anymore. The packages `tcl` and `tk` are provided by the Debian `tcltk-defaults` source package, in order to manage modules and extensions packaging and upgrading better. Modules should preferably use those packages when appropriate (i.e. they are either version independent or properly versioned to inhibit the use of a non compatible versions, see 'Dependencies' on page 5), but it is not mandatory. The default packages are

```
tcl
tk
tcl-dev
tk-dev
tcl-doc
tk-doc
```

The default Debian Tcl/Tk version should always be the latest stable upstream release that can be integrated in the distribution. Starting from 8.0, Tcl and Tk share the same version numbering. The default packages depend on the appropriate versioned packages and provide useful additional symlinks. Default packages versions follow upstream versions, so that packages can use appropriate versioning constraints on them when it is needed.

Apart from the default version, legacy versions of Tcl/Tk may be included as well in the distribution, as long as they are needed by other packages, or as long as it seems reasonable to provide them. (Note: For the scope of this document, Tcl/Tk versions mean the result of 'info tclversion' command, i.e. Tcl/Tk 8.5 and 8.5.14 are subminor versions of the same Tcl/Tk version 8.5, but Tcl 8.6 and 8.5 are indeed different versions. The patchlevel intends the result of the 'info patchlevel' command, i.e. Tcl/Tk 8.5.14 and 8.5.13 have the same version but different patchlevels).

In addition, unstable/development version of Tcl/Tk may be included in the unstable/experimental distribution.

For any version, the main Tcl and Tk packages are called `tcl`*X*`.`*Y* and `tk`*X*`.`*Y* respectively. They are always packaged as separate sources, as for upstream. Names of related packages or extensions must follow the same convention if the inclusion of multiple versions make sense or if they work only with specific versions of Tcl or Tk.

To avoid definition clashes with Debian terminology, we will call *modules* any Tcl/Tk packages which consist uniquely of Tcl/Tk sources, and *extension* any program which extends consistently Tcl/Tk using TEA and shared libraries. Note that this is not completely consistent with Tcl terminology, which started from version 8.5 also introduces *.tm modules* and traditionally deals with *packages* and *script libraries*.

1.2 Main packages

For every Tcl/Tk versions provided in the distribution, the packages `libtcl`*X*`.`*Y* and `libtk`*X*`.`*Y* comprise a corresponding Tcl/Tk libraries, core modules and extensions of the upstream Tcl/Tk distribution. They provide infrastructure for embedding Tcl into external programs. Any such packages includes a *Provides:* item of the virtual package `libtcl` and a *Provides:* item for the `libtk` virtual package.

Also, the packages `tcl`*X*`.`*Y* and `tk`*X*`.`*Y* ship the binaries `/usr/bin/tclsh`*X*`.`*Y*, `/usr/bin/wish`*X*`.`*Y*. Starting from Jessie they don't provide alternatives for files `/usr/bin/tclsh` and `/usr/bin/wish`. If an application uses one of those, it has to depend on `tcl` or `tk` package.

Tools and files for the *development* of Tcl/Tk extensions are split off in two separate packages `tclX.Y-dev` and `tkX.Y-dev`. Documentation is provided separately in packages `tclX.Y-doc` and `tkX.Y-doc`.

1.3 Tcl and Tk Interpreters

1.3.1 Interpreters Names

Tcl/Tk scripts depending on the default Tcl/Tk version (see 'Main packages' on the previous page) or not depending on a specific Tcl/Tk version must use `tclsh` and/or `wish` (unversioned) as the interpreter name and must depend on `tcl` and/or `tk` package.

Tcl/Tk scripts that only work with a specific Tcl/Tk version must explicitly use the versioned interpreter name (`tclshX.Y` and/or `wishX.Y`) and must depend on the specific Tcl/Tk versioned package (`tclX.Y` and/or `tkX.Y` respectively).

1.3.2 Interpreters Locations

The path name for the Tcl interpreter is `/usr/bin/tclsh` or `/usr/bin/tclshX.Y`.

The path name for the Tk interpreter is `/usr/bin/wish` or `/usr/bin/wishX.Y`.

If a maintainer would like to provide the user a possibility to override the Debian Tcl interpreter, he may want to use `/usr/bin/envtclsh` or `/usr/bin/envtclshX.Y`. The same consideration applies for Tk and the `wish` interpreter. Starting from Jessie administrators no longer can override default versions of the interpreters using `update-alternatives`, which provided some flexibility but caused confusion for program maintainers before.

1.4 Tcl/Tk libraries

The Tcl and Tk libraries are provided by `libtclX.Y` and `libtkX.Y` respectively. These packages install `/usr/lib/$(DEB_HOST_MULTIARCH)/libtclX.Y.so` (soname is `libtclX.Y.so`) and `/usr/lib/$(DEB_HOST_MULTIARCH)/libtkX.Y.so` (soname is `libtkX.Y.so`).

