Debreate Usage

Version 0.3

Contents

Section 1 – Introduction

Disclaimer

This usage documentation is comprised of what I know about the Debian packaging system and what I have been able to decipher from the Debian Policy Manual. If any mistakes are found herein, please notify the development team (a.k.a. me) and we will investigate whether the issue needs to be addressed or not. I am not accountable for any mistakes or misinformation within this usage document, but I am willing to make corrections.

Some information has been gathered from the Debian and Ubuntu policy manuals. Any text appearing to be taken directly from those sites falls under the pertaining licenses.

Packages and Package Managers

Most Linux and BSD based operating systems use package managers to install and organize software. This software is bundled in packages which are compressed archives. Debian based systems generally use the Debian Package Management System (dpkg) and the Advanced Packaging Tool (apt) or Aptitude for package management. There are also GUI frontends like Synaptic that can be used as well.

Binary Packages

Debian (deb) packages are Unix ar (a.k.a The Archiver) archive files. A package contains three items: A text file named “debian-binary” and two gzipped, bzipped or lzmaed tarball archives named “control” and “data”.

The debian-binary file simply contains the deb format version number.

The control tarball contains a file named “control” which contains all the package's meta data. It can also contain any of four executables to be executed when the package is installed or removed. These executables are named “preinst”, “postinst”, “prerm” and “postrm”. It also may contain a text file called “md5sums”. This is a checksum of the files contained in the data tarball. The debsums program can use this file the check the integrity of the installed files of the package.

The data tarball contains the actual files that will be installed to the system. These files are kept in a Unix style directory structure.

Debian packages usually follow a specific naming convention made up of the name of the package, its version and the intended architecture separated by underscores and cannot contain any spaces. An example would be **`hello_0.1_i386.deb`**.

Debreate

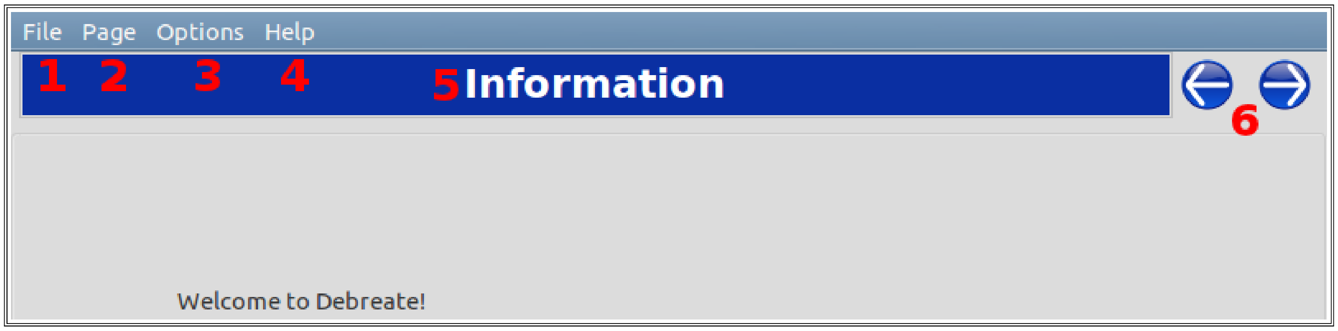
Debreate is a utility for creating binary deb packages. It walks the user through the process of making the control file and offers other options such as creating maintainer scripts (executables), a changelog, a copyright notice, and a system menu launcher.

In the build process, files specified by the user are copied into a directory structure (referred to as “build tree” from within the program). The control file is created using the information from the “Control” and “Dependencies” sections. This file is placed in a folder named “DEBIAN” at the root of the directory structure. Optionally, maintainer executables (scripts) and an md5checksum are created in this folder. The program then uses the command **`dpkg-deb -b`** for the actual building of the package.

Section 2 – Using Debreate

Menu Bar and Basic Usage

Figure 2-1



1: File Menu

Save or open projects.

2: Page Menu

Navigate to a different page.

3: Options Menu

Use custom open/save dialogs in place of the standard Gtk+ dialogs.

4: Help Menu

Information about Debreate and Debian packages and Debreate usage.

5: Title

Displays the name of the current page.

