Specifications

Project : OBS Light

Document status : Draft 0.6

Authors :

- Ronan Le Martret
- Gustav Hellmann

Reviewer:

- Dominique Le Foll

Abstract:

TV Integrators are coming from the embedded world and are more comfortable with the tradition of local build tools than a with a full cloud based system like OBS.

The project's propose is to create and to publish a OBS light which can be delivered with a packaged MeeGo TV (or any other MeeGo vertical) release on a DVD image.

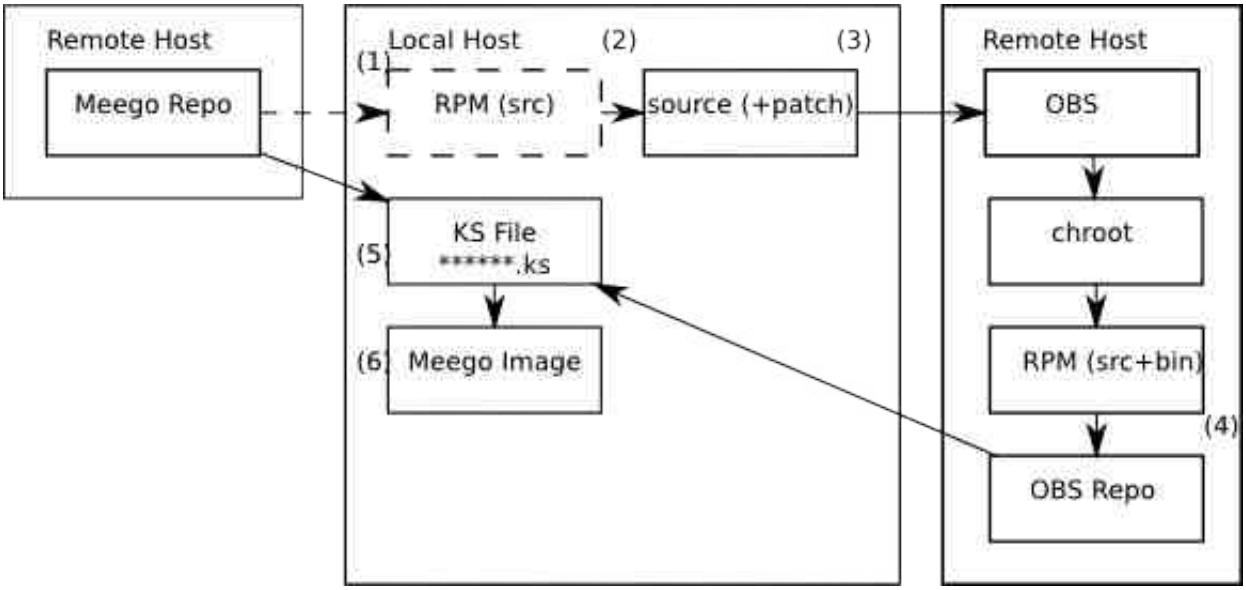
It is based on an extension of the osc command line tools delivered with OBS and on an extension of mic2.

The same solution could be reused by any embedded MeeGo project (e.g. IVI).

Name	MeeGo ID	Date	Operation on the document	Version
Ronan Le Martret	ronan	10/06/11	First publication	0,6
Gustav Hellmann	Gustav	10/06/11	review	0,6

Table of contents

1 Current State



We present here the current way to manage a MeeGo build project.

1. (Facultative) The user integrates a RPM source file into his project.
2. (Facultative) The user expands the RPM source file.
3. The user integrates the new sources into the OBS system and selects a build target.
4. OBS builds from the (new) sources the source RPM and the binary RPM.
5. The user updates or creates the kickstart file (*.ks) and links to the OBS Repo and the MeeGo Repo the correct target.
6. The user generates a MeeGo image with MIC2 with a script.

To (1) :

The user checks out from the MeeGo repository a RPM file containing the source files where he would like to add a patch which takes into account the particular technical characteristics of his MeeGo deployment platform (netbook, ivi device, etc) or allows a better exploitation of these characteristics. For a new or an outstanding MeeGo deployment platform the user will provide his own sources.

To (2) :

In the case that the user did a checkout of an RPM file containing the source files from the MeeGo repository, the user will decompress the obtained RPM file and add a patch to the sources.

To (3) :

The modified source files are added to a package in the OBS system.

To (4) :

An OBS build is triggered. If the build is successful, the user will find the new RPM source files as well as the target RPM binary files in the local OBS repository.

To (5) :

The kickstart file *.ks will now need packages respectively package groups from both the MeeGo and the OBS repositories. It should be updated in order to take into account the new RPM files generated in (4). If necessary, the user can create a completely new *.ks file.

To (6) :

A MeeGo image based on the updated kickstart file *.ks is created as explained in (5).

2 Objective

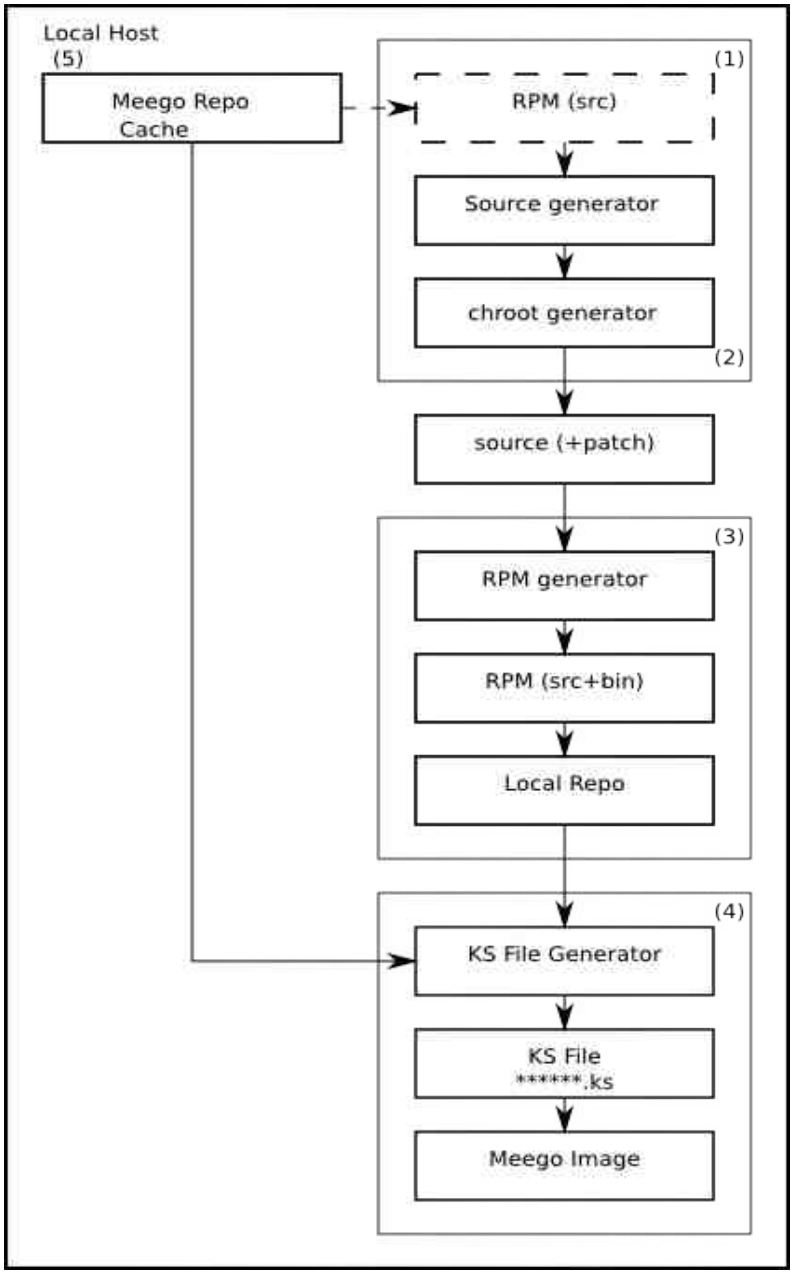
We present here the objective to be attained at the end of the OBS light project.

