PiSiDo Help

Packaging Basics

A pisi package can be build by using “pisi build” command. Pisi build command takes a pspec.xml file as an argument, and creates a binary install file. While giving a pspec.xml file, there should be some other files to build a package. Here is the layout of the the build files :

```
<package_name>
  |__ actions.py
  |__ pspec.xml
  |__ translations.xml
  |__ files
  |__ comar
     |__ packages.py
     |__ service.py
```

<package_name>, files and comar are directories. Let's explain each of these files:

<package_name> : These is no special mean at the package_name, just a directory that holds our build files.

actions.py : This file defines configure, compile and install steps. It uses Actions API to define what will do on each step. PiSiDo will provide you some templates like autotools, cmake, kde4, qt4, etc. For more information refer to the Pardus developer pages.

pspec.xml : This file defines the package information. It holds package source, package detailed information and a history of the build files. The file have a RNG rule file. You can find it from here. For more information refer to the Pardus developer pages.

translations.xml : This is an optional file. This file defines the package summary and description translations. PiSiDo does not produce a translation file for now ! Here is an example transltion file :

```
<?xml version="1.0" ?>
<PISI>
    <Source>
        <Name>pisido</Name>
        <Summary xml:lang="tr">Pisi paketi yapıcı</Summary>
        <Description xml:lang="tr">Grafik arabirim ile basit pisi paketleri
yapabilirsiniz.</Description>
    </Source>
</PISI>
```

files : This is an optional directory. You can add your aditional files that is not come with the source archive. Also, you can add patches to this directory. Each file under here should informed in the pspec.xml file to process. PiSiDo provides graphical tools to make this operation as easy as possible.

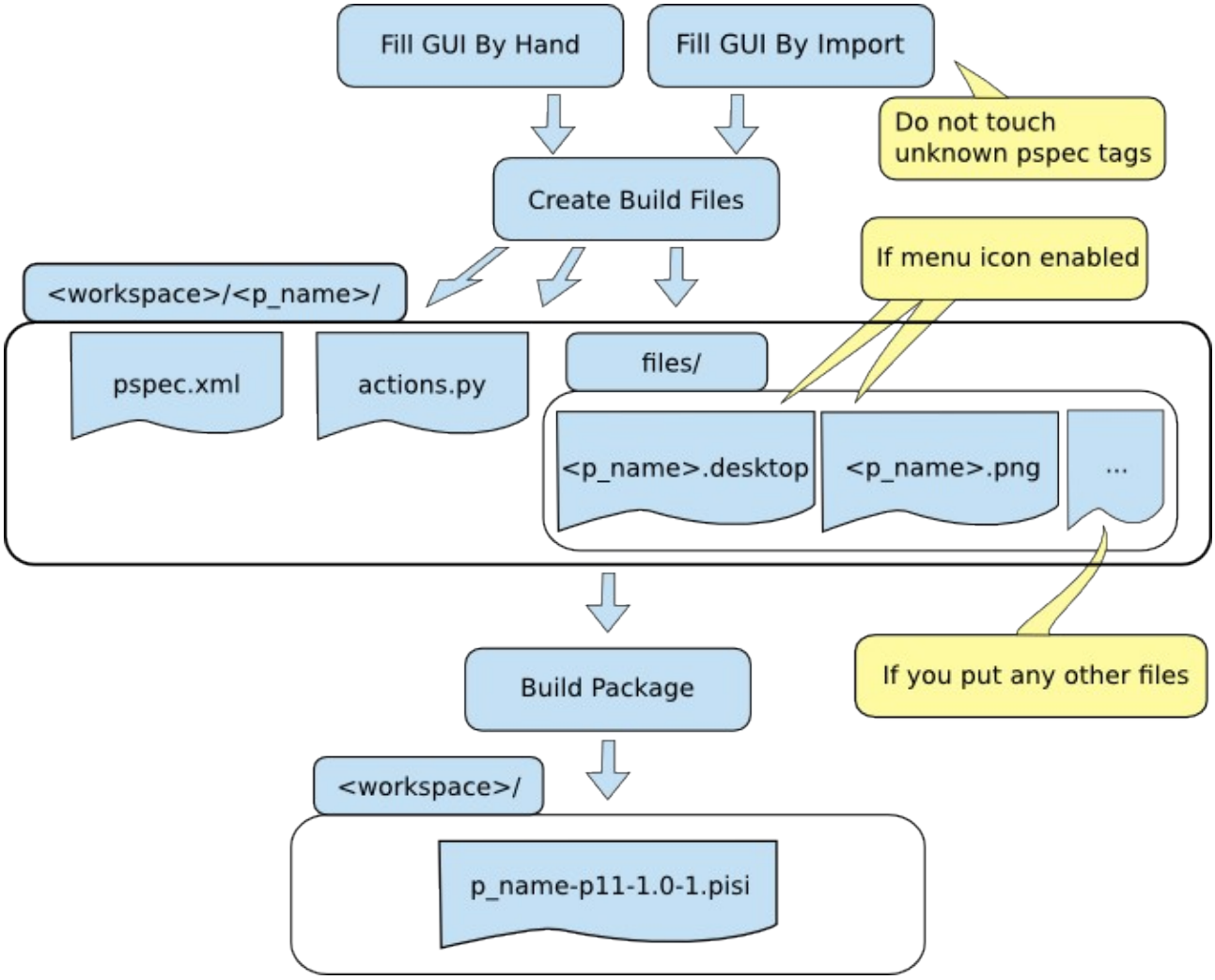
comar : This is an optional directory, too. This directory hols COMAR (COnfiguration MAnageR) files. These files help Pardus to define service operations. Which is out of PiSiDo's scope, and PiSiDo does not create this directory and its contents. For more information refer to the Pardus developer pages.