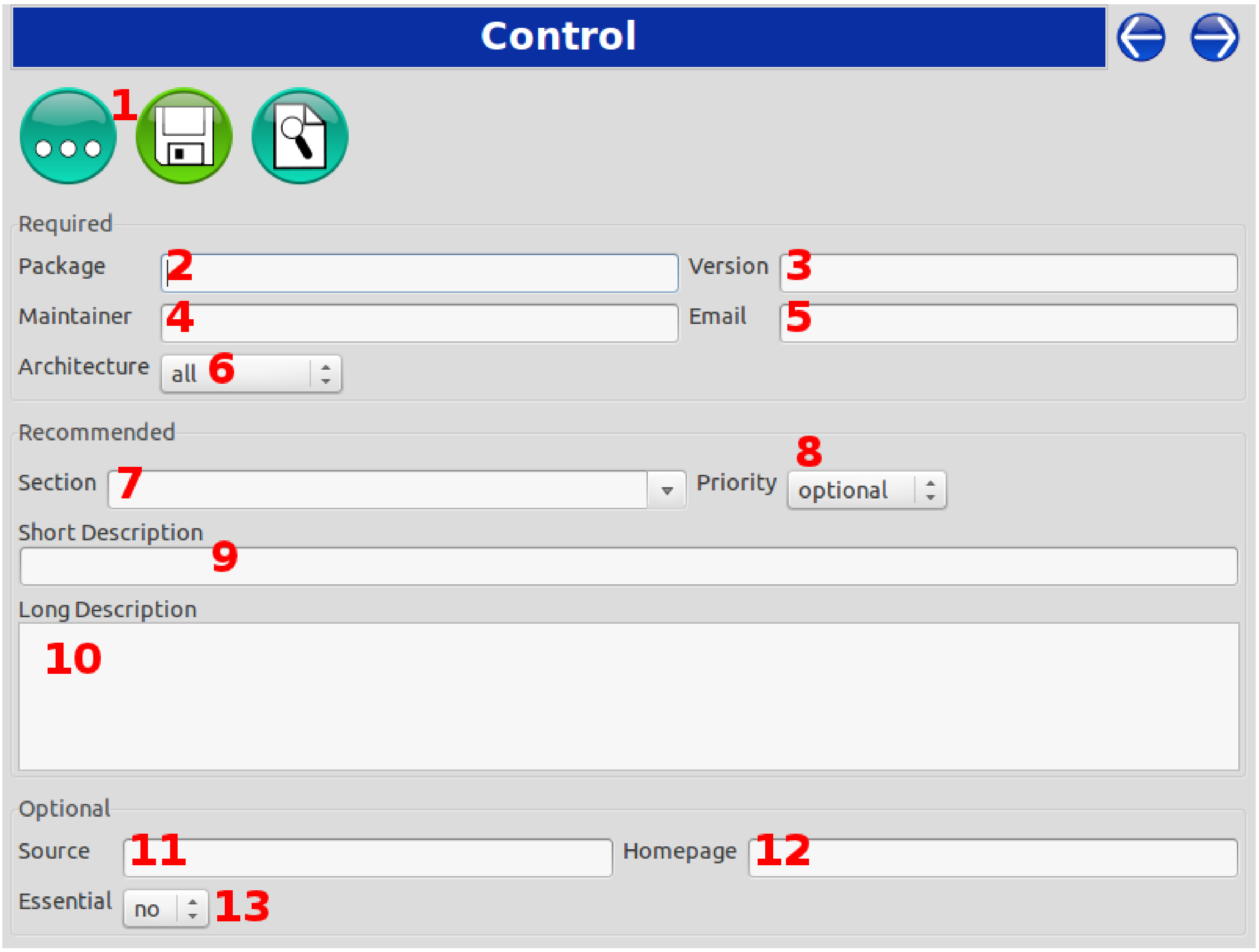
6: Navigation Buttons

Increment or decrement page.

Control

The control file is a text file that is the core of the deb package. This file contains all the information about the package needed by the package manager. The "Control" section of Debreate has a field for the necessary entries of the control file.

Figure 2-2



1: Buttons

Browse: This button can be used to open an existing control file.

Save: Save all the information from the Control and Dependencies and Conflicts sections to a control formatted text file.

Preview: Display a preview of the control file.

Required Fields

2: Package

This is the name of the package. It can only consist of alphabetic characters (a-z), digits (0-9), plus (+) and minus (-) signs, and periods.

3: Version

The version number of the package which should follow the format: [epoch:]upstream[-revision]. Some examples are: 0.2.11, 0.2.11-3, 1:0.2.11, 1:0.2.11-3.

4: Maintainer

The name(s) of the package maintainer(s), usually first and last.

5: Email

The package maintainer's email address.

6: Architecture

The computer architecture for which the package is intended. A list of usable architectures can be found with the command `dpkg-architecure -L`. If the package is intended for any architecture "all" should be specified.

Recommended Fields

7: Section

The classification of the package. Package managers use this to organize packages into sections. Debreate lists the recommended sections but it is not required to use one from this list. A custom section can be specified.

8: Priority

This represents how important it is that a user have the package installed. To view the priority options and their definitions check the Policy Manual's priority section.

9: Short Description (Synopsis)

This is a brief description of the package and should not include the package name.

10: Long Description (Extended Description)

A more detailed description should be placed here. It should describe what the package does and how it is related to the system but should not include instructions on how to use or configure the package. Significant dependencies and conflicts with other packages should be described here as well.

Optional Fields

11: Source

This is the name of the original source package. It can consist of the same characters as the package name and can optionally be followed by a space and the version number in parenthesis: **`hello (0.1)`**.

12: Homepage

Usually the URL of the website of the package.

13: Essential

Indicates whether the package is essential to the system. If set to "yes" the package manager will refuse to remove the package.