The solution must not use remote OBS but still retains compatibility with it.

The install of the full OBS Light development kit, include tools and a distribution's MeeGo.

The working method is classical :chroot, command line, GUI ...

The RPM 's creation is transparent.



1. The user is able to generate a package from a RPM source. Alternatively, the user can create

himself his own source for his package:

- The user must specify within the project:
  - A ***ProjectTarget***.
  - A ***ProjectArchitecture***.
- A ***chrootDirectory*** is created.
- The user must select one or more ***RPMFile*** source, that he wants to modify (equivalent to the package structure on OBS).
- The “source generator” should:
  - Expand the ***RPMFile*** source file into a [***RPMName***] folder in the ***ProjectDirectory***.
  - All the dependencies of the ***RPMFile*** are expanded too, each into their own [***RPMName***] folder in the ***ProjectDirectory***.

    All the ***RPMFile*** are installed into ***chrootDirectory*** and built within it.
  - All the information about the files are stored in a data structure. ***SourceDataFile***.
- The user can work as he wants within the ***chrootDirectory.***

2. If a file changes:
   - The sources in the ***chrootDirectory*** are compared to the sources in the ***ProjectDirectory***, thanks to the ***SourceDataFile***.
   - A patch is created in the [***RPMName***] folder in the ***ProjectDirectory***.
   - If a file is created in the chroot, the user chooses the [***RPMName***] folder in the ***ProjectDirectory***.
   - The *.spec is updated.

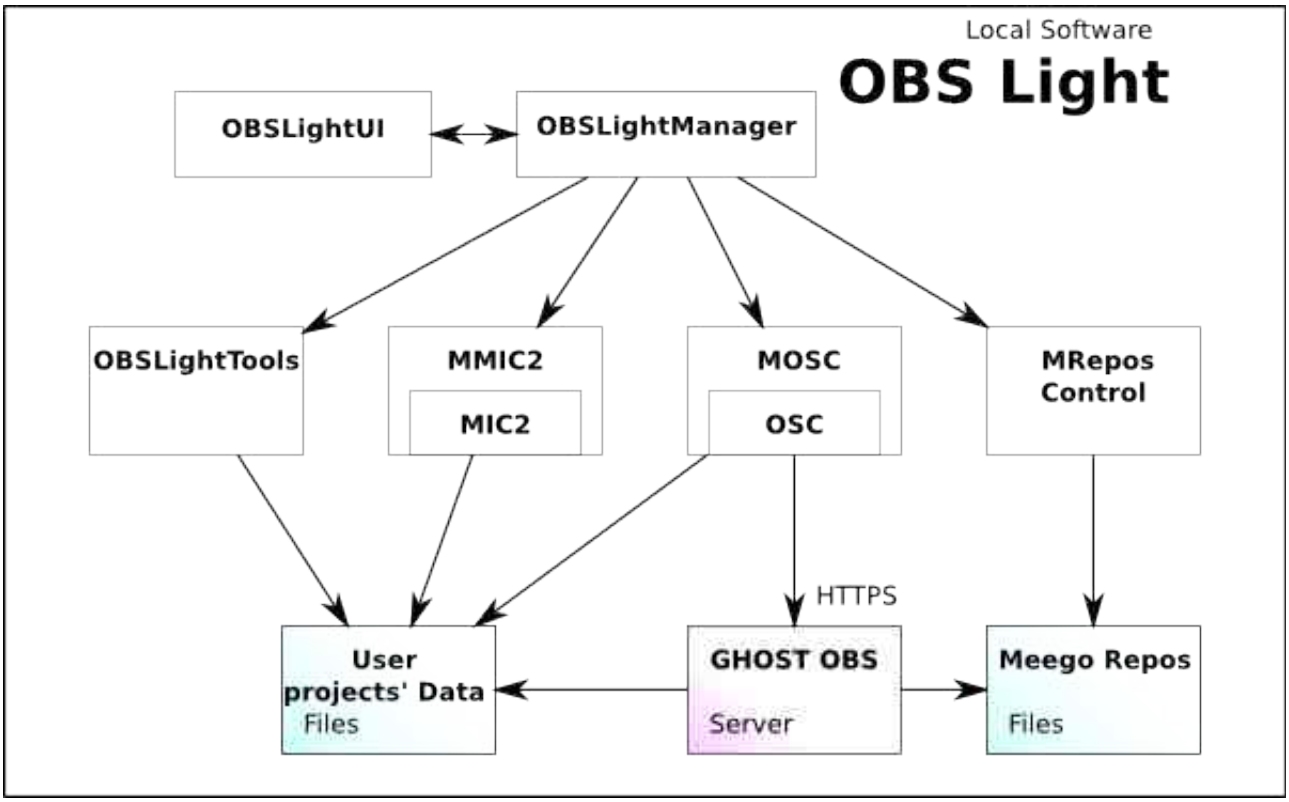
3. The user is able to generate the RPM sources and the RPM binary files:
   - If any change:
   - A RPM source is built.
   - The ***RPMPackageVersion*** is incremented.
   - The RPM bin is built.
   - The RPM, source and binary, are copied to the ***ProjectRepos***.
   - The change Log is expanded.
   - Version management is left to the discretion of the user.

4. The user is able to generate an image from ***ProjectRepos***:
    - The user can select all the RPM from the package.
    - The user can see all the RPM' s dependencies tree for a RPM' s master.
    - The user can see all the RPM' s master tree for a RPM' s dependency.
    - The user can choose among many ***RPMPackageVersion***.
    - The user can see all the RPM of ***ProjectRepos***.
    - The user can choose RPM by group.
    - The user can see all the selected RPM.
    - A KS file can be generated.
    - The user can choose the ***ImageType***.
    - The “KS file generator” is able to create an image file.

