

Common Lisp in Debian Manual

René van Bevern <rvb@debian.org>

Abstract

The Common Lisp in Debian Manual describes packaging practices for Common Lisp libraries and implementations for the Debian operating system and the standard way of using Debian-installed libraries. The packaging practices are based on the Common Lisp Controller design documentation by Kevin M. Rosenberg and Peter Van Eynde.

Copyright Notice

Copyright © 2006 René van Bevern.

This document 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 package 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 General Public License (<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	Common Lisp Controller	1
1.1	Customizing default images	1
1.2	Common Lisp Controller Setup	1
2	Implementations	3
2.1	Startup	3
2.2	Implementation Control Script	3
2.3	Maintainer Scripts	4
3	Libraries	5
3.1	Usage of libraries	5
3.1.1	User-specific Installation	5
3.2	Package Name	6
3.3	Installation Paths	6
3.3.1	Lisp Data	6
3.3.2	Foreign Data	6
3.4	Dependencies	6
3.5	Maintainer Scripts	6
4	Common Lisp Debhelper	9

Chapter 1

Common Lisp Controller

As Common Lisp is an ANSI Standard (<http://www.lispworks.com/documentation/HyperSpec/>), multiple implementations of the language exist in Debian. Because these have different objectives and distinct features beyond ANSI Common Lisp, this document cannot declare a default implementation.

Instead, Debian provides the Common Lisp Controller (CLC) from the `common-lisp-controller` package. The CLC manages Common Lisp implementations and libraries in a Debian system by compiling sources on demand for desired implementations and handling implementation updates.

E.g., it allows for loading libraries using `(clc:clc-require :asdf-system-name)` or rebuilding all installed sources using `(clc:clc-build-all-packages)` on a number of different implementations.

1.1 Customizing default images

Common Lisp Controller can pre-load the images of various implementations with frequently-needed libraries. It is working for all implementations that support saving the current memory image at run time (e.g. SBCL, Clisp, and CMUCL) and for some that need to link an image from pre-compiled objects (i.e. ECL at the moment).

To make use of this feature, write the names of the ASDF systems (one per line) you wish to have pre-loaded in files of the form `/etc/common-lisp/images/implementation`.

1.2 Common Lisp Controller Setup

Compiling the Common Lisp Controller for an implementation is done by loading `/usr/share/common-lisp/source/common-lisp-controller/common-lisp-controller.lisp`, and running `(common-lisp-controller:compile-common-lisp-controller-v5`

"implementation") afterwards. This function returns a list of object files that have resulted from the compilation of CLC. They are supposed to make up the new image and can be used by implementations like ECL to link one.

Setting up the Common Lisp Controller is done by invoking `(common-lisp-controller:init-common-lisp-controller-v5 "implementation")` afterwards. This can be done in the implementation's startup stage after an image with the above objects has been loaded, or you can set it up prior to saving a new default lisp image at runtime (this is the more usual case).

It *must* be possible to restore the original CLC-less state of the implementation.

Chapter 2

Implementations

Note that in the following text, each occurrence of *implementation* is to be replaced by the *same* name of the implementation in question.

2.1 Startup

Each Common Lisp implementation in Debian *must* provide the definitions of the Common Lisp Controller after startup. Their presence is *required* in non-interactive as well as interactive execution of the implementation. See ‘Common Lisp Controller Setup’ on page 1.

The implementation *must* load `/etc/lisp-config.lisp`.

2.2 Implementation Control Script

Implementations *must* provide a script located at `/usr/lib/common-lisp/bin/implementation.sh` that takes at least the following arguments:

install-clc Installs the definitions of the Common Lisp Controller into the implementation. See ‘Common Lisp Controller Setup’ on page 1.

remove-clc Restores the original state of the implementation without CLC definitions

The following actions are *optional*:

rebuild *asdf-system-name* recompiles the library described by *asdf-system-name*.

remove *destination-dir* deletes implementation-specific compiled object files of the library that is installed in *destination-dir*.

See ‘Libraries’ on page 5 for meanings of *destination-dir* and *asdf-system-name*. Installation of this script can be handled by the Debhelper for Common Lisp, see ‘Common Lisp Debhelper’ on page 9.

2.3 Maintainer Scripts

The following actions are *required*:

postinst invokes `/usr/sbin/register-common-lisp-implementation` *implementation* when called with the `configure` argument and invokes `/usr/sbin/unregister-common-lisp-implementation` *implementation* when called with the `abort-upgrade`, `abort-remove` or `abort-deconfigure` argument.

prerm invokes `/usr/sbin/unregister-common-lisp-implementation` *implementation* when called with the `remove`, `upgrade` or `deconfigure` argument.

The Common Lisp Debhelper (see ‘Common Lisp Debhelper’ on page 9) generates these scripts automatically.

3.2 Package Name

Common Lisp library packages *should* carry the prefix `cl-`. If the package is highly implementation specific, it *may* carry a corresponding suffix, e.g. `cl-clx-sbcl`.

3.3 Installation Paths

3.3.1 Lisp Data

Packages *must* install the Lisp *source code* to `/usr/share/common-lisp/source/destination-dir` and establish a symbolic link from `/usr/share/common-lisp/source/destination-dir/asdf-system-name.asd` to `/usr/share/common-lisp/systems/asdf-system-name.asd`

3.3.2 Foreign Data

Foreign data *must* be built at package build time and follow the rules below, unless a policy specific to that kind of data states otherwise.

Platform dependent data that is solely required by the provided library *should* reside in `/usr/lib/package-name`. This for example affects glue code in form of shared libraries for foreign function interfaces.

Platform independent data that is solely required by the provided library *should* reside in `/usr/share/package-name`.

These locations are also recommended by the Debian Policy Manual.

3.4 Dependencies

The dependencies of the Debian package *must* reflect the dependencies of the ASDF system definition, i.e. all ASDF systems that are depended on *must* also be available as Debian packages and be listed in `debian/control`. Libraries *must* also depend on the `common-lisp-controller` package.

3.5 Maintainer Scripts

The following actions are *required*:

`postinst` invokes `/usr/sbin/register-common-lisp-source destination-dir` when called with the `configure` argument.

preem invokes `/usr/sbin/unregister-common-lisp-source` *destination-dir* when called with the `remove`, `upgrade` or `deconfigure` argument.

See 'Libraries' on page 5 for the meaning of *destination-dir*.

Chapter 4

Common Lisp Debhelper

The Common Lisp Debhelper is provided by the `dh-lisp` package. To use it, you have to set a `Build-Depends`: on it.

The basic operations are taken by just invoking `dh-lisp -ppackage` in your `debian/rules` after all files have been copied to their destination, i.e. after `dh_install` or similar things are run. `dh_lisp` searches the package tree for ASDF system definitions, establishes symbolic links and generates `postinst` and `prerm` scripts automatically. This means that the above steps mentioned for library packaging are done automatically.

`dh_lisp` also scans the directory for pre-compiled Lisp code (i.e. CMUCL's `x86f`, SBCL's `fasl` and CLISP's `fas`) and adds dependencies for the correct implementation and version to the substitution variable `${misc:Depends}`. A dependency for the `common-lisp-controller` is also added, so your `debian/control` should contain a line like this for binary packages:
`Depends: ${misc:Depends}`.

An additional argument to `dh_lisp` is taken for an implementation name. (e.g. `dh_lisp -ppackage implementation`). It will then — additionally to above actions — look for your implementation script in `debian/implementation.sh`. `dh-lisp` installs the implementation control script and generates the required maintainer scripts.