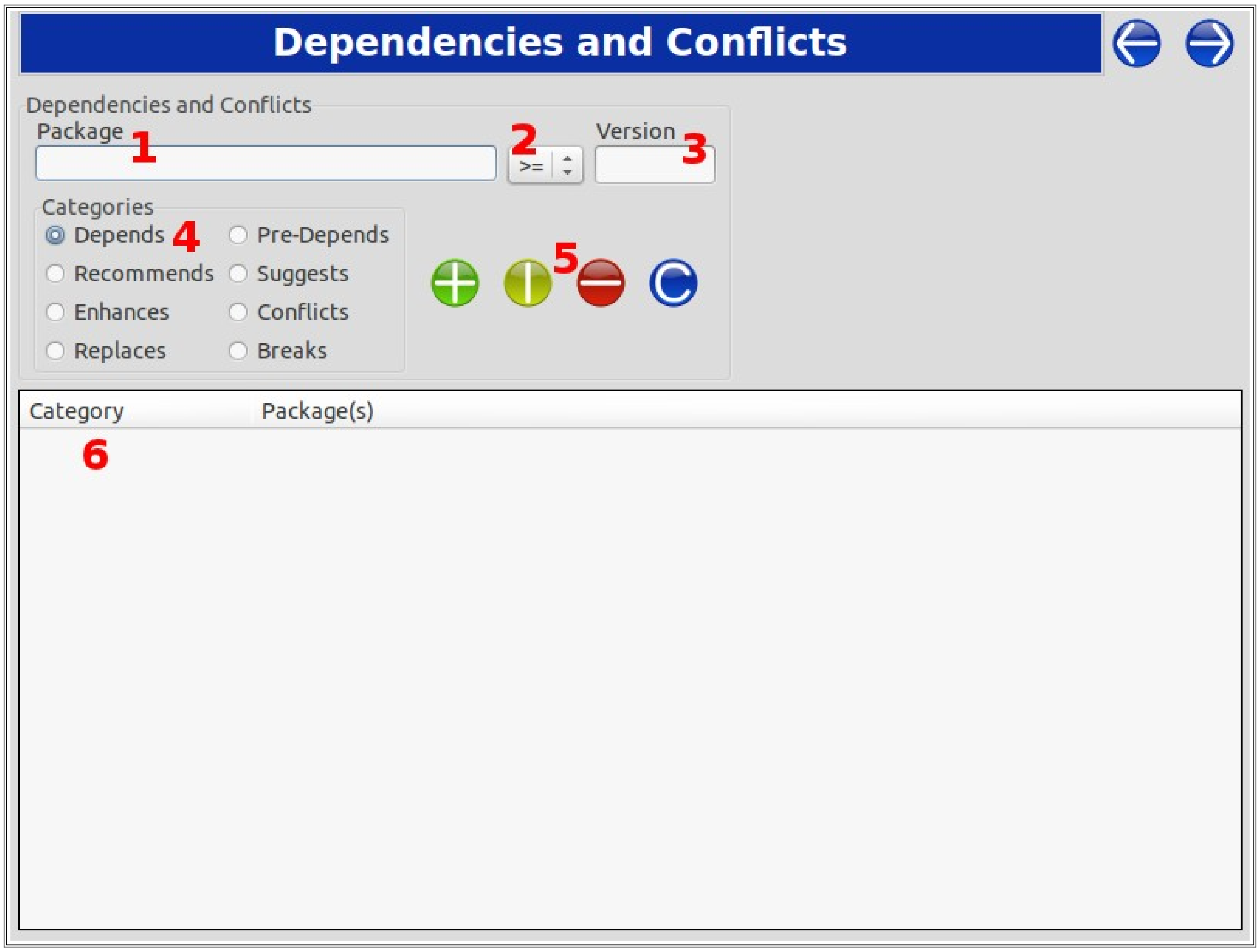
Note: There is another field in the control file that is useful but not listed. It is called *Installed-Size*. This is the combined size (in bytes) of the files to be installed to the system. Debreate automatically calculates this number and adds it to the control file.

Dependencies and Conflicts

Many packages require that other packages be installed in order to work properly. The Dependencies and Conflicts section provides an interface to specify which packages are required for correct installation.

Dependencies can either be declared by themselves or as alternatives. If a dependency is not available for the system but an alternative has been specified the package manager will look for the alternative to satisfy it. An unlimited number of alternatives may be specified.

Figure 2-3



1: Package Name

The name of the depended-on package.

2: Relations

This is the relation of the depended-on package to its version number.

Later or equal (>=): The version number must be equal to or greater than the specified.

Earlier or equal (<=): The version number must be equal to or less than the specified

equal (=): The version number must be equal to the specified.

Later (>>): The version number must be greater than the specified.

Earlier (<<): The version number must be less than the specified.

3: Version

The specified version of the depended-on package.

4: Categories

These categories control what is to be done with the specified package.

Depends: The package will not be installed unless the specified package is available for installation.

Pre-Depends: Specified package must be available and will be installed anterior.

Recommends: These packages are strongly recommended and by default will be installed alongside the package unless otherwise specified.

Suggests: These packages are not necessary for the package to work correctly but may be useful. They will not be installed by default.

Enhances: Specifies that the package may be useful for the specified package.

Conflicts: The specified package is in conflict and will be removed from the system if it is installed.

Replaces: The package may overwrite the specified package or its files.

Breaks: The specified package is in conflict and installation will not occur until the specified package has been de-configured.

Note: Another option not available yet in Debreate is “Provides”. Provides allows a package to provide dependencies for other packages. Say the package is called “foo” but it provides “bar”, the package “foo” will satisfy any other packages dependency on “bar”.

5: Buttons

Add: Adds the specified package to the category list.

Append (or | pipe): Adds an alternative to the selected dependency.

Remove: Removes the selected item from the category list.

Clear: Clears the category list.

6: Category List

Displays all the specified dependencies and conflicts.

For more information on control files:

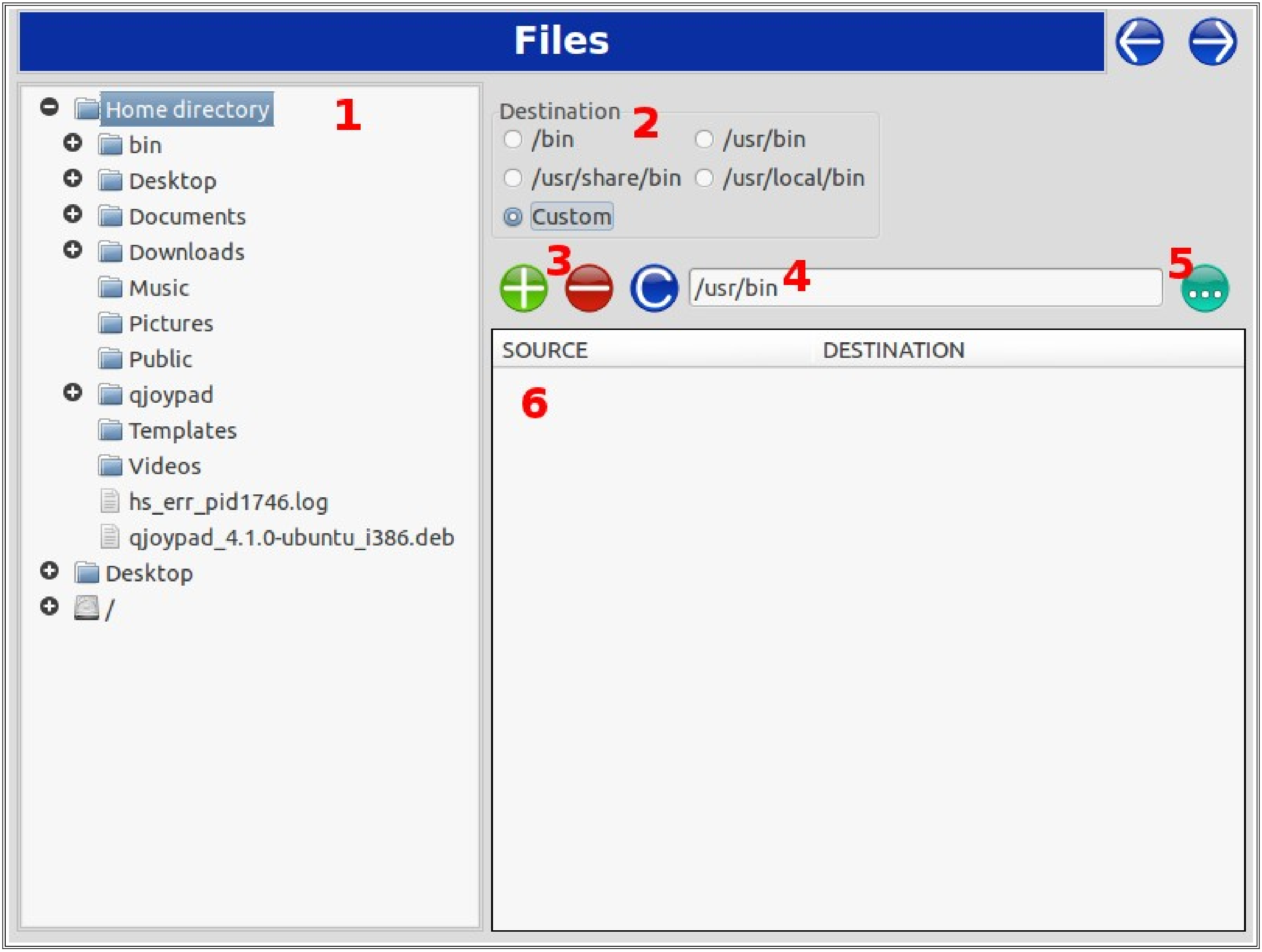
Debian Policy Manual – Control files and their fields

Ubuntu Policy Manual – Control files and their fields

Files

This section is for specifying which files will make up the contents of the package.

Figure 2-4



1: File System Hierarchy

Shows a tree view of the file system. Files and folders can be selected here to add to the package.

2: Destination

Defines where the selected file/folder will be installed on the system.

3: Buttons

Add: Adds the selected file/folder to the file list with the selected destination.

Remove: Removes the selected item from the file list.

Clear: Clears the file list.

4: Custom Destination

If “Custom” is selected from the destinations, this string will be used as the destination.

5: Browse Button

Can be used to browse for a custom destination.

6: File List

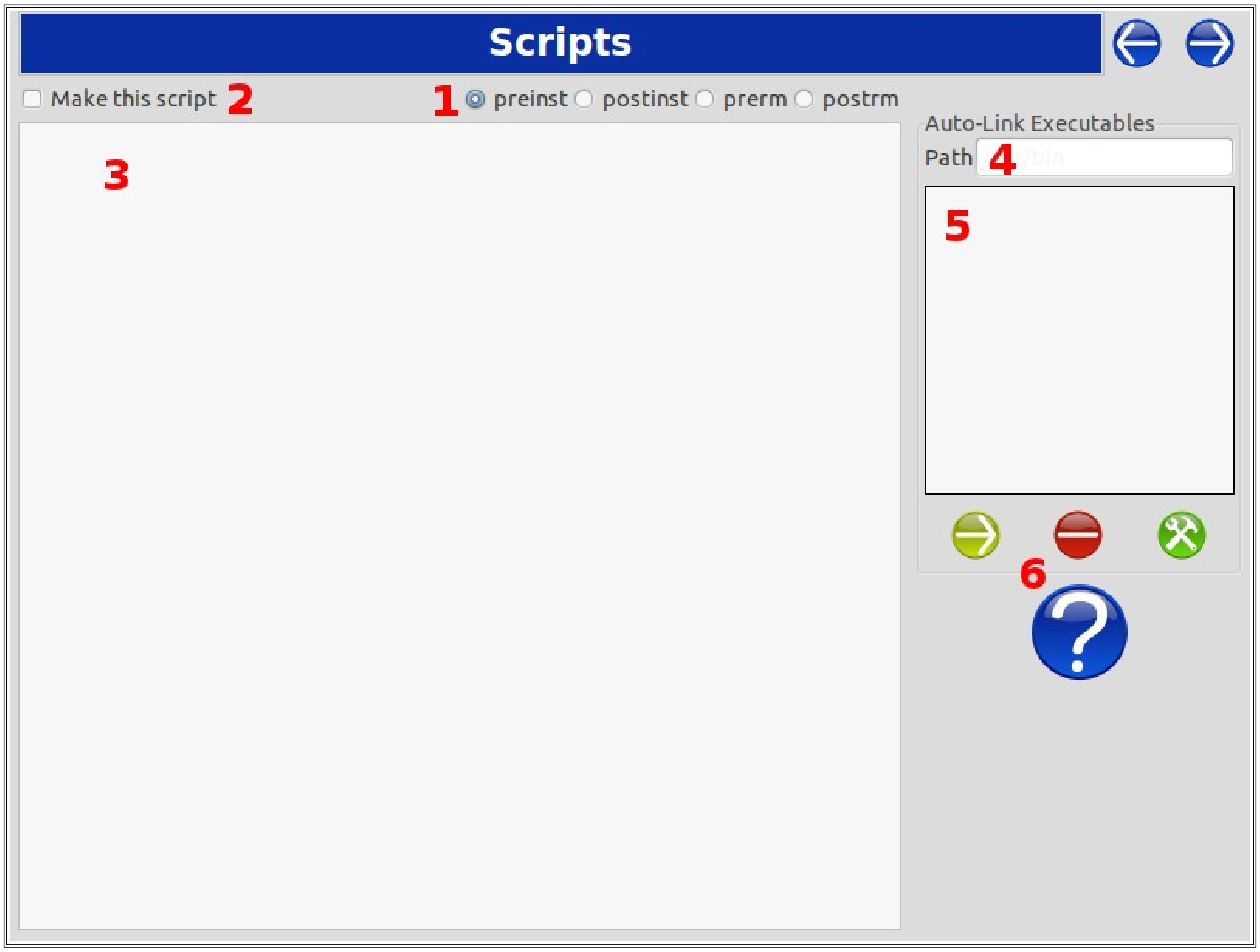
Displays the base file names for files to be included in the package and their destination on the system.