5. The user is able to update or create a local MeeGo repos.
   To be loaded, the target simply needs to be pointing to used that repo.
   By opposition to OBS there is no signing service.

3 Flow Diagram

Global flow diagram



OBSLight

Language:

- Python 2.7.
- QT 4.x for OS abscraction

OBSLightUI:

- Does nothing except communicating with the **OBSLightManager**.
- Can run in stand alone mode, without **OBSLightManager**.
- **OBSLightUI:** Qt, Pyside**.**

OBSLightManager: **OBSLightManager** Manages all the software.

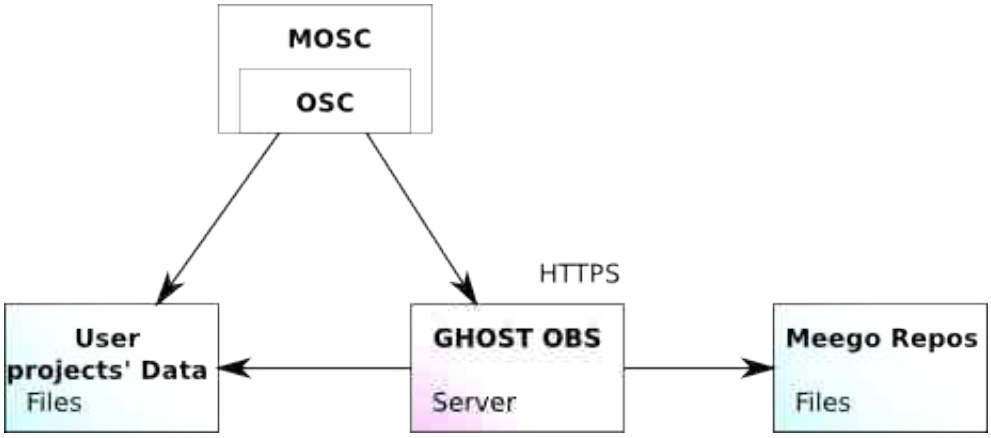
- Knows everything the software needs and serves as dispatcher of the tasks to accomplish.
- Receives information from the **OBSLightUI.**

- Transmits the information to display to the **OBSLightUI.**
- Can command and control **OBSLTools**, **MMIC2**, **MOSC**, **MReposControl,** but does nothing on its own.

At the beginning of the OBS Light project, **MMIC2** uses the existing **MIC2**, however we could enhance the **MIC2** code later if advantageous.

MOSC:Master OSC.

- Controls **OSC**.
- Knows nothing.
- Can only run script and function.
- Every function is based on a dedicated python script.
- Solves objectives (3) generates the RPM.



MMIC2: Master MIC2.

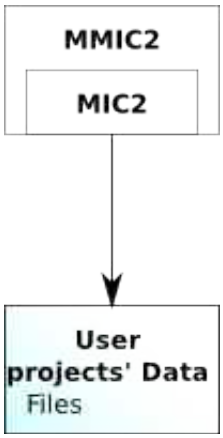
- Controls **MIC2**.
- Knows nothing.
- Can only run a script or a function.
- Every function is based on a dedicated python script.
- Solves the objective (4) generates an image.

At the beginning of the OBS Light project, **MOSC** uses the fixed **OSC**, however we can change the **OSC** code later if advantageous.

MRepos Control: Master Repos Control.

- Controls the **MeeGo Repos**.
- Knows nothing.
- Can only run script and function.

- Every function is based on a dedicated python script.



- Solves objectives (1) generate a package and (5) create a local MeeGo repos.

User projects' Data: All the data needed for the user.

- Source code.
- RPM (src, bin).
- All the files are XML files.
  - A backup file and a MD5 file are associated to the XML file.
- All the files are OBS compatible, if possible.

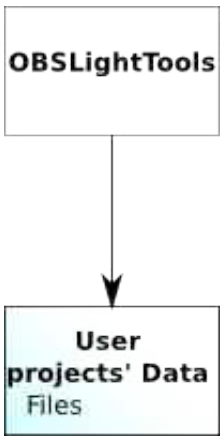
GHOST OBS: It's a fake OBS server used in order to keep the compatibility with the OBS without actually using full OBS.

- Can receive and transmit a XML file in the **OSC** format.
- Can read/write data in the OBS format from **Projects' System Data**.
- Not all **OSC**'s functions are implemented, only those useful for OBS Light.

MeeGo Repos: All the data for all the repos.

OBSLightTools:

- **Can synchronize between the file in chroot and the file in package (diff, patch, new, delete,...).**