1.5 Tools/files for Development of Tcl/Tk modules and extensions

Some tools and files for development of Tcl/Tk modules and extensions are packaged as `tclX.Y-dev` and `tkX.Y-dev`. These packages provide header files as well as static and stub libraries. Header files are installed in `/usr/include/tclX.Y` directory (for both Tcl and Tk). Default packages `tcl-dev` and `tk-dev` provide symlinks to the right versioned header files directory

```
/usr/include/tcl -> /usr/include/tclX.Y
/usr/include/tk -> /usr/include/tclX.Y
```

See net section and 'Possible issues building Tcl/Tk extensions' on page 13 for more information about possible issues with extension building due to Debian customizations.

1.6 Multiarch support

The Tcl and Tk libraries and development packages can be installed for several architectures simultaneously. They ship architecture dependent files in `/usr/lib/$(DEB_HOST_MULTIARCH)` directory. See also section 'Possible issues building Tcl/Tk extensions' on page 13 for information on how to package multiarchified Tcl/Tk extension.

1.7 Auto_load Path

The package search path (*auto_path*) for both Tcl and Tk is a list searched in the following order:

Site modules and extensions: `/usr/local/lib/tcltk (architecture dependent files)`
`/usr/local/share/tcltk (architecture independent files)`

Packaged modules and extensions: `/usr/lib/tcltk/$(DEB_HOST_MULTIARCH) (architecture dependent files)`
`/usr/lib/tcltk (architecture dependent files)`
`/usr/share/tcltk (architecture independent files)`

Version specific core modules and extensions: `/usr/share/tcltk/tclX.Y`
`/usr/share/tcltk/tkX.Y (for Tk only)`

Maintainers must ensure that modules and extensions are correctly installed in subdirs of the paths above consistently. See 'Tcl/Tk modules loading' on page 11 for more information about Tcl/Tk specific ways of dealing with modules and extensions loading. Developers must consider that these defaults impact TEA-based modules and use preferably system-wide `tcl.m4` (it isn't TEA-compatible though) instead of private one (see 'Possible issues building Tcl/Tk extensions' on page 13).

1.8 Documentation

Default packages `tcl-doc` and `tk-doc` which depend on default versioned `tclX.Y-doc` and `tkX.Y-doc` are provided. Since different `tclX.Y-doc` and `tkX.Y-doc` conflict in files and cannot be installed simultaneously, `tcl-doc` and `tk-doc` only recommend `tclX.Y-doc` and `tkX.Y-doc` to allow administrators to install any desirable package with Tcl/Tk manual pages. The package `tcl-doc` also includes a copy of the up-to-date version of this policy.

Chapter 2

Packaged Modules

2.1 Module Names

Packages should be named by the primary module provided. The naming convention for a module `foo` is `tcl-`*`foo`* or `tk-`*`foo`* when the module is version independent.

The naming convention for a module `foo` is `tcl`*`X`*`.`*`Y`*`-`*`foo`* or `tkX.Y-`*`foo`* when the module is version dependent.

If the package already uses the name `tcl`*`foo`* or `tk`*`foo`*, that is, naming without Tcl/Tk versioning even if version-dependent and/or without hyphen, may use the name for now. However, the package naming proposed above is recommended for consistency.

2.2 Dependencies

Packaged modules available for one particular version of Tcl/Tk must depend on the corresponding `libtcl`*`X`*`.`*`Y`* and/or `libtk`*`X`*`.`*`Y`* package if they don't contain Tcl/Tk scripts, and on `tcl`*`X`*`.`*`Y`* and/or `tk`*`X`*`.`*`Y`* package if they do.

The recommended dependencies of version-independent packages are the following: If the package works in all available Tcl/Tk versions it should depend on `libtcl` or `libtk` (`tcl` or `tk` if it includes Tcl/Tk scripts). If the package works for several Tcl/Tk versions (but not for all) it should depend on all of them alternatively (e.g. `libtcl8.5 | libtcl8.4`).

Chapter 3

Tcl/Tk Programs

3.1 Version Independent Programs

Programs that can run with any version of Tcl/Tk should be started with `#!/usr/bin/tclsh` or `#!/usr/bin/wish`. They must also specify a dependency on default packages `tcl` and/or `tk`. You're free to use `#!/usr/bin/env tclsh` and `#!/usr/bin/env wish`, if you'd like to give the user a chance to override the Debian Tcl/Tk package with a local version.