Scripts

Maintainer scripts are executables that can be called during the installation and/or removal processes. The four possible scripts are named “preinst”, “postinst”, “prerm”, and “postrm”. The first line of each script must start with a shebang (#!) and specify the path to the proper interpreter. For example, if the script is a bash script the first line would be `#! /bin/bash`. It is also recommended to set error checking (`set -e`). The preinst script runs before the package is unpacked and configured, postinst runs after, the prerm script is run before the package is removed from the system, and postrm after.

Debreate offers an Auto-Link Executables feature. What this does is finds any executables in the Files section and creates a postinst script that will create soft links to them in the specified path. This is useful if you are installing executables to a directory that is not found in the system PATH but want to access it from the PATH. For example, if you install an executable “bar” to the directory “/usr/share/foo” in order to execute “bar” from a terminal you would have to type `/usr/share/foo/bar`. Auto-Link can be used to place a link to “bar” somewhere on the system path like “/usr/bin”. Then all that needs to be typed is `bar` to execute the program. Auto-Link also creates a prerm script that will delete the link upon removing the package.

Figure 2-5



1: Current Script

The currently selected script for editing.

2: Script Creation

Needs to be checked for each desired script to be created.

3: Script Text

Editable preview of the script.

4: Auto-Link Path

When Auto-Link is used, this path specifies to where the link should be installed.

5: Executables List

Lists the executables found in the Files section.

6: Buttons

Import: Imports executables from the files section into the executables list.

Remove: Removes the selected item from the executables list.

Generate Scripts: Makes the postinst and prerm script templates for creating links to executables.

Help: Displays instructions on using Auto-Link.

For more information on scripts:

Debian Policy Manual – Maintainer Scripts

Ubuntu Policy Manual – Maintainer Scripts

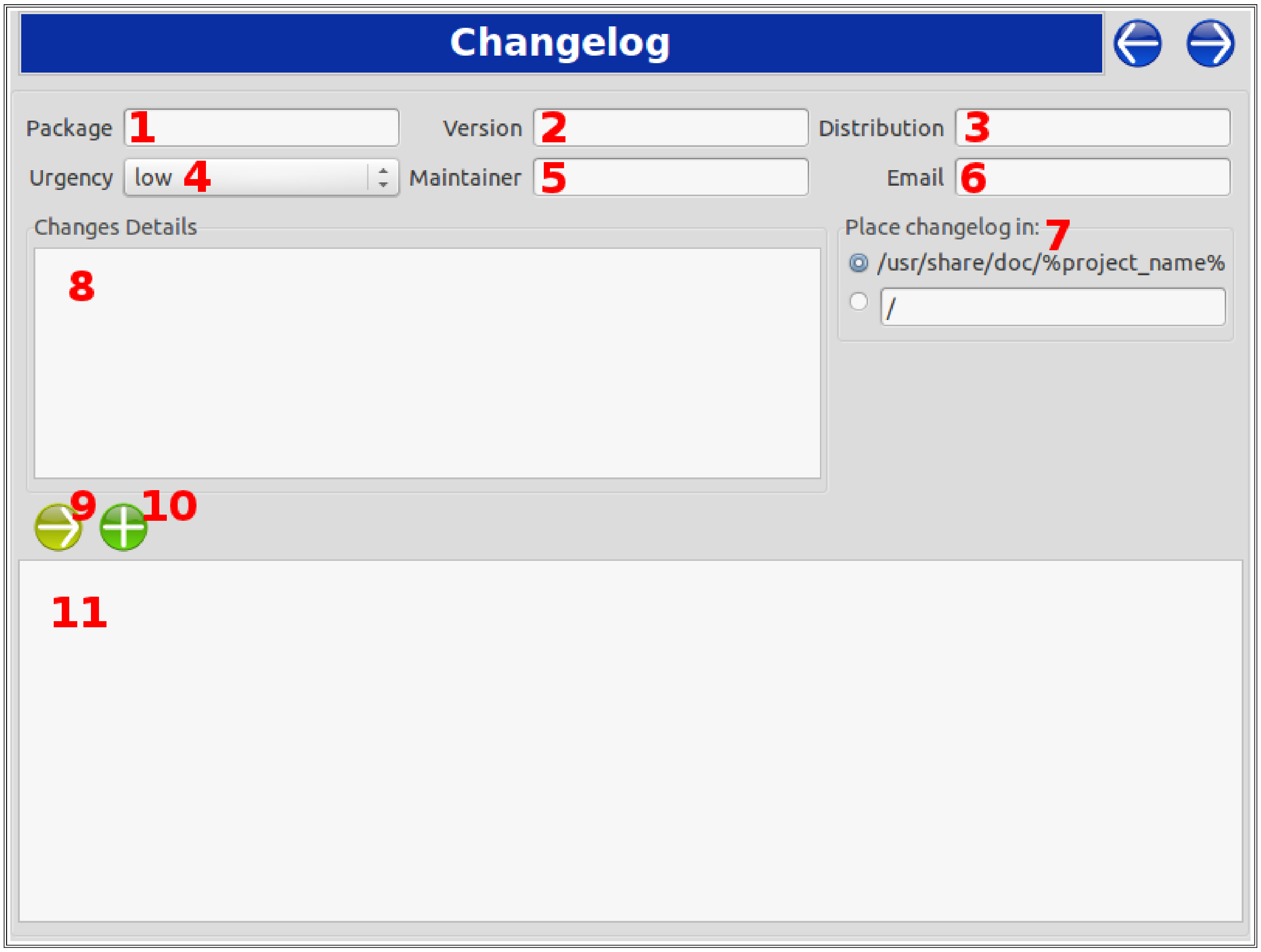
Changelog

The changelog is a documentation of changes across different versions. Debian has a specific format to be used and Debreate utilizes this section to make the process easier.

Changelogs are gzip compressed text files containing the information on the package's changes. This file is usually located in "/usr/share/doc/*package*". If the package is not Debian native it should have a compressed copy of the changelog found in "debian/changelog" from the source package. This changelog will have "Debian" in the filename. So it would be called `changelog.Debian.gz`. Currently Debreate does not offer inclusion of the source's changelog. For this to be achieved the file must be compressed manually and included in the Files section.

Changelogs can also be made available in the form of gzip compressed HTML files, but Debreate does not support this.

Figure 2-6



1: Package

The name of the package.

2: Version

Version information of the package.

3: Distribution

The name of the distribution for which the package is intended. For example, “squeeze” would be used if the package was intended for Debian 6.0.3 as this is its distribution name.

4: Urgency

Indicates the level of urgency to upgrade to the said version.

5: Maintainer

Package maintainer's name.

6: Email

Package maintainer's email address.

7: Target

Target destination for changelog file. According to the policy manual changelog files should be kept in /usr/share/doc/*package* (referred to as “%project_name%” in debreate).

8: Changes

Changes can be listed here separated by line breaks. If there are too many characters on a single line lintian will return a warning in the build process.

9: Import Button

Imports Package, Version, Maintainer, and Email from the Control section.

10: Add Button

Prepends the information from the above fields to the preview. It will automatically put the preview in Debian changelog format including the date/time stamp.

11: Preview

Editable preview of the contents of the changelog.

For more information on changelogs:

Debian Policy Manual – Changelog files & Debian changelog

Ubuntu Policy Manual – Changelog files & Ubuntu changelog

Copyright

The Copyright section is simply a plain text editor. Here can be declared the entire copyright license or a copyright declaration with the path to a common license available on the system. Here is an example of a declaration and common license path:

```
Copyright 2011 John Smith


/usr/share/common-licenses/GPL-3
```

For more information on copyright files:

Debian Policy Manual – Copyright information

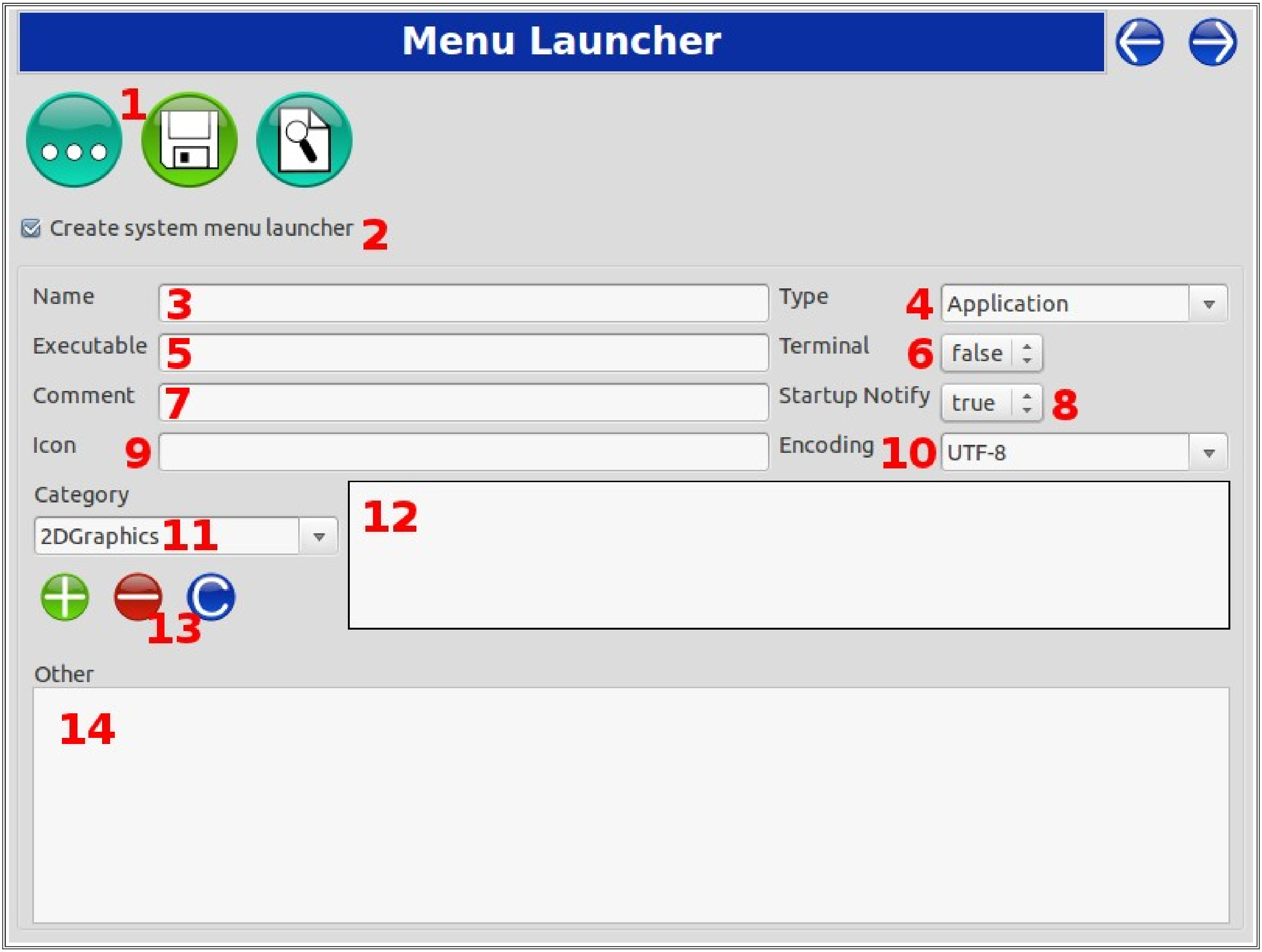
Ubuntu Policy Manual – Copyright information

Menu Launcher

Menu launchers can be found in the main menu of many desktop environment such as Gnome, Kde, and Xfce. These menu entries are normally based on text files located in /usr/share/applications. These text files use the extension ".desktop" and are often referred to as desktop files.

These desktop files contain information about what the launcher should look like and how it should act. Debreate offers a section to create a single launcher to be installed to /usr/share/applications. If more are desired they must be created individually and included in the Files section. This section of Debreate can be used to create these individual desktop files/launchers with a little bit of editing. Use the save button to export the information into a text file. Open the file in a text editor and put `[Desktop Entry]` as the first line and save it. You can optionally rename the file to have a .desktop extension, but I do not think that it is necessary.

Figure 2-7



Type: Application
Terminal: false
Startup Notify: true
Encoding: UTF-8
Category: 2DGraphics

1: Buttons

Browse: Open a text file that is formatted like a desktop file.

Save: Save the current information displayed to a text file.

Preview: Display a preview of the contents of the launcher/desktop file.

2: Create Menu Launcher

Must be checked to create and include a menu launcher in /usr/share/applications.

3: Name

The text that will be displayed on the launcher in the main menu.

4: Type

This is the desired type of launcher. The options are Application, Link, FSDevice, and Directory. An application launcher will launch an executable. A link will open a URL in a web browser (the URL key must be specified in the “Anything else I forgot” field). And a directory launcher will open a specified directory in the default file manager. ***I'm not sure that FSDevice is a legitimate value for this option. I will be looking into it to see if it needs to be removed.***

5: Executable

The program or command to be launched if the Type is Application. If the Type is Directory I believe this should be the path to the directory but I am unsure.

6: Terminal

Specifies whether or not the application should be run from a terminal.

7: Comment

Text that will be displayed in a bubble when the mouse is hovered over the launcher.

8: Startup Notify

If the desktop environment supports it, a notification that the application is starting will display in the panel.

9: Icon

The path to a bitmap to be set as the icon for the launcher. If the bitmap is located in /usr/share/pixmaps the absolute path is not necessary and the filename alone may be specified.

10: Encoding

This option is deprecated and will be removed. From freedesktop.org:

```
The Encoding key is deprecated. It was used to specify whether keys of type
localestring were encoded in UTF-8 or in the specified locale.
```

11: Category

Used to organize launchers in the main menu. For example, if you want your launcher to appear in the Internet sub-menu you would want to add either “Network” or “WebBrowser” to the categories. Debreate offers a list of available categories but the list is not complete. This is also customizable and does not require the use of one of the available options.

12: Category List

Displays all the categories that have been selected.

13: Category Buttons

Add: Adds the text from the category option to the category list.

Remove: Removes the selected item from the category list.

Clear: Clears the category list.

14: Other

This is a plain text section where any keys that Debreate does not specify can be declared. It can be useful for adding locale options to the launcher's name or setting a mimetype for the application. Keys are declared using a *key=value* format. Values can be a string, localestring, boolean, or numeric. Here is a list of optional keys and their value type for which Debreate does not have a field:

`MimeType`	**`string(s)`**
`GenericName`	**`localestring`**
`NoDisplay`	**`boolean`**
`Hidden`	**`boolean`**
`OnlyShowIn`	**`string(s)`**
`NotShowIn`	**`string(s)`**
`TryExec`	**`string`**
`Path`	**`string`**
`StartupWMClass`	**`string`**
`URL`	**`string`**

For more information on keys: Recognized desktop entry keys

For more information on launchers:

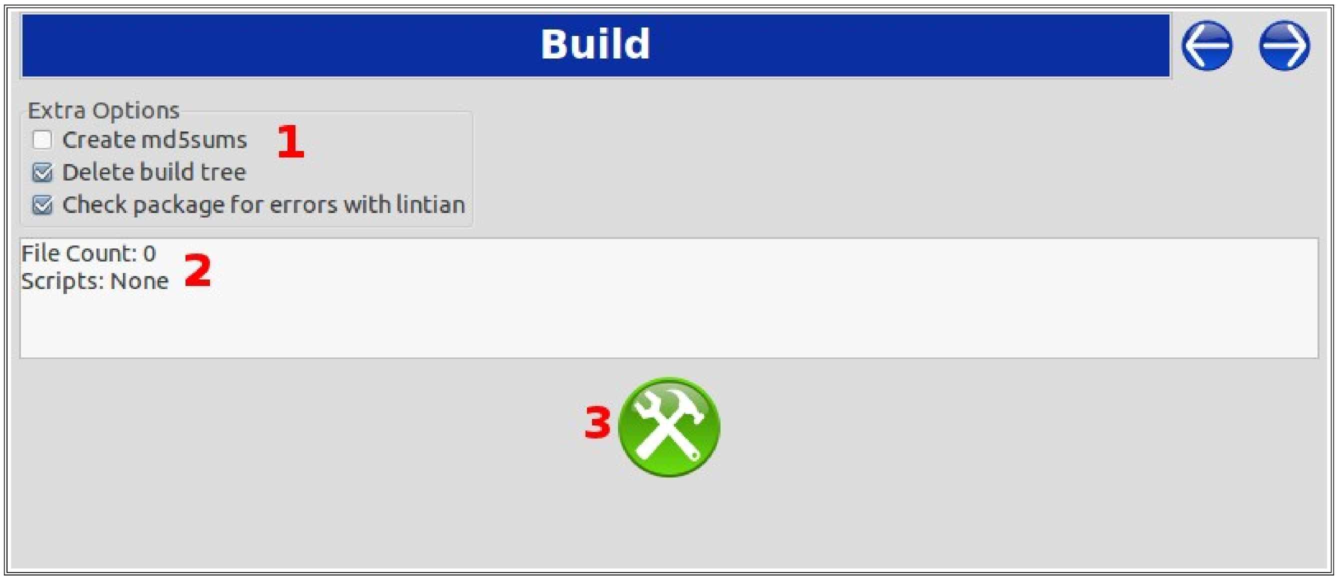
freedesktop.org – Desktop Entry Specification

freedesktop.org – Desktop Menu Specification

Build

Here is where the actual building of the package begins. Once the "Build" button is pressed Debreate will gather all of the information input into each section and start the process.

Figure 2-8



1: Extra Options

md5checksum: Option to create a file that contains a checksum for each file contained within the deb. This file can be used to check the integrity of the installed files using a program called debsums.

Delete build tree: Debreate will create a file system hierarchy using the files specified in the Files section to build the package. Checking this options ensures that this temporary directory structure is deleted once the package is built.

Lintian: If selected the lintian program will scan the built deb package for any problems. If any are found a dialog will pop up displaying warnings and/or errors. Information on these warnings and errors can be found on the Lintian Tags explanation page.

Note: Warnings and errors displayed by lintian does not necessarily mean that the package will not work correctly.

2: Summary

Displays how many files are going to be added to the package and which scripts will be created.

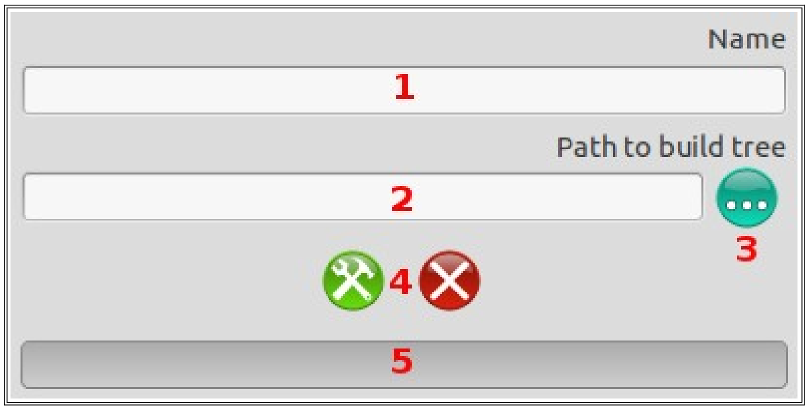
3: Build Button

Starts the build process. When pressed it will make sure all of the required fields are filled out. Then a dialog will open asking for a name to give the deb package. The default name is recommended. Press save and the build process will begin.

<u>Quick Build</u>

From the File menu there is an option called Quick Build. This will bring up a small dialog that allows the user to build a package from a preexisting build tree. This would be the same as navigating to the folder from a terminal and invoking `dpkg-deb -b`. From the dialog enter the desired filename for the package in the field named "Name" (if this is left empty the name of the build path folder is used). Enter the path into the field named "Path to build tree". Press the 🛠 button to build the package.

Figure 2-9



1: Name

The output filename of the package.

2: Path to Build Tree

The location of the root folder of the directory tree.

3: Browse Button

Add a directory "Path to build tree" field.

4: Buttons

Start build: Begin building the package.

Cancel: Close the dialog.

5: Status Bar

Displays status of build process

Section 3

Credits

This document is written by Jordan Irwin.

Last updated November 24, 2011

License

Reference

Debian Policy Manual

Ubuntu Policy Manual

Debreate Website