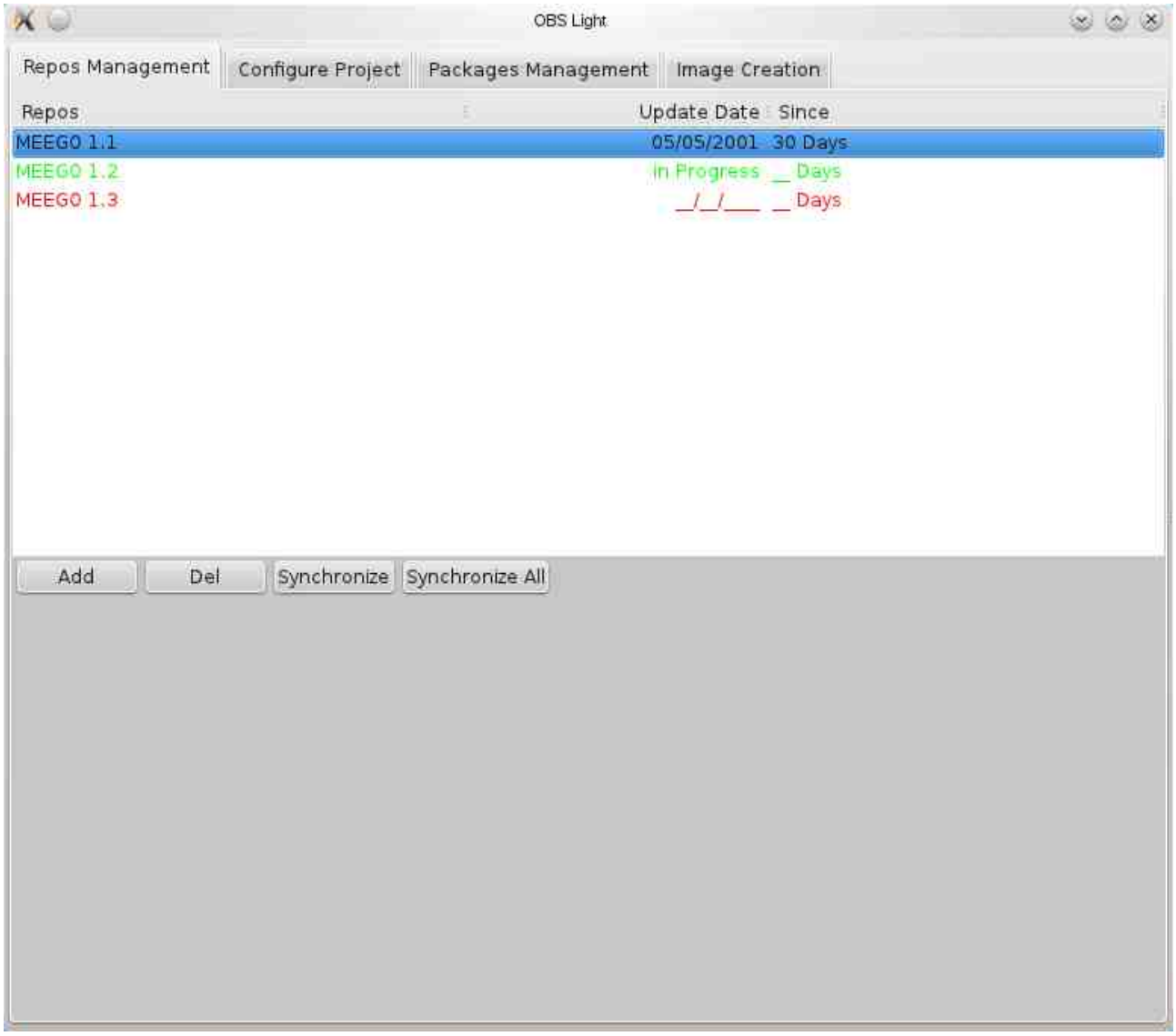

OBSLightTools
User projects' Data Files

4 User Interface

The user interface has 4 parts.

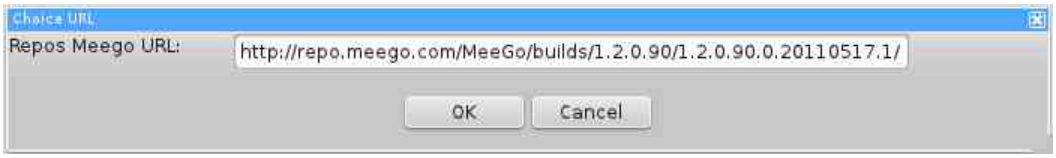
Note: UI images for information only and will be changed in real implementation.

1 Repos Management



What the user can do:

- By interface:
  - Select a local repository of OBS light.
- By action on a Button :



- **Add :** Import a distant MeeGo repository to the local repository of OBS light by doing copy/paste of its ***ReposURL*** and validating, in a mini Frame. (Note this is repo remote access not acess to a remote OBS API).
- **Del:** Delete a local repository of OBS light.
- **Synchronize :** Synchronize the selected local repository of OBS light with the distant MeeGo repository.
- **Synchronize All:** Synchronize all the local repositories of OBS light with the associated distant MeeGo repositories.

What the user can see:

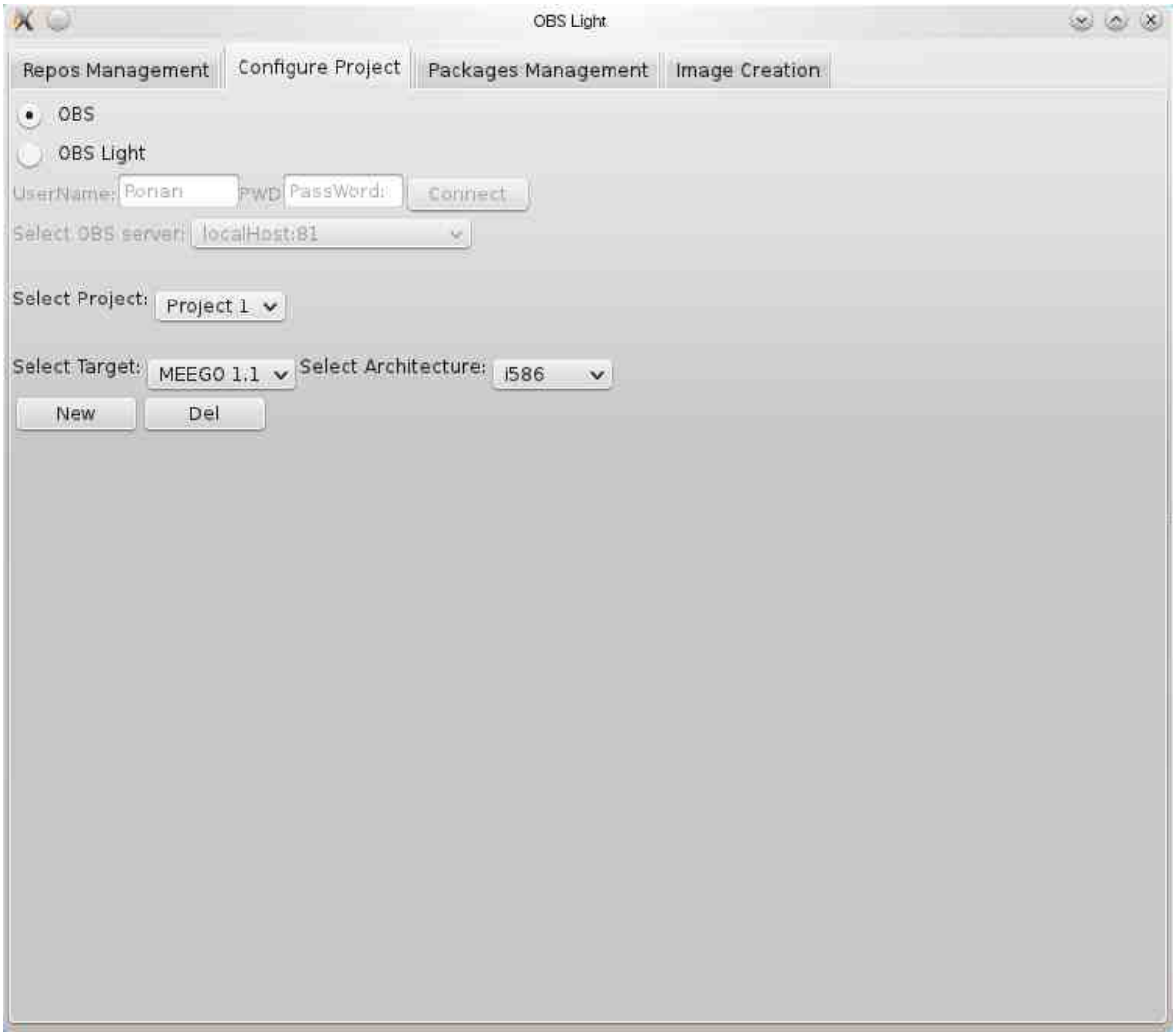
- The list of all local repositories of OBS light.
- The name of the local repository.
- The date of the last update.
- The number of days since the last update.