3.2 Version Dependent Programs

Programs which require a specific version of Tcl must start with `#!/usr/bin/tclsh`*X*`.`*Y*. They must also specify a strict dependency on `tcl`*X*`.`*Y*. Programs which require a specific version of Tk must start with `#!/usr/bin/wish`*X*`.`*Y*. They must also specify a strict dependency on `tk`*X*`.`*Y*. Again, if you're using `#!/usr/bin/env tclsh`*X*`.`*Y*, or `#!/usr/bin/env wish`*X*`.`*Y* please be aware that a user might override the Debian Tcl/Tk packages with a local version and that release default packages can change also.

Appendix A

Build Dependencies

Build dependencies for Tcl/Tk dependent packages must be declared for every Tcl/Tk version, that the package is built for. In order to build for a specific version, add the versioned Tcl/Tk packages dependencies; it is generally better and recommended depending on the appropriate default packages with an eventual strict or relaxed versioning.

Extension packages and applications which link to Tcl/Tk libraries should depend appropriately on one or more of the following packages (with or without additional package version relationships):

```
tcl-dev
tk-dev
tclX.Y-dev
tkX.Y-dev
```

For example, `tclreadline` build dependencies are the following:

```
Build-Depends: debhelper (>= 8.0.0), dpkg-dev (>= 1.16.1~), tcl-dev,
 libreadline5-dev, autotools-dev
```

Module packages, script libraries and Tcl-only applications should depend on the

```
tcl
tk
tclX.Y
tkX.Y
```

For example, `tcllib` build dependencies are the following:

```
Build-Depends: debhelper (>= 8.9.7)
Build-Depends-Indep: tcl
```

Due to limitations of current autobuild daemon it is forbidden to use build dependencies on the virtual packages `libtcl` or `libtk` only. These dependencies cannot guarantee consistent build environment, so it is mandatory to prepend a preferred Tcl/Tk version before each of the virtual packages.

Appendix B

Tcl/Tk modules loading

Tcl/Tk supports a few alternative ways for modules loading. Modules can be implemented as shared libraries, Tcl/Tk scripts or a combination of them. Generally, specific index scripts are used for that and they need to be placed in a directory included in the `auto_path` list along with scripts and libraries. Old non-package script libraries require a `tclIndex` generated with the `auto_mkindex` Tcl instruction. Packages require a `pkgIndex.tcl` file generated with the `pkg_mkIndex` Tcl instruction. Tcl 8.5 introduced Tcl Modules scripts (.tm) which do not require an index script for single file implementations. Another peculiar way of modules providing is based on *Starkit* and the use of the Tcl Virtual File System (TclVFS). In each of those cases, the maintainer must ensure that the all required files are included and installed in the right place to allow using of the module.

Appendix C

Possible issues building Tcl/Tk extensions

In order to support this policy, a number of changes have been applied in upstream `init.tcl`, `tclConfig.sh`, `tkConfig.sh` and `tcl.m4` files. Extensions using TEA with local copies of `tcl.m4`, or which use custom guesses about Tcl/Tk configuration could require explicit use of

```
--with-tcl=/usr/lib/tclX.Y
--with-tk=/usr/lib/tkX.Y
--with-tclincludes=/usr/include/tclX.Y
```

or other custom hacks.

Appendix D

Maintainer's Checklist

1 Tcl/Tk has in general a very backward compatible API both at C and commands set levels. When packaging a module or extension, verify if it requires a specific minimal version and if it includes run-time checks about that, possibly (which is always convenient). Scripts can include a `packagerequire?-exact? TclX.Y` or `packagerequire ?-exact? TkX.Y`, for instance. In those cases, use a versioned dependency on one of the default packages (`tcl`, `tk`, `tcl-dev` or `tk-dev`), e.g. `tcl` (>= *X.Y*). That is recommend instead of versioned packages dependencies, which are anyway supported for compatibility with past conventions. Note also that `tk` depends on `tcl` and `tk-dev` depends on `tcl-dev`.

2 Always install your package stuff in a per-package sub-directory of `/usr/share/tcltk` (for scripted modules) and/or `/usr/lib/tcltk` (for shared library extensions) along with the needed index file (see 'Tcl/Tk modules loading' on page 11).

3 If your package supports multiarch install its index file into a subdirectory of `/usr/lib/tcltk /$(DEB_HOST_MULTIARCH)` directory. Tcl/Tk cannot determine multiarch triplet in runtime, so package indices for different architectures must differ.

4 This policy customizes *auto_path* differently with respect to generic upstream UNIX platforms, so that you should use preferably system provided `tcl.m4`. Occasionally that could either require custom hacks for non TEA-based building systems, or using `-with-tcl` or `-with-tk` argument for TEA scripts.