To make a pisi package you must create an actions.py and a pspec.xml file atleast. Then you can build it with this command :

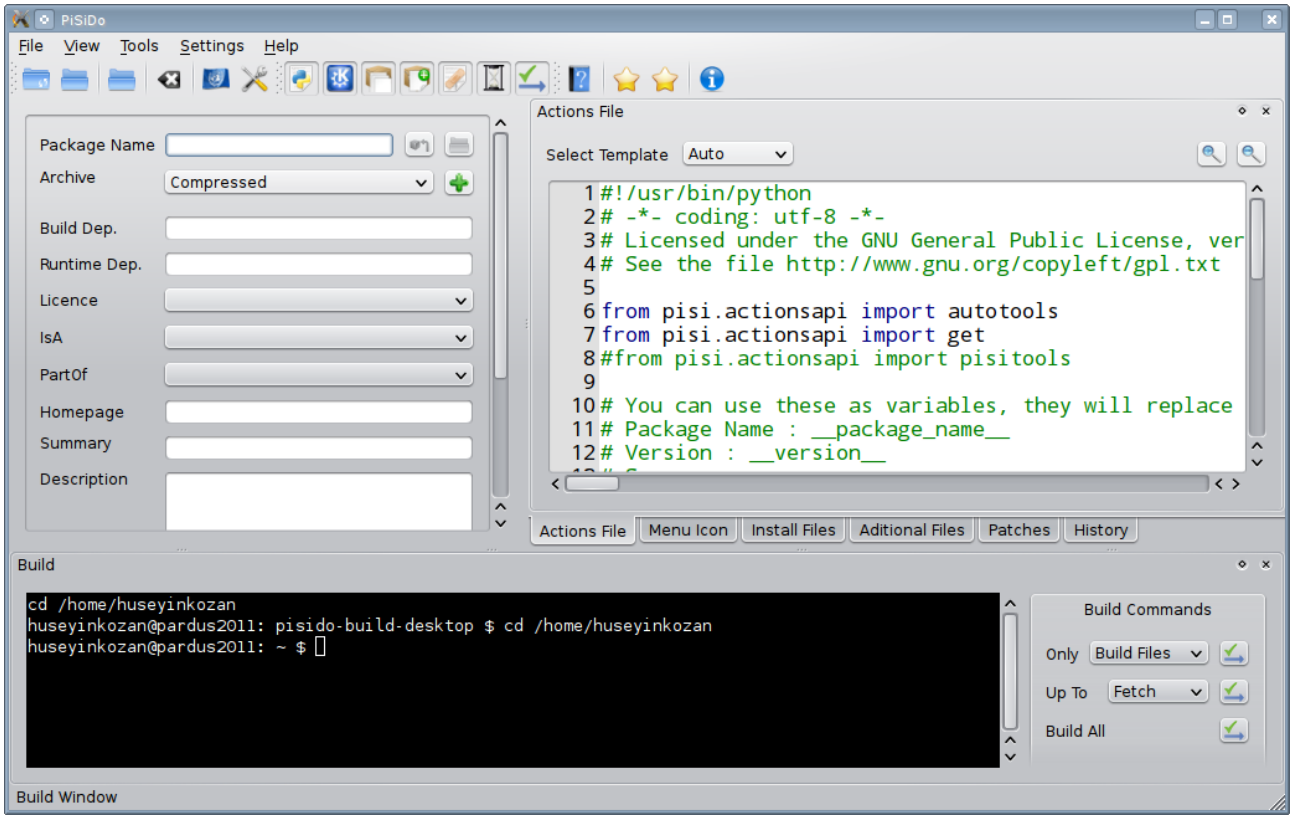
```
$ sudo pisi build pspec.xml
```

You can find some valuable information on the unofficial Pardus Wiki. Official documentations are on the Pardus developer pages.

How PiSiDo Make Package



PiSiDo GUI



Central Fields

Package Name:

Name of the package. This field will be used to create a same named directory under the workspace. On each character you type, application checks for the existing packages. And you can import an existing package by pressing Enter.

Archive:

Here, you can define the source of the package. The selection box have two values:

1. Compressed: For local files. Application compute the sha1 value of the archive after you add it.
2. Url: For local, and global files. You must fill archive path and sha1 value.

Build Dep.:

Here, you can define build dependency packages. Pisi build system will install before building the package. You can write more than one dependency separated each with a comma. Also, you can limit version by writing these special parameters:

- qt-devel[>4.7] : qt package version must be bigger than 4.7
- qt-devel[=4.7] : qt-devel package version must be equal to 4.7
- qt-devel[<4.7] : qt-devel package version must be less than 4.7
- qt-devel[>>4.7] : qt-devel package release must be bigger than 4.7
- qt-devel[==4.7] : qt-devel package release must be equal to 4.7

- qt-devel[<<4.7] : qt-devel package release must be less than 4.7

For example: qt-devel[>4.7], gtk2-devel[2.20]

Runtime Dep.:

Here, you can define run time dependency packages. Pisi will install before installing the package. You can write more than one dependency separated each with a comma. Also, you can limit version by writing these special parameters:

- qt[>4.7] : qt package version must be bigger than 4.7
- qt[=4.7] : qt package version must be equal to 4.7
- qt[<4.7] : qt package version must be less than 4.7
- qt[>>4.7] : qt package release must be bigger than 4.7
- qt[==4.7] : qt package release must be equal to 4.7
- qt[<<4.7] : qt package release must be less than 4.7

For example: qt[>4.7], gtk2[2.20]

License:

Here, you can select license type of your package.

IsA:

Here, you can select package type. Like application, service, localization, etc.

PartOf:

Here, you can define component of the package. Like desktop, game, hardware, etc.

Homepage:

Here, you can define homepage of the package.