What the **OBSLightManager** must do:

- Load the list of local repositories at the start of OBS Light.
- Save the list of the local repository after any change.
- Add a local repository to the list of local repositories.
- Delete a local repository from the list of local repositories.
- Run an independent thread to synchronize a local repository with the distant MeeGo repository.
- The independent thread stores in a file the status of the synchronization.
- Check the status of the list of the local repositories at the start of OBS Light.

.

2 ***Configure Project:***



What the user can do:

- By radio button :
  - **OBS / OBS Light**: Select if the user works locally or with an OBS server. By default,the box is unchecked and the user works locally isOBSConnected=False.
- By action on Combo Box:
  - **Select Target**: Select the current build target by the repos name.
  - **Select Architecture**: Select the build architecture.
- By writing in a textfield:
  - **UserName**: Specify the user name.
  - **PWD**: Specify the user's password.
- By action on a Button:

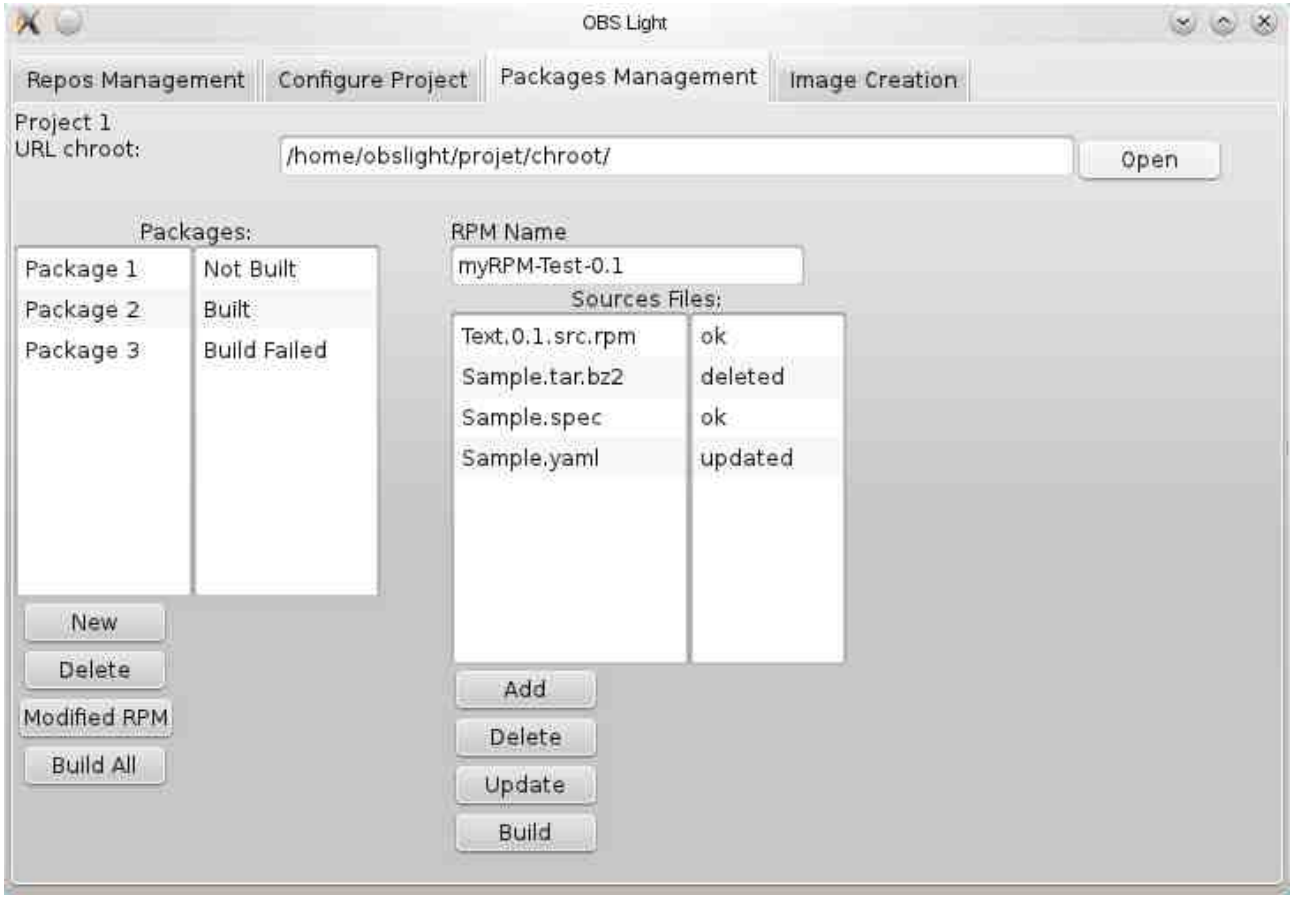
- **New**: Add a new project to the user's account. Project can only be created localy.
- **Del**: Delete a project from the user's account.
- **Connect**: If the user works with an OBS server, he can test the connection to the server.

- By action on a Combo Box:
  - **Select OBS server**: If the user work with an OBS server, he can choose it in the list of available servers: *[localHost:81,build.pub.MeeGo.com:81,build.MeeGo.com:81], by default localHost:81 is select.*
  - **Select Project**: If the user works locally , he can choose a local project. If the user works with an OBS server, he can choose a project after successfully connecting to the server via "**Connect**".

What the **OBSLightManager** must do:

- At the connection with an OBS server , the ***OBSLightManager*** gets access to the existing user's projects.
- If the OBS Light works locally, it reads the list of existing user's projects from a file.
- Following a change in a project, the manager stores the change.
- Following the user's selection of a project, the OBSLightManager loads it from a file.

3 Packages Management



What the user can see:

- The name of the current project.
- The names of all the Packages and their ***PackageStatus***.
- The ***SourcesFileName*** of all the ***SourcesFile*** of the selected ***Package***, and their ***PackageStatus***.

Note: All dependent package are loaded automatically. The user only has to select the packages that he wish to modify.

What the user can do:

- By write in textfield:
  - **RPM Name**: Write the name of the RPM.
- By action Button:

  1) Below the “Packages” window:
  - **New**: Create a new package.

- **Delete**: Delete the selected package.
- **Expand RPM**: Generate a package from a RPM source file.
- **Build All**: Build all the packages in the packages list.

