



# Collabora OBS migration guide

## Build and Integration

**Author:** Hector Oron  
**Version:** v0.3  
**Status:** Final  
**Date:** Jun 16, 2014  
**Last Reviewer:** Jack Leighman  
**Classification:** Public

This proposal was produced exclusively using free and open source software.

Please consider the environment before printing this document.

Version	Date
V0.3	2014-06-16

## Document Change Log

Version	Date	Changes
0.1	2013-11-13	<ul style="list-style-type: none"><li>Initial version</li></ul>
0.2	2014-02-04	<ul style="list-style-type: none"><li>Reviewed cleaned up version</li></ul>
0.3	2014-06-16	<ul style="list-style-type: none"><li>Add subsections in DoD section, improvements over prj conf configuration.</li></ul>

## Table of Contents

Document Change Log.....	2
1 Overview.....	6
1.1 Purpose.....	6
1.2 Definitions and Abbreviations.....	6
1.3 References.....	6
1.3.1 Reference documents.....	6
1.4 Summary.....	6
2 Collabora's Build Infrastructure Overview.....	8
3 Host Operating System.....	9
3.1 Install host distribution.....	9
3.2 Install Collabora certificate.....	9
3.3 Add package repositories.....	9
4 OBS server Installation.....	11
4.1 Installation of OBS packages.....	11
4.2 Optional installation of osc tool.....	11
4.3 Enable OBS WebUI.....	12
4.4 Initial configuration using the WebUI.....	13
4.5 Selecting build architectures.....	15
5 OBS Server projects.....	17
5.1 Project creation.....	17
5.1.1 Adding a project with WebUI.....	17
5.2 Project Configuration.....	21
5.2.1 Meta project file.....	21
Editing using the WebUI.....	22
Editing using the commandline.....	23
5.2.2 Project configuration file.....	23
Editing using the WebUi.....	24
Editing using the commandline.....	26
5.3 Chaining projects.....	26
5.4 Migrating projects into new OBS instance.....	27
6 OBS Project bootstrap.....	28
6.1 Reference another OBS instance as base project.....	28
6.1.1 Example DOD project.....	31
6.1.2 Managing the scheduler.....	41
7 Managing repositories with reprepro.....	43
7.1 Using reprepro along with OBS.....	43
7.2 Architecture all packages.....	44
7.3 Removing/adding packages from shared repository.....	44
8 Configure signing keys.....	46
8.1 Assumptions & Requirements.....	46
8.2 GnuPG Keys Creation.....	46
Note about security.....	46
OBS GnuPG Key.....	46
Public repository GnuPG key.....	49
Exporting the keys.....	49
8.3 Adding keys to the keyring package.....	49

- 8.4 Enabling signatures for OBS-created repositories.....50
  - 8.4.1 Setting up GnuPG.....50
  - 8.4.2 Setting up the OBS Signer service.....50
- 8.5 Creating and verifying signatures on Debian-style reprepro repositories.....51
- 8.6 Configure BZ hooks.....52
- 9 Configure merge-o-matic (mom).....53
  - 9.1 Install and configure merge-o-matic.....53
  - 9.2 Using merge-o-matic.....55
  - 9.3 Configuration files.....56
  - 9.4 Output files.....56
- 10 OBS worker installation.....58
  - 10.1 The worker package should be installed with apt.....58
- 11 OSC API client tool.....60
- Appendix A - Files.....61
  - Generic Project configuration (prjconf) for Debian repository type.....61

## Index of Tables

- Table 1: Terms and Meanings.....5
- Table 2: Reference Documents.....5

# 1 Overview

Open Build Service is a tool to manage software distributions. This document aims to provide the reader with instructions on how to install, setup, configure and start running new projects in Open Build Service.

How to migrate existing projects into a new OBS instance is also covered within this document. The flow of the document scopes a step by step process to get to the final goal of having a system fully working and running.

## 1.1 Purpose

This document guides the reader into setting up Collabora's Open Build Service infrastructure and migrating projects from old service setup to newer one.

## 1.2 Definitions and Abbreviations

Term	Meaning
OBS	Open Build Service

Table 1: Terms and Meanings

## 1.3 References

### 1.3.1 Reference documents

Reference documents are documents that do not apply to this specification but are referenced or were used as input to write this specification.

Ref No.	Document
[RD-1]	OBS Documentation ( <a href="http://en.opensuse.org/Portal:Build_Service">http://en.opensuse.org/Portal:Build_Service</a> )

Table 2: Reference Documents

## 1.4 Summary

- Section 1 presents a general description and provides the reader with useful information for better clarification of the contents.
- Section 2 presents a general overview of the full infrastructure system deployed at the Collabora site.
- Section 3 focuses on describing step by step what needs to be done and how to deploy Open Build Service (server side).
- Section 4 focuses on describing step by step what needs to be done and how to deploy

Version	Date
v0.3	2014-06-16

Open Build Service (worker side).

- Section 5 describes a command line tool (OSC) that communicates with OBS API for job management and monitoring.

## 2 Collabora's Build Infrastructure Overview

An overview of the current Collabora setup is provided in this section. While the actual resource requirements depend on the usage, Collabora's setup can be used as a reference for an initial deployment.

The various machines run as the following services for distribution building and where 'project' refers to the specific project name using the service:

- **build.collabora.co.uk**: Main OBS server, shared between projects
- **repositories.project.collabora.co.uk**: Provide repositories of the packages built by OBS
- **images.project.collabora.co.uk**: Builds and provides access to disk images
- **worker build machines**: Build worker slaves for OBS to build packages

Currently Collabora uses a small set of virtual machines to run these services which are resourced as follows:

Services	Distro	CPU	RAM	Storage
build.collabora.co.uk	Debian wheezy	2, amd64	8GB	250GB, raid1 rotating media
images.project.collabora.co.uk, repositories.project.collabora.co.uk	Debian wheezy	1, amd64	4GB	250GB, raid1 rotating media
Various x86 build machines	Debian wheezy amd64	4, amd64	16GB	50G SSD
Various armhf build machines	Debian wheezy or ubuntu precise	4, Calxeda Ecx-1000 (ARM cortex A9)	4GB	60GB SSD

Note the Intel CPU based machines are virtual machines that are on purpose very thinly resourced especially in the storage area and it is expected they will be upgraded over time to meet the requirements of usage.

## 3 Host Operating System

All components of an OBS system run on a machine with a host operating system which must be configured appropriately. This configuration is the same regardless of the use of the machine (OBS master, build slave, client etc.)

### 3.1 Install host distribution

All packages are available in the deb format for 64bit Intel compatible processors. Packages required for build slaves are also available in deb format for ARM hardfloat machines.

Distribution support by Collabora to use for OBS include:

- Ubuntu 12.04 LTS (Precise), 64 bit server edition<sup>1</sup>.
- Debian 7 (Wheezy), 64 bit.

it should be noted that the Debian/Ubuntu "amd64"<sup>2</sup> architecture name is an implementation detail which may refer to both Intel x86 and AMD 64-bit processors.

The distribution specific documentation for operative system installation should be referred to and is not discussed further here. The following sections assume an installation of either distribution has been completed.

### 3.2 Install Collabora certificate

To securely retrieve and authenticate packages from a Collabora repository using the https protocol a certificate authority (CA) certificate is required.

The Collabora https certificate must be placed in `/usr/local/share/ca-certificates` and the CA update performed on the host machine with the `update-ca-certificates` tool.

The certificate is available from the collabora server and may be retrieved with the command:

```
# wget http://collabora.com/Collabora-CA.pem -O /usr/local/share/ca-certificates/Collabora-CA.crt
```

### 3.3 Add package repositories.

The software packages are retrieved from a repository (archive) by the APT package manager. Access to the repository is secured with a user name and password.

The repository server name, user name and password details will be provided by Collabora.

The APT package manager is configured by the addition of a list file to specify the repository.

---

1 <http://releases.ubuntu.com/12.04/ubuntu-12.04.3-server-amd64.iso>

2 <http://wiki.debian.org/DebianAMD64Faq#line-17>



The file `/etc/apt/sources.list.d/collabora.list` should contain a line similar to:

```
deb https://repositories.collabora.co.uk/obs/ubuntu/ precise tools
```

Once configured the APT cache must be updated

```
# apt-get update
```

This will complete with a warning that the added repository signature could not be verified which is rectified by installing the Collabora repository keyring using the `collabora-obs-archive-keyring` package. Grab the latest `collabora-obs-archive-keyring_*_all.deb` from here:

<https://repositories.collabora.co.uk/obs/{debian,ubuntu}/pool/tools/c/collabora-obs-archive-keyring/> and install with.

```
# dpkg -i collabora-obs-archive-keyring_*_all.deb
```

It is advisable at this point to perform an upgrade to ensure the base OS installation is updated to the current versions and contains all relevant security fixes.

```
# apt-get upgrade
```

## 4 OBS server Installation

These installation instructions assume that the host Operation System has been configured as outlined in Section 3.

### 4.1 Installation of OBS packages

Before OBS is installed the ruby language installation must be correctly configured. OBS requires version 1.8 of ruby be installed.

```
# apt-get install ruby1.8
```

The install solution APT selects may add other packages which should be accepted when prompted.

There may be several versions of ruby installed on your system, if this is the case it is necessary to ensure the default interpreter is the 1.8 version. This is achieved by using the alternatives system.

```
# update-alternatives --config ruby
```

Once version 1.8 is selected this may be checked with

```
# update-alternatives --display ruby
ruby - manual mode
  link currently points to /usr/bin/ruby1.8
```

If ruby1.8 is not set by default the alternative should be updated and version 1.8 selected.

The OBS packages may now be installed with.

```
# apt-get install obs-api obs-webui obs-server
```

The APT tool will prompt to install many additional packages which is expected and should be permitted.

If the system did not previously have MySQL installed it will be installed as part of this process and will prompt for a administrative password (for the MySQL "root" user) which is installation dependent and must be kept secure.

If this is the first time the OBS packages have been installed this will create a MySQL database which will prompt for passwords during installation.

### 4.2 Optional installation of osc tool

The OBS API service is accessed with the osc tool which must be installed to perform administrative tasks from the command line.

```
# apt-get install osc
```

The APT tool will prompt to install additional packages which is expected and should be permitted.

### 4.3 Enable OBS WebUI

The OBS WebUI is accessed via a correctly configured Apache HTTP/S server. The apache2 virtualhost is configured using an already installed site configuration file which must be enabled.

```
# a2enmod ssl
# a2dissite 000-default
# a2ensite obs.conf
```

It is necessary to configure an appropriate certificate for the HTTPS site. A self-signed certificate can be created and installed with:

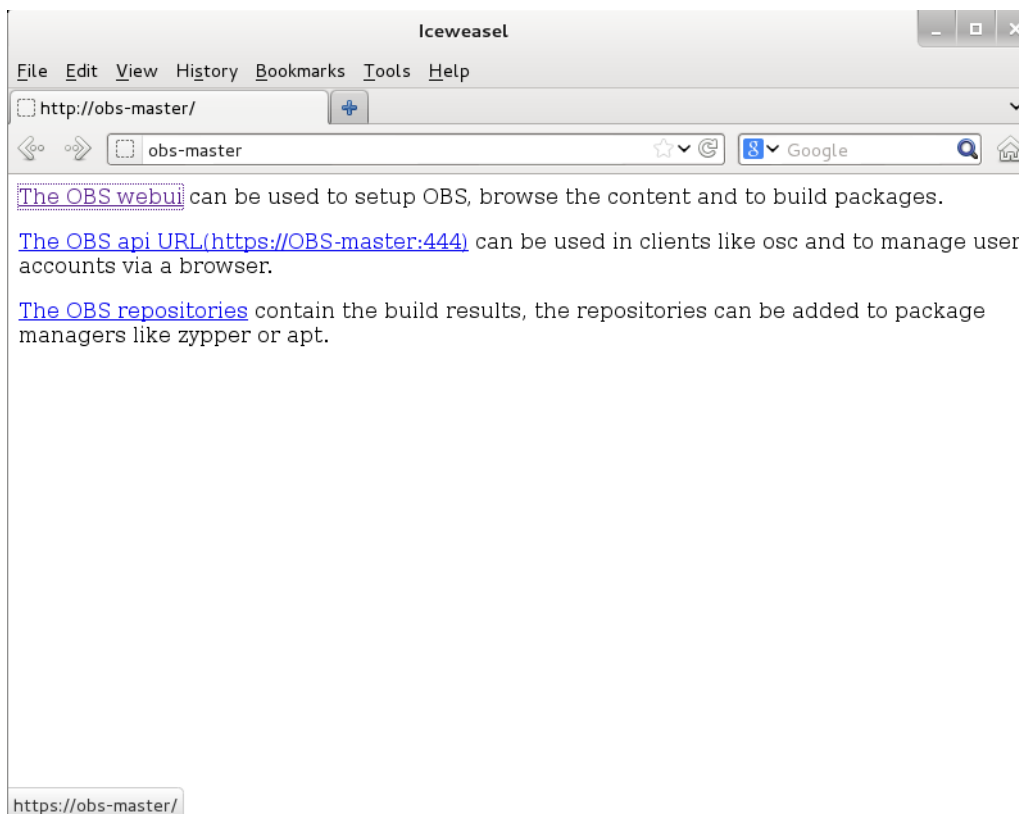
```
# make-ssl-cert generate-default-snakeoil --force-overwrite
```

Alternatively /usr/share/doc/apache2.2-common/README.Debian.gz details configuring a real certificate.

Once Apache is configured it must be restarted for the changes to take effect.

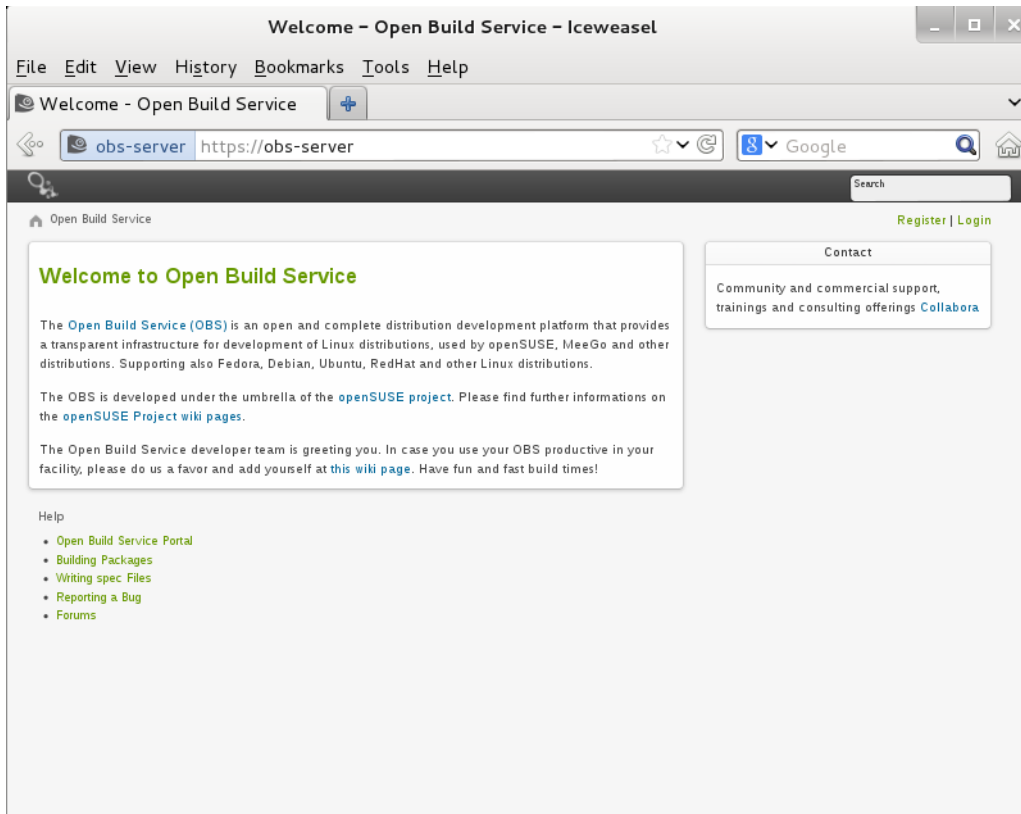
```
# service apache2 restart
```

It should be checked that web server is available using a browser.

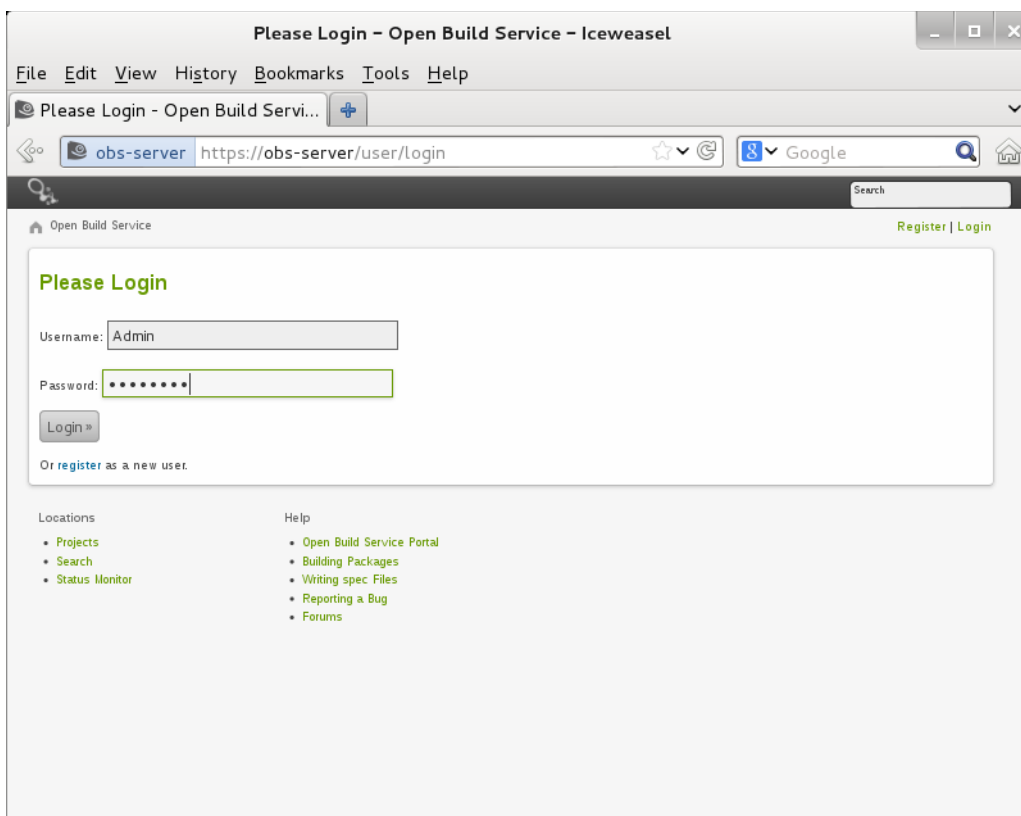


### 4.4 Initial configuration using the WebUI

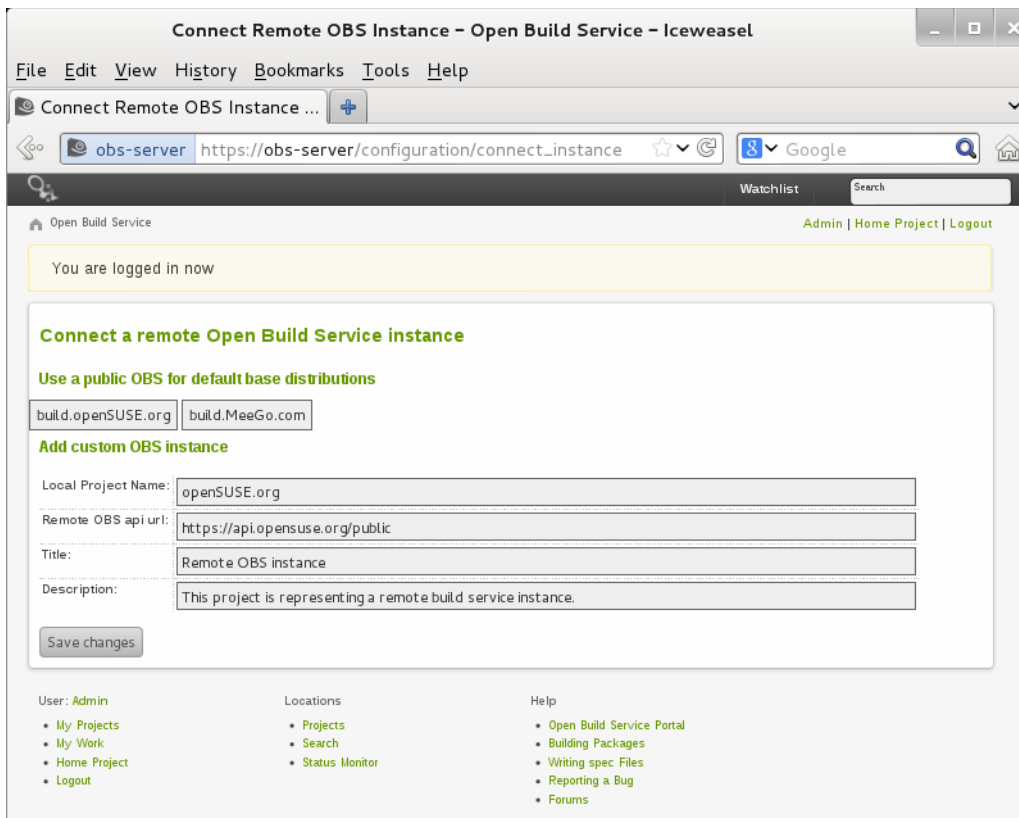
The OBS WebUI is available via https. If self-signed certificates were used the browser may require configuring to accept the new CA.



The examples in this document use the obs-master hostname to access the WebUI and API services, this must be adjusted to match the actual installation.



The default login is the user Admin with the password opensuse.



After the initial login there are no projects configured and the default interface prompts for a linkage to an upstream OBS service.

Although not recommended, the SUSE public OBS project may be added which will give simple overlay projects to be created using the SUSE OBS public repositories. Projects cannot be based on the Debian squeeze found there as the packages contain modern build infrastructure which is incompatible with these older Debian releases.

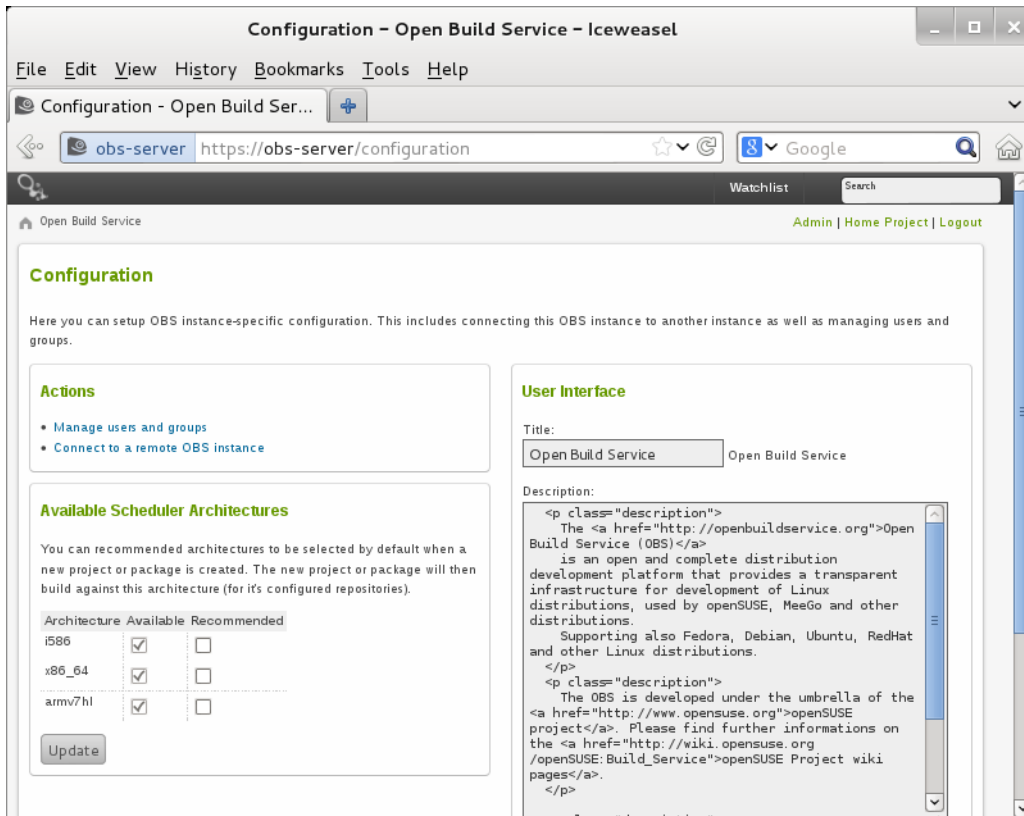
## 4.5 Selecting build architectures

The architectures that can be built by the OBS instance are configured in the `/etc/default/obs-server` file with the **OBS\_SCHEDULER\_ARCHITECTURES** parameter.

To select i586, x86\_64 and armv7hl architectures this would be set to

```
OBS_SCHEDULER_ARCHITECTURES="i586 x86_64 armv7hl"
```

If this parameter is altered it is recommended that the OBS server instance be rebooted to ensure correct operation.



After the server is restarted the server configuration page must be visited and the "Available Scheduler Architectures" updated.

## 5 OBS Server projects

Within an OBS instance a project represents the resources required to build a set of packages. The resources include source archives patch files etc. and produce repositories which are simply collections of built packages with the correct directory layout and meta files for tools like APT.

The project can be considered to be synonymous with a distribution for the purposes of Debian tools.

### 5.1 Project creation

The Open Build Service documentation <sup>3</sup> details the process more fully but the basic steps are the creation of a project which then requires configuring with metadata and a project configuration to setup the build environment for packages within a project.

Projects can either be *public* or *private*. A public project is accessible by anyone with access to the OBS API or web interface, whereas a private project is only accessible by its creator and other users in the project's ACL. If a project has been made public, it cannot be switched to private.

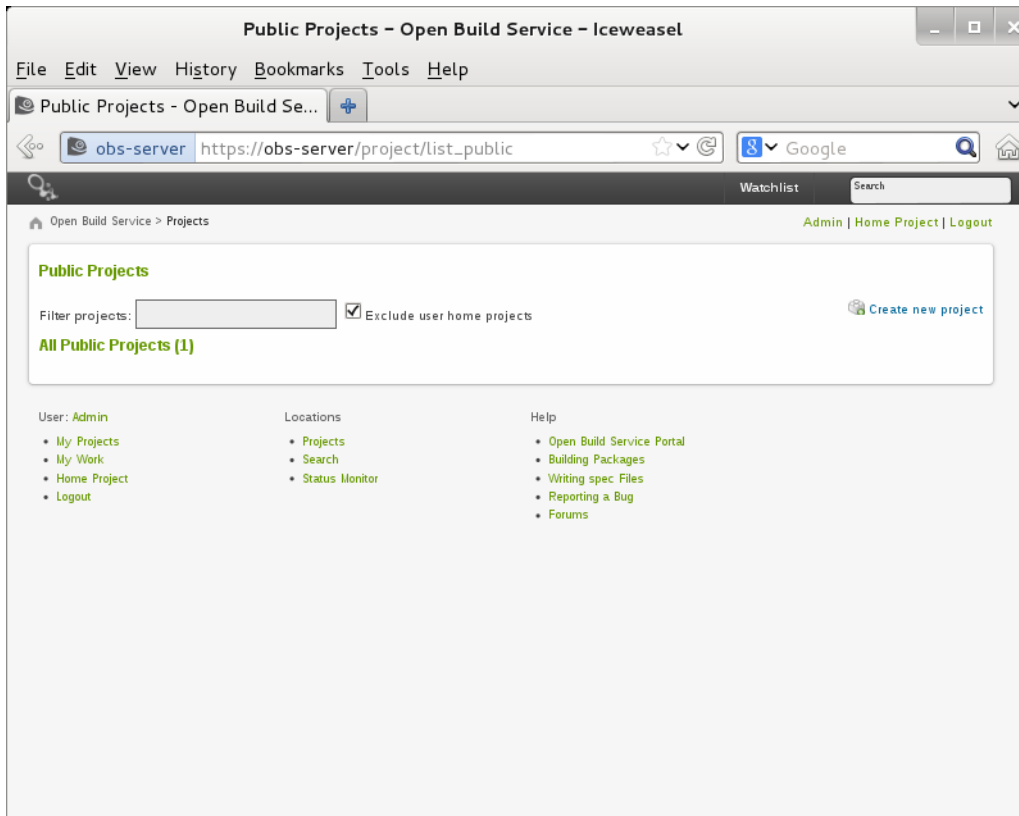
To create a private project, ensure that "Hide the entire project" is checked. The default is to create a public project, with that checkbox not checked. The project metadata for a private project (see section "Meta project file" below) will contain an `<access><disable/></access>` block.

#### 5.1.1 Adding a project with WebUI

Start by selecting the "Projects" link from the Locations list in the lower centre of the web page which should take you to the public project list.

---

<sup>3</sup> [http://en.opensuse.org/openSUSE:Build\\_Service\\_Tutorial](http://en.opensuse.org/openSUSE:Build_Service_Tutorial)



Select the "Create New Project" link. This will open a page where the project name, title and description can be entered. Although this name may be anything there is a convention to ease administration. Note if you intend to use merge-o-matic the project name must follow this convention or at least end with :<release codename>:<component>.

Assuming you are developing a debian-derived distribution the convention when naming a project is to use a form

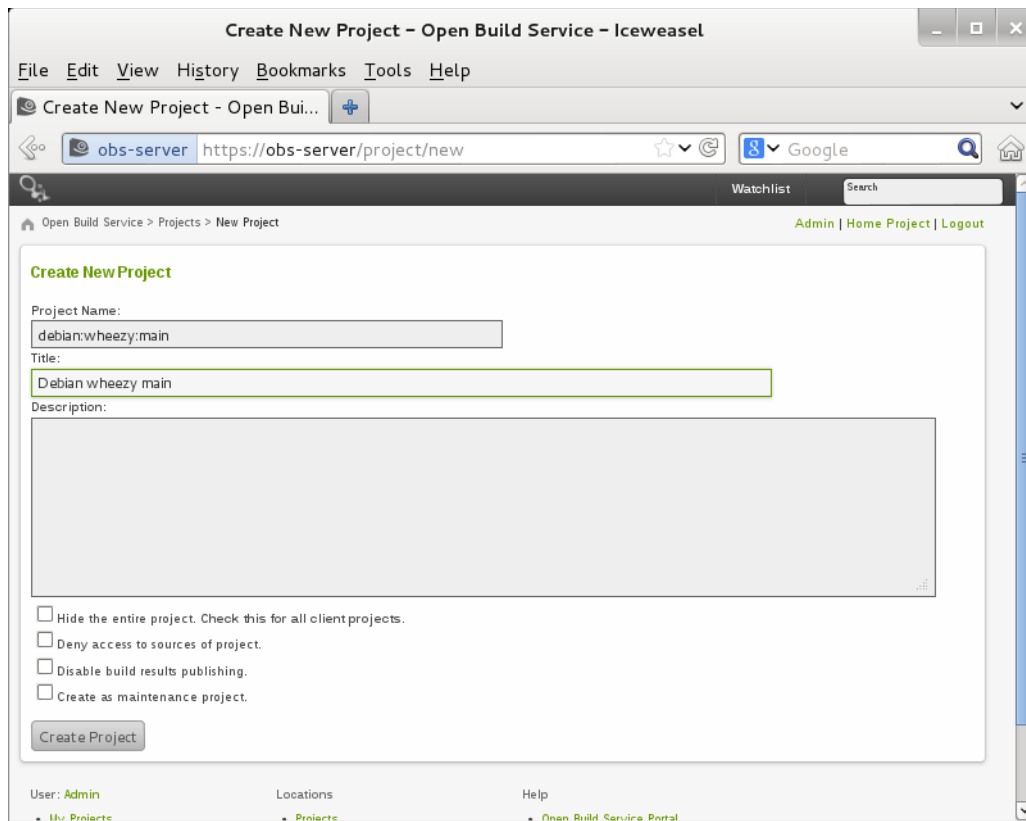
<distribution name>:<release codename>:<component>
--

The three parts of this are:

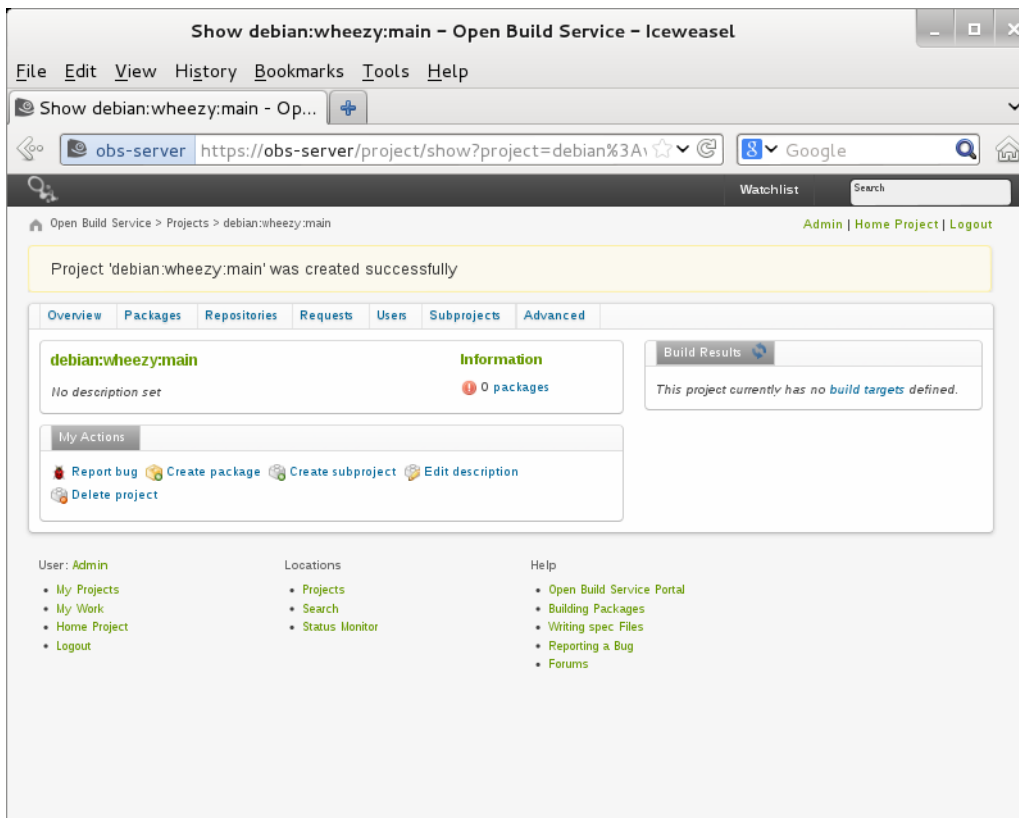
- distribution (analogous to debian, ubuntu, maemo, tanglu etc.)
- release name (analogous to wheezy in Debian, precise in Ubuntu, etc.)
- components (analogous to main, contrib, non-free in Debian; main, restricted, universe, multiverse in Ubuntu)

Some example project names are: debian:jessie:main, ubuntu:precise:main





Once the project is created it is available to configure and perform other administrative tasks on.



## 5.2 Project Configuration

Once a project is created there are two configuration files that can be edited:

- The project meta information including user and group ACL, repository information, and other general project settings.
- The project configuration which controls how the distribution is built.

This section shows how these files are edited and their formats and the following section OBS Project bootstrap outlines what their contents will be for differing use cases.

### 5.2.1 Meta project file

The meta project file describes the OBS project properties regarding:

- Project title
- Project description
- Project users (ACL)
- Enable privacy, so other users not in ACL are unable to have a project view
- Enable architectures
- Packages repositories to use for build, to publish.

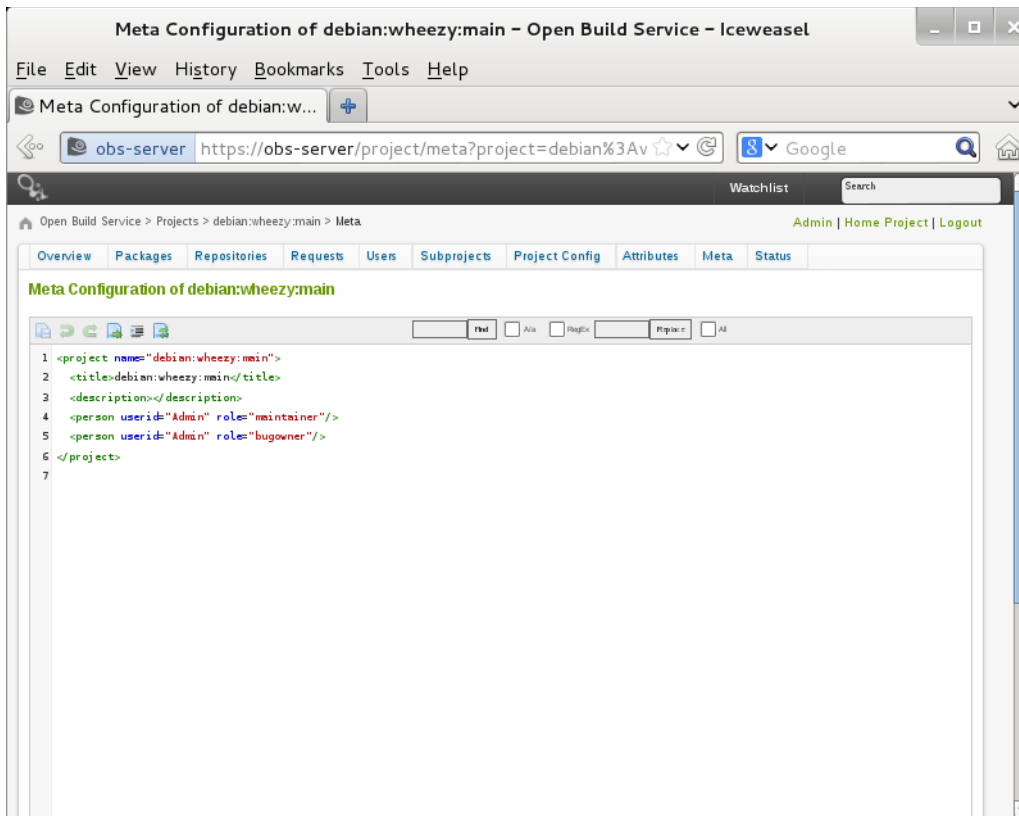
When editing project meta information ensure there is an access block if you wish your project

to be **private**, only users listed in the file will have project access. If removed, this cannot be undone.

```
<access>
  <disable/>
</access>
```

## 1.Editing using the WebUI

Using the WebUI the metadata can be edited using the Advanced->Meta entries.



## 2.Editing using the commandline

It is not recommended to edit the project metadata using the commandline but it can be achieved using the osc tool.

```
# osc -A https://obs-master:444 meta prj -e <project name>
```

### 5.2.2 Project configuration file

Project configuration file contains software distribution information for creating build environment.

The OBS packages provided by Collabora have been extended to support additional features.

The first additional feature is the capability to Inject build time suffix to Debian style repositories.

To ensure all binary builds have different versions (even over different projects) a suffix is appended to the debian build versions. Currently the convention is to use "b<first letter of the release name><rebuild count>".

Note that this currently even happens on the first build, hence updated packages will have suffix.

When updating packages to newer debian versions it should ensure that older release either have earlier debian versions, to ensure global uniqueness and an upgrade path even when coming from old releases.

When a newer release is forked, please ensure build suffix gets sorted after build suffix from previous release. (eg. Initial release, suffix "ba<B\_CNT>"; next release "bb<B\_CNT>"). The tag <B\_CNT> is a counter variable incremented during package rebuilds.

In Debian-style versioning, letters compare lower than punctuation such as "+" and "-", and "~" compares lower than the version without it. For instance, these versions are sorted in increasing order: 0.1-2co3~alpha1 < 0.1-2co3 < 0.1-2co3ba1 < 0.1-2co3+ba1 < 0.1-2ubuntu1. See "section 5.6.12 of the Debian Policy Manual" for full details, or use commands like

```
dpkg --compare-versions 0.1-2~ lt 0.1-2+ && echo yes
```

to check that versions behave as you expect.

A generic project configuration file to be used for Debian derivatives can be found in the appendix Generic Project configuration (prjconf) for Debian repository type. The file is also provided in the /usr/share/doc/obs-server/examples directory.

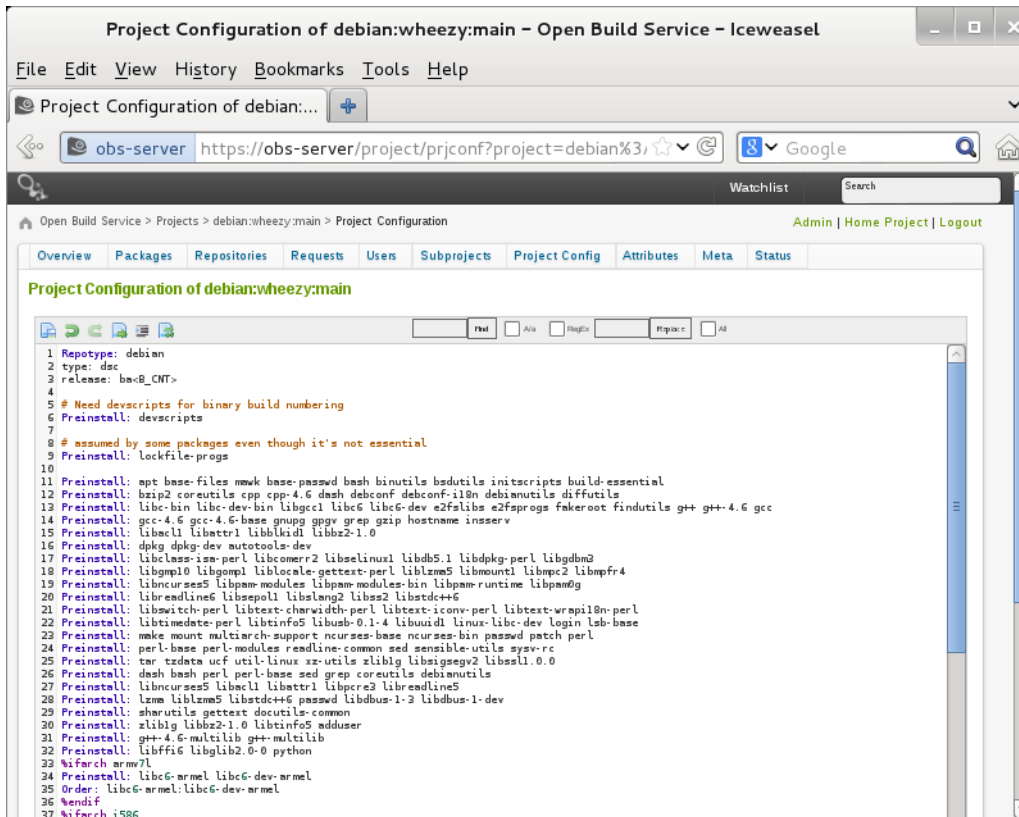
The most common tags found within the configuration file are:

- **Reptype:** <type> - Where type can be one of two choices, \_debian\_ or \_rpmmd\_
- **Preinstall:** <packages> - Packages which will be preinstalled in build chroot
- **Order:** <first package>:<second package> - install second package after first
- **Runscripts:** <packages> - Which packages should have their scripts executed or not. Unlike Debian or SUSE chroot strategies, default OBS instances don't install packages in a standard way, leaving the choice of whether to run the package scripts to the project administrator.
- **Support:** <packages> - Support packages that will be considered as a requirement to buildsystem structure, but not directly related to the build process
- **Prefer:** <packages> - Preferred packages (in case of multiple choices like gawk or mawk)
- **Ignore:** <packages> - Packages that will be simply ignored for the chrooted environment
- **Conflict:** <first package>:<second package> - Packages that can't be installed simultaneously
- **Substitute:** <packages> - Possible substitutions for first entry in the list  
RPM tags are commonly accepted, most common being the architecture ifdefs choices to specific packages. For example:

```
%ifarch %ix86 <Do Something> %endif
%define ubuntu_version 1204
```

## 1.Editing using the WebUI

The project configuration can be edited using the webUI.



## 2.Editing using the commandline

It is not recommended to edit the project configuration using the commandline but it can be achieved using the osc tool.

```
# osc -A https://obs-master:444 meta prjconf -e <project name>
```

### 5.3 Chaining projects

In some cases, some projects build on top of other projects. OBS projects can be chained, so project A packages can be used as build dependencies for building project B.

Projects are linked via meta configuration file, under repository directive. As an example, if the project singularity:betelgeuse:sdk is a SDK based on core packages found in a separate project singularity:betelgeuse:core, the following snippet must appear in the meta configuration file for singularity:betelgeuse:sdk:

```
<repository rebuild="local" linkedbuild="all" block="never"
name="betelgeuse">
  <path project="singularity:betelgeuse:core" repository="betelgeuse"/>
  <arch>i586</arch>
</repository>
```

The path directive links core project to betelgeuse repository. This directive is used to link current project to external project, being in OBS, in a remote instance or setup as download on

demand.

The repository directive allows setting up different build scheduling strategies and setting up tags, those are described under OBS documentation<sup>4</sup>, however for Debian builds Collabora recommends to set it up as the above example, to avoid useless rebuilds of packages and to not block packages on others building.

## ***5.4 Migrating projects into new OBS instance***

For migrating projects from old OBS to new Debian based OBS, you could perform the following steps:

- Add repository configuration
- Create projects in webUI
- Copy the project and meta configuration from previous to the current project (modify as needed)
- Stop schedulers
- Copy binaries from previous project to current one
- Copy source packages from previous project to current one
- Start schedulers

---

<sup>4</sup> [https://en.opensuse.org/openSUSE:Build\\_Service\\_Concept\\_build\\_scheduling\\_strategies](https://en.opensuse.org/openSUSE:Build_Service_Concept_build_scheduling_strategies)

## 6 OBS Project bootstrap

After a project has been created it must be bootstrapped with a source of packages, possibly binary packages and repository information.

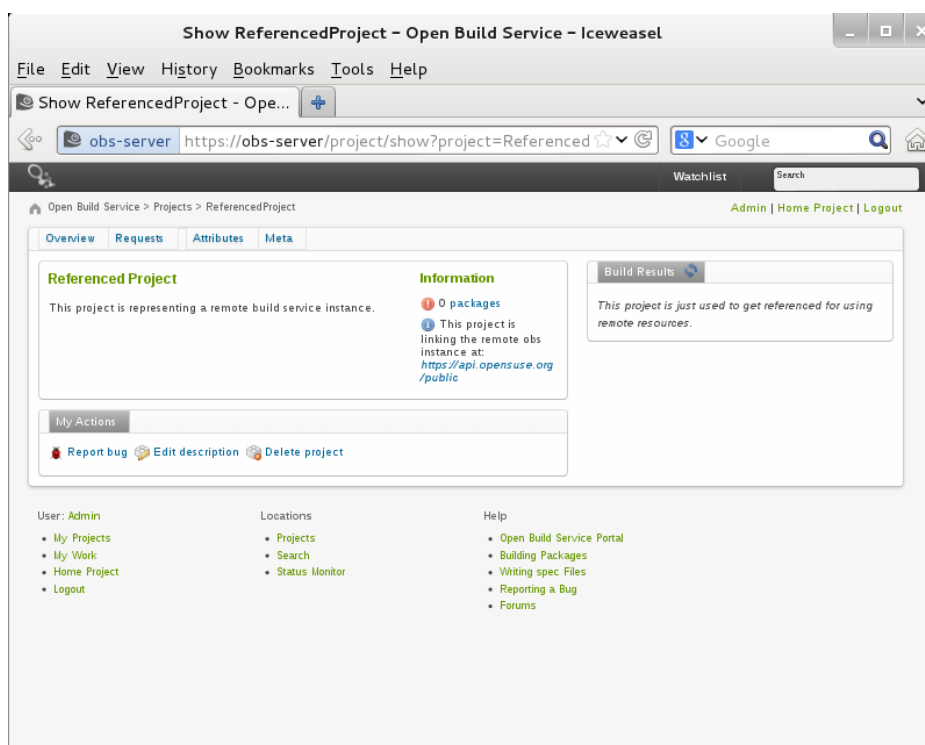
There are three principle ways to initialize the distribution:

- Reference another OBS instance as a base project
- Download on demand strategy
- Binary import

### 6.1 Reference another OBS instance as base project

This type of project is the most simple, it uses a remote OBS instance as a source of configuration and packages .

The local project configuration is simply the editing of the meta data to add a reference to the remote instances API service.



For example to use the SUSE public service the stanza would be

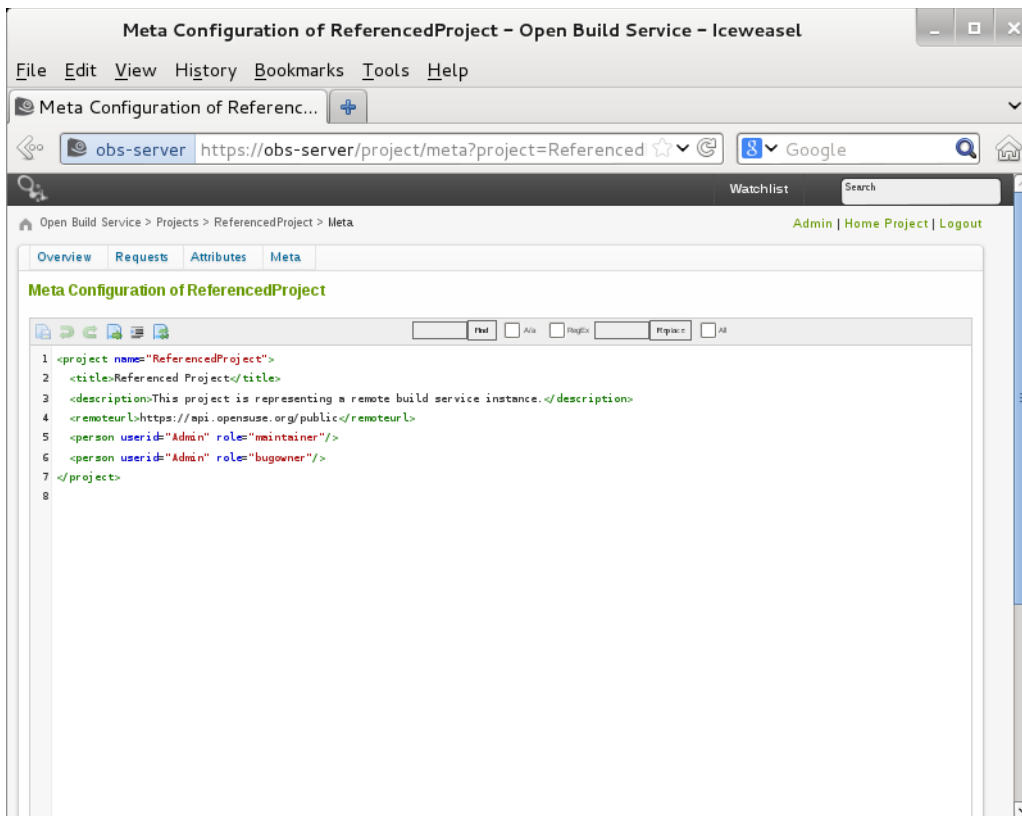
```
<remoteurl>https://api.opensuse.org/public</remoteurl>
```

Because of the simplicity of this type of project the WebUI provides a simple single click interface to generate such projects against the SUSE service requiring no manual configuration. As noted previously the Debian (version 6) repository found on this service cannot be used with the modern tools in the Collabora packaged OBS.

A complete example project meta information would look something like:

```
<project name="ReferencedProject">
  <title>Referenced Project</title>
  <description>This is a referenced project to an external OBS
server</description>
  <person userid="Admin" role="maintainer"/>
  <person userid="Admin" role="bugowner"/>
  <remoteurl>https://api.opensuse.org/public</remoteurl>
</project>
```

As shown in the WebUI



Download On Demand (DOD)

Download on demand repositories can be set in one of two modes:

- Direct on the project
- As a referenced internal project

The process is exactly the same for either type:

- Define your architectures
- Define the name of your repository
- Access the server OBS directory
- Create the directories needed <Project>/<Repository>/<Arch>/:full
- Copy the reference Packages file from the download server
- Restart the scheduler.

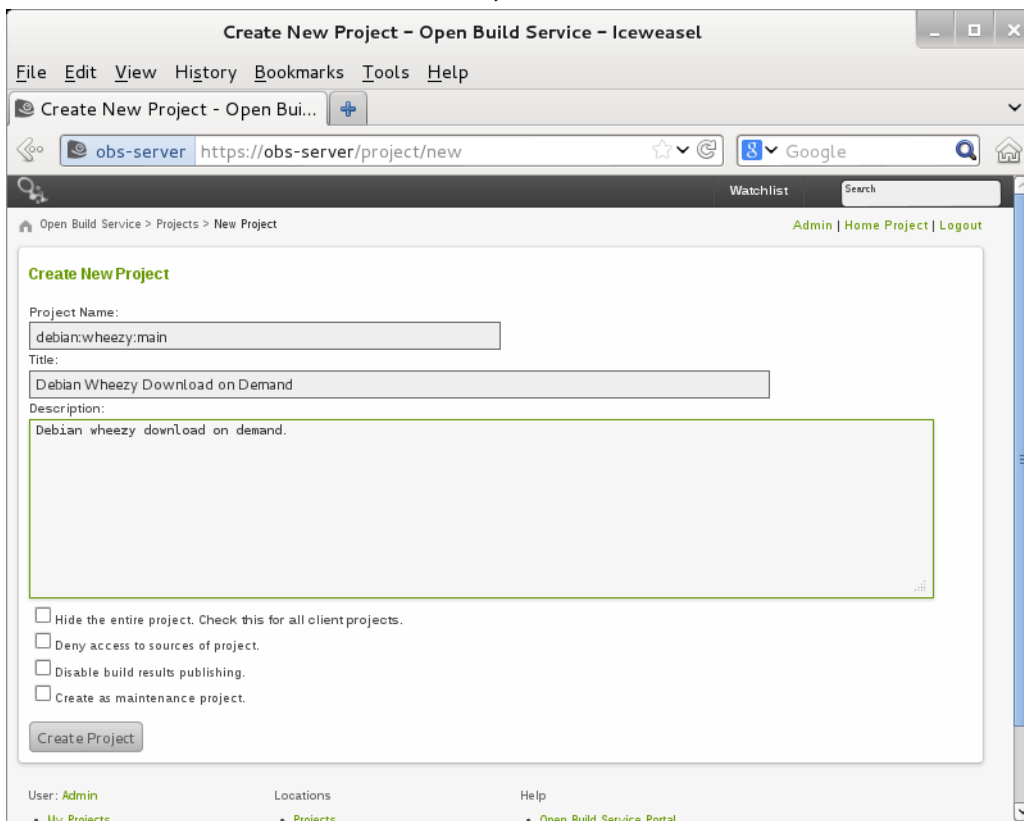


## 6.1.1 Example DOD project

This example shows how a DOD project can be created which uses Debian wheezy main as the repository source.

### 1. Create a new project

A project must be created as described in Project creation with the name **debian:wheezy:main**



**Create New Project**

Project Name:

Title:

Description:

Hide the entire project. Check this for all client projects.

Deny access to sources of project.

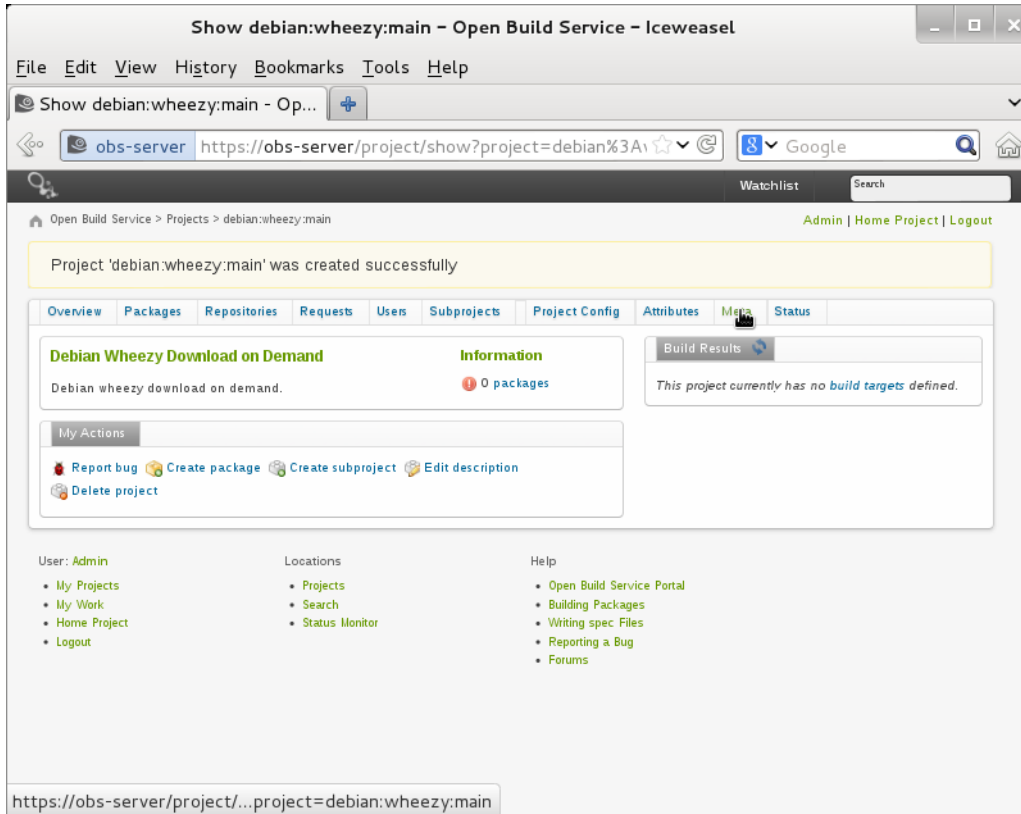
Disable build results publishing.

Create as maintenance project.

User: Admin      Locations      Help

• My Projects      • Projects      • Open Build Service Portal

## 2. Set Meta information



Once created the project meta information must be changed as described in Meta project file by selecting the Advanced option on the project and then the Meta option once it is revealed.

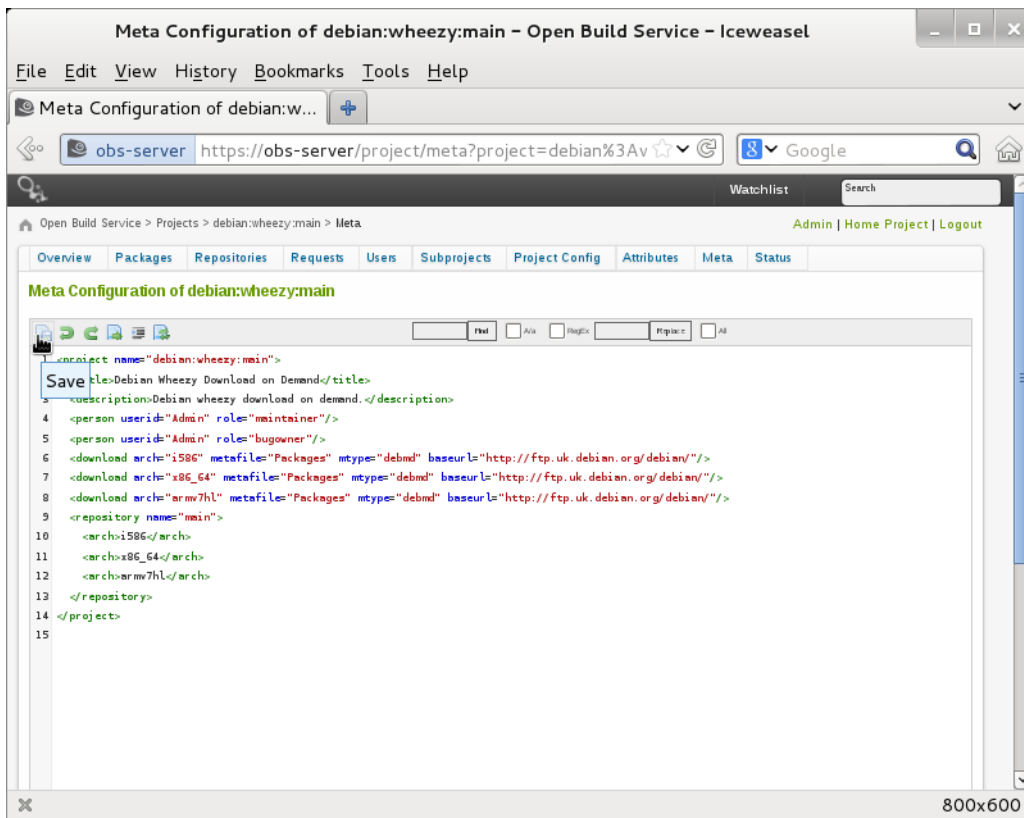
The meta information must be updated with base download entries for each architecture. For the Debian wheezy main example on i586, x86\_64 and armv7hl the following would need to be added:

```
<download baseurl="http://ftp.uk.debian.org/debian/" metafile="Packages"
mtype="debmd" arch="i586" />
<download baseurl="http://ftp.uk.debian.org/debian/" metafile="Packages"
mtype="debmd" arch="x86_64" />
<download baseurl="http://ftp.uk.debian.org/debian/" metafile="Packages"
mtype="debmd" arch="armv7hl" />
```

Additionally a basic repository stanza also needs adding