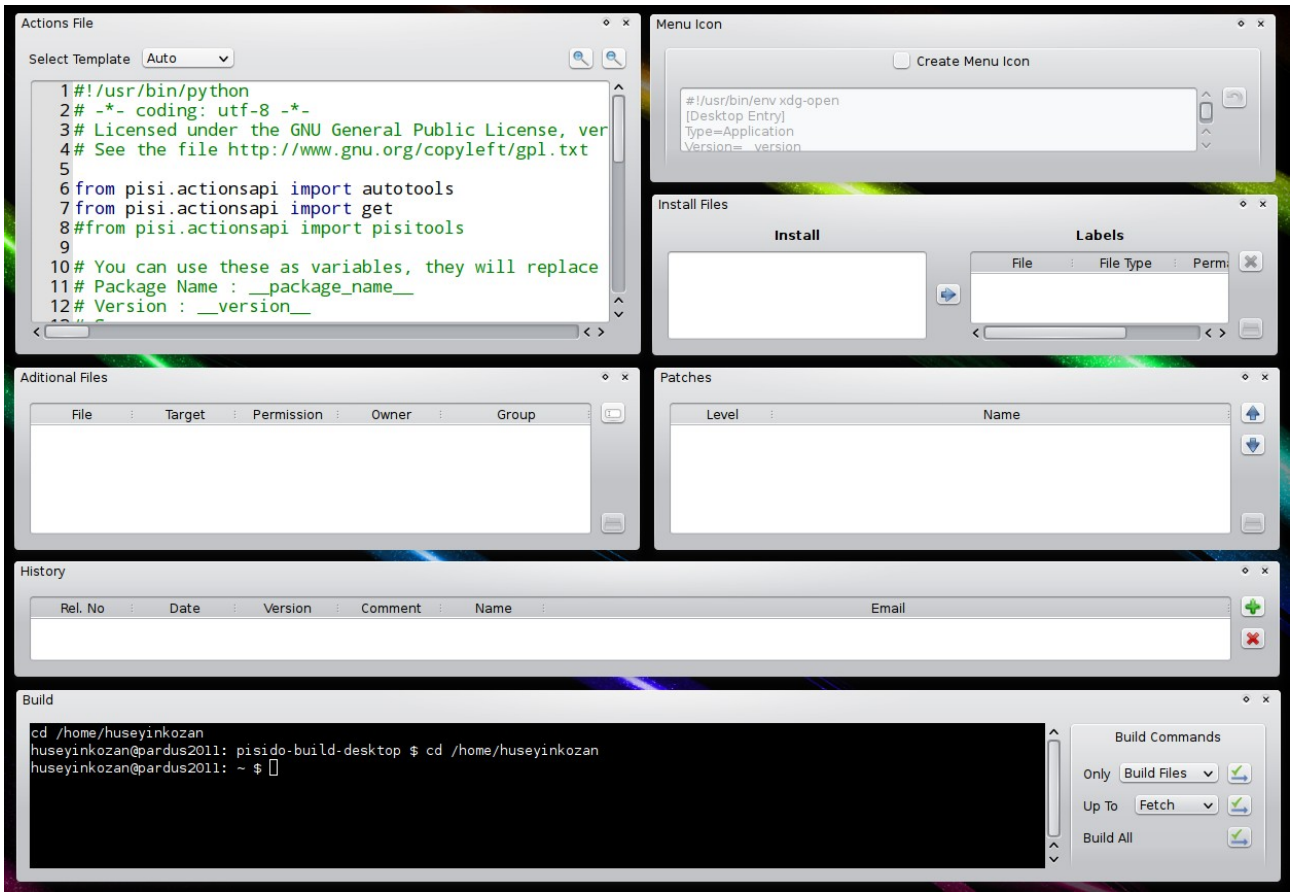
Summary:

Here, you can define a short description. This will shown to the user while listing the packages.

Description:

Here, you can define a detailed description.

Dock Windows



Actions File

Actions file is using for the managing the build process by pisi build system. You can change templates with selection box. Actions file uses python and Actions API. For more information about the Actions API refer to the Pardus developer pages.

Actions file templates have some comments to help user. Also, there are some special variables which are PiSiDo specific. These variables will replace proper values while creating the build files. Special variables are :

- __package_name__ : replaces Package Name field
- __version__ : replaces last update release number from History Window
- __summary__ : replaces Summary field

Warning ! : Your changes will not save, until you create the build file. Creation of the build file is automatically done by any commands that are on the Build Window.

Menu Icon

This is an optional field. After enabling, application will create a desktop file and a png file under *files* directory. You can change the png file, if you want. You must change the save location of these files from *Aditional Files* window to show a menu image to the user. Here is an example of the locations:

`<package_name>.desktop file : /usr/share/applications/<package_name>.desktop`

`<package_name>.desktop file : /usr/share/pixmaps/<package_name>.png`

If you define Package Name, than <package_name> values will automatically added while creating these files.

Install Files

Here, you can define where they will install each installable file. *Install* list will filled after a successful build. Application watches packages pisi build directory for the changes. If you start a build and it fails, the file listing will be cleared.

As a required field, there is a default label which defines the root (/) of the file system for the all installing files in the package. You should add the proper locations for the files or folders and delete the default label.

Aditional Files

This is an optional field. Application watches for the file changes under <workspace>/<package_name>/files/ directory. You can manually add new files to this directory and define where it will goes during the installation in this window. If you enabled Menu Icon, a desktop and a png file will be created under *files* directory and lists here. You should define where they will go during the installation. A brief description gave in the Menu Icon.

Patches

This is an optional field. Application watches for the file changes under <workspace>/<package_name>/files/ directory. You can manually add new patches (*.patch) to this directory and define the patching order from this window.

History

This fields shows the history of the package build files. It is defined in the pspec.xml file by the packager. You can add new update information. Also note that, you can define packager information from Application Configuration.

Build

With this window, you can give some commands to build your package. Here is the list of supported commands :

- Only
  - Build Files : Create build files and stop.
  - Package : Create build files and trys to package a successful build. This is useful changing actions.py and not need to recompile the package. This can be think as a last step of Up To.
- Up To
  - Fetch : Create build files and gives `pisi build pspec.xml –fetch` command. This will download source archives and stop.

- Unpack : Apply Fetch step and checks sha1 value of the source archive, and unpack it, and apply patches, and stop.
- Setup : Apply Unpack step and apply setup (configure) step and stop.
- Build : Apply Setup step and apply build (compile) step and stop.
- Install : Apply Build step and apply install step and stop.
- Check : Apply Install step and checks package and stop.

For more information refer to the Pardus developer pages.

PiSiDo Menus

File

Change Workspace

Changes workspace. Workspace is the main directory that application saves the package build files and builded pisi files. You can enable or disable opening the workspace dialog from itself.

Reset Fields

Clears all graphical interface fields.

Exit

Exits program. Application will not warn you for unsaved changes.

View

You can show and hide each dock window and toolbar in this menu.

Tools

Open Workspace

Opens workspace that you choose while starting the application with a file manager.

Open PISI Packaging Directory

Opens pisi working directory which is located under */var/pisi* with a file manager. You can change this path from Settings.

Settings

Application Language

You can change the default application language from here. Change will effect after restart the application.

Configure Application

You can change application specific settings from here. You can define packager information which will assist you while adding a update to the history. You can change pisi packaging directory and Actions API page and PISI Spec file path.

Help

Help...

Opens proper help file through selected language.

PISI Spec

Opens PISI Spec RNG file. This value can change from Application Configuration.

Actions API

Opens Actions API page. This value can change from Application Configuration.

About...

Shows application information.

About Qt...

Shows Qt toolkit information.