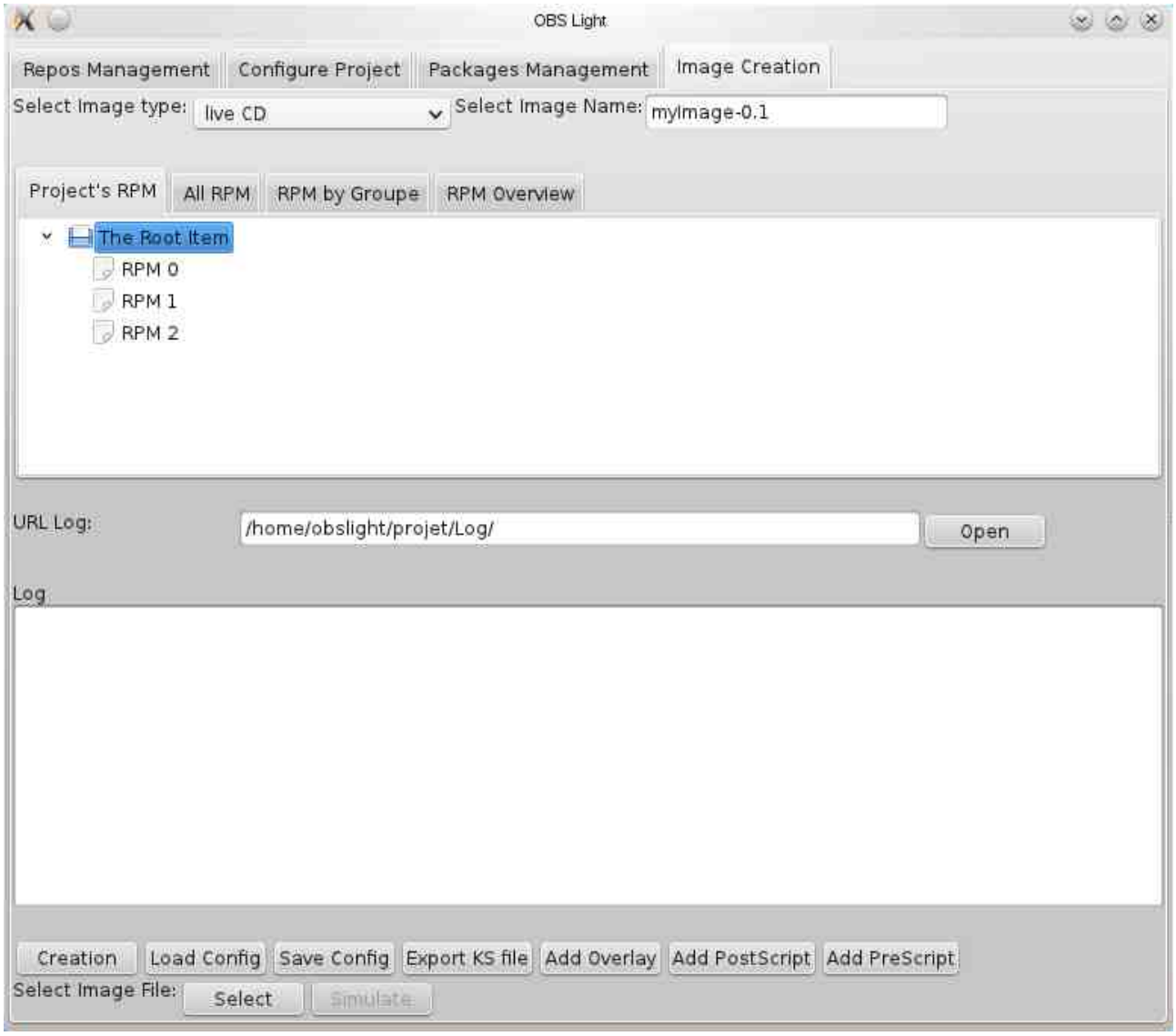
2) Below the “Source Files” window:

- **Add**: Add a source file to the current package.
- **Delete**: Delete the selected source file.
- **Update**: Update the selected source file via the source file link.
- **Build**: Build the current package.

What the **OBSLightManager** must do:

- Add a package to a project.
- Delete a package from a project.
- **OBSLightManager** is able to copy an original source file to a specified location.

4 Image Creation



What the user can do:

- By action on Combo Box:
  - **Select Image Type**: Specify a valid image type.

- By write in textfield:
  - **Select Image Name**: Specify an image name.

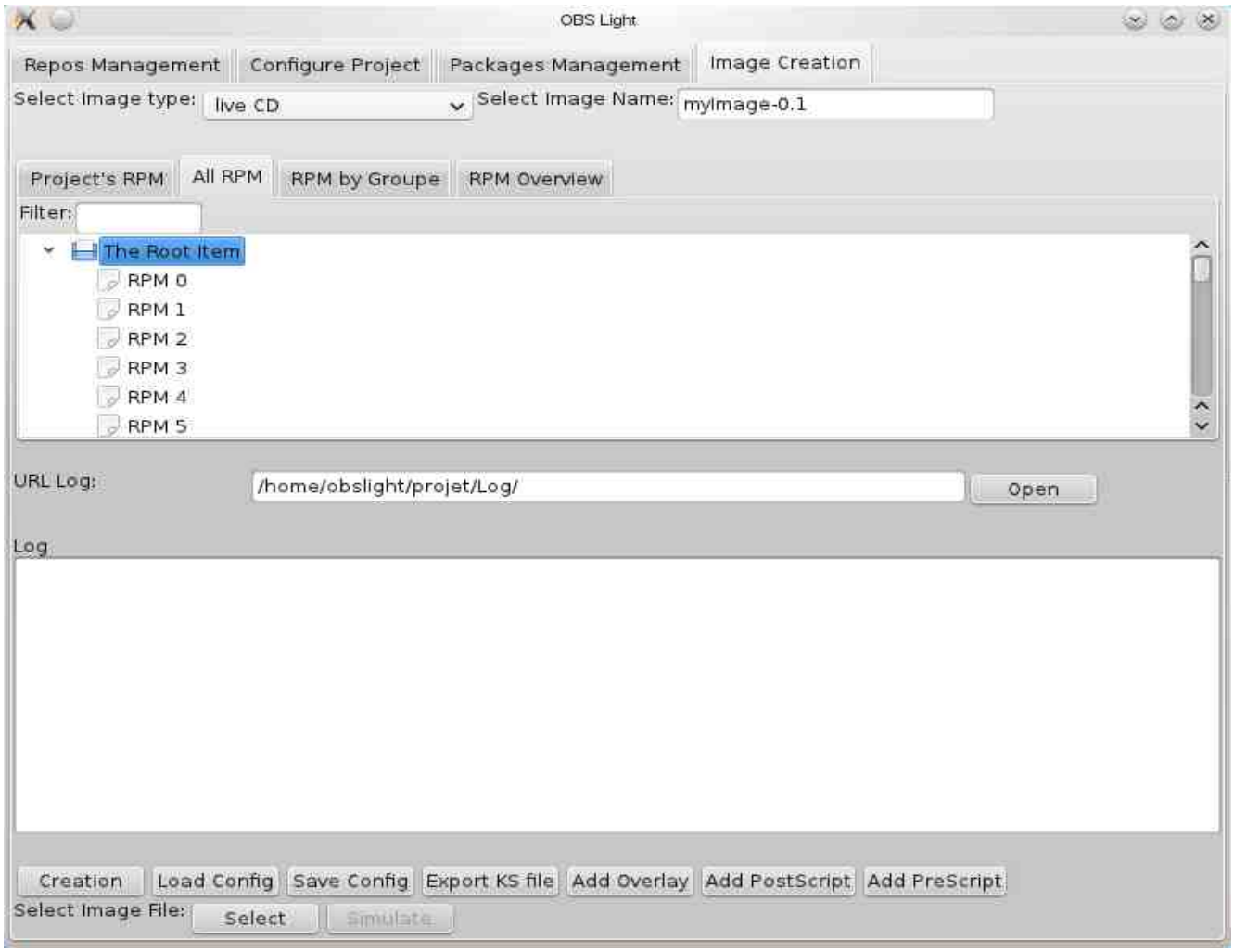
- By action on Button:
  - **Build:** Build a distribution from project.

- **Load Config**: Save a list of RPMs and RPM groups.
- **Save Config**: Save the list of the selected RPMs and RPM groups.
- **Export KS file**: Export the KS file of the current project.
- **Add Overlay**: Add a overlay file.
- **Add PostScript**: Add a postscript file.
- **Add PreScript**:Add a prescript file.
- **Select**: Select an image file.
- **Simulate**: Run a simulation of the selected file.

- By the tree:
  - User can select a RPM or a group of RPM.

What the user can see:

- The log of the build of the build solution.
- The URL of the log repertory.
- The RPM of the current project, by default they are selected.

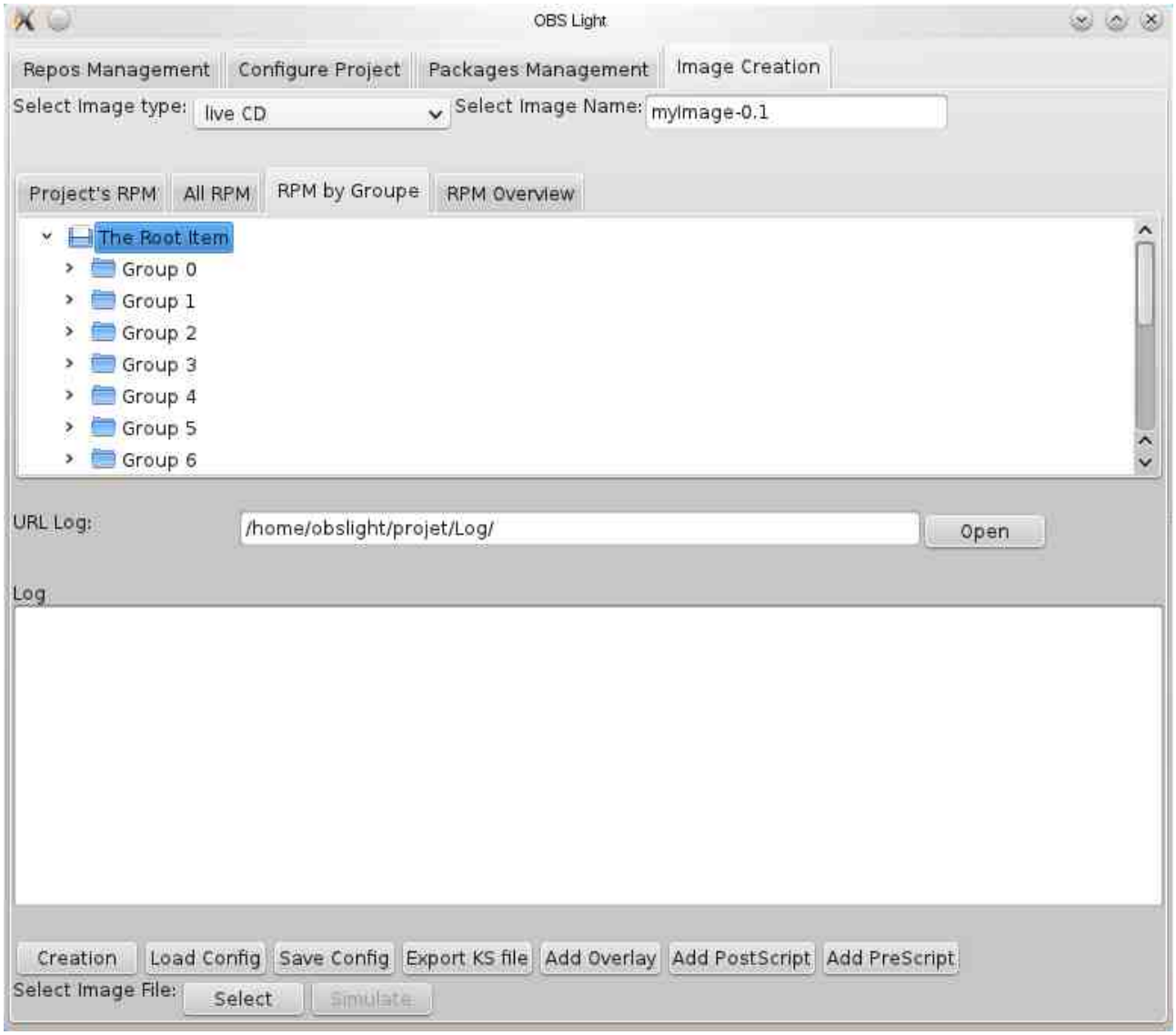


What the user can see:

- All RPM of the RPM list, by default they are unselected.

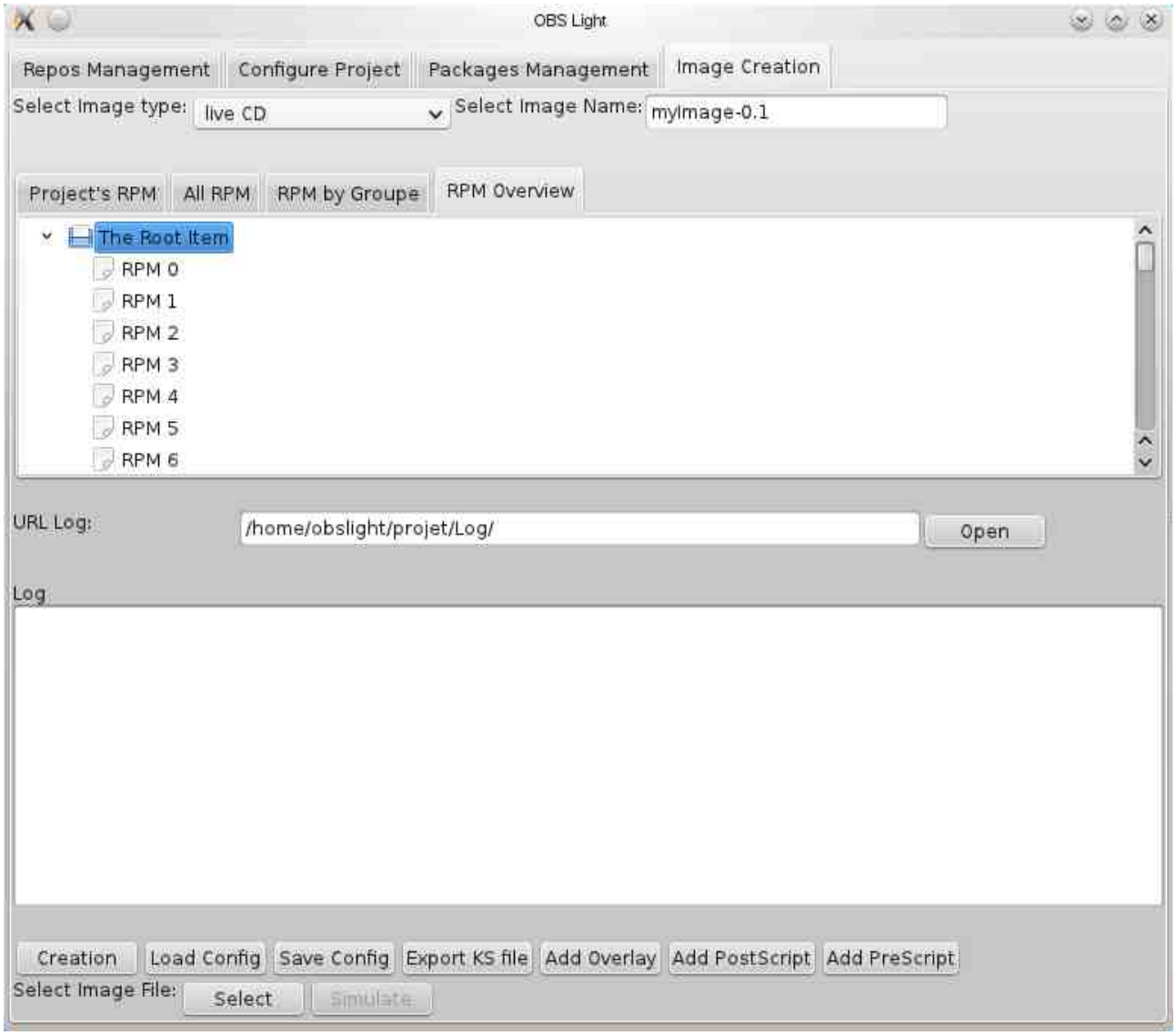
What the user can do:

- By write in textfield:
  - **Filter**: Filter the list of RPM by name.



What the user can see:

- All the RPM Groups of the RPM list, by default they are unselected.



What the user can see:

- All the selected RPM.

What the OBSLightManager must do:

- Find the RPM list from the current Repository.
- Find the XML file containing the RPM group.
- Save the current selection of RPMs and RPM groups.
- Load the current selection of RPMs and RPM groups.
- Export the current project configuration into a KS.

5 Architecture

- An ***OBSLightManager*** has:
  - A ***LocalRepositories***.
  - A ***WorkingDirectory***. Default : /home/OBSLight.
  - A ***isOBSConnected***: Default : False.
  - A ***OBSLightProjects.***

- A ***OBSLightProjects.***
  - A list of ***OBSLightProject.***
  - A current ***ProjectName.***

- A ***OBSLightProject*** is specified by:
  - A ***ProjectName***.
  - A ***ProjectDirectory***. The default is ***WorkingDirectory***/Project/***ProjectName***.
  - A ***chrootDirectory***. The default is ***ProjectDirectory***/chroot.
  - A ***ReposCacheDirectory***.
  - A list of ***Package***.
  - A current ***Package***.
  - A List of ***DependentPackage***.
  - A ***ProjectTarget***. The target's Name is ***RepositoryName***.
  - A list of ***ProjectArchitecture*** The architectures are [i586, armv7el].
  - A ***ProjectArchitecture***. By default i586 selected.
  - A list of ***ImageType:*** The possible statuses are: *"liveCD","liveUSB","loop","raw/KVM/QEMU","VMWare/vmdk","VirtualBox/vdi","Moorestown/mrstnand","jffs2",”nand”,”ubi”].*
  - An *ImageType*. By default *liveCD* selected.
  - An ImageName. By default ***ProjectName***.
  - A list of ***RPM*** selected.

- A ***Package*** has:
  - A ***PackageName***.

- A ***RPMFileObject***.
- A ***ListStatus***. The possible statuses are:*[NotBuilt,Built,BuildFailed]*.
- A ***PackageStatus***. Default: *is NotBuilt.*
- The list of ***SourcesFile***.
- A ***RPMName***.
- A ***RPMVersion***.
- A ***RPMPackageVersion***.

- A ***LocalRepositories*** has:
  - A list of ***LocalRepository.***

- A ***LocalRepository*** has:
  - A ***RepositoryName***.
  - A ***CreationDate***.
  - A ***BaseReposUrl***.
  - A ***ReposID***.
  - A ***ReposVersion***.
  - A ***RepositoryStatus***. [Update; NoUpdate; UpdateInProgress].
  - A list of all the available ***RPMName***.
  - The list of all the available ***GroupRPMName***.

- A ***SourcesFile*** has:
  - A ***SourcesFileName***.
  - A status of creation.The possible creation statuses are: *[FromRPM,FromSourceFile,NoFile]*. By default *NoFile.*
  - A status. The possible statuses are *`["deleted","ok","updated"].`* The default status is *ok*.
  - A link to the original source.
  - A MD5 signature.

- A ***SourceDataFile*** has:
  - A ***FileName***.
  - A MD5.

- A ***RPMFileObject*** has:
  - A ***RPMName***.
  - A ***RPMVersion***.
  - A ***RPMPackageVersion***.
  - A ***RPMFile***. Default: Empty file.

- The valid ***RPMFile*** names are:

  [***RPMName***]-[***RPMVersion***]-[***RPMPackageVersion***].rpm
  - ex: PyOpenGL-3.0.1-2.4.src.rpm.

- The valid repository ***ReposURL*** are:

  [***BaseReposUrl***]/[***ReposID***]/[***ReposVersion***]
  - ex: http://repo.MeeGo.com/MeeGo/builds/1.2.80/1.2.80.3.0.20110525.2

6 Solution

TV is the first target for OBS Light but it might be used as well for IVI, Phone, notebook, etc, as well.

The solution is delivered as a self-contained DVD image.

Links to all the GPL sources are provided on the DVD image as well.

An installation procedure is available on the DVD.

It has been tested for:

Initial versions :

- OpenSuse

Later:

- Ubuntu
- Fedora
- Debian
- Redhat
- Mandriva

7 Licenses overview

Project	License
OBS Light	GPL2
MIC2	GPL2; MIT.
OSC	GPL2

8 OBS Light vs Full OBS

	OBS Light	Full OBS
Cross compilation	**Yes**	**Yes**
Local and remote reference	**Yes**	**Yes**
Automatic creation of patch	**Yes**	**No**
Bootable Image creation	**Yes**	**Add on**
Integrated version control	No	**Yes**
Multi-user	No	**Yes**
Multi-target by project	No	**Yes**
Auto built	No	**Yes**
Signing service	No	**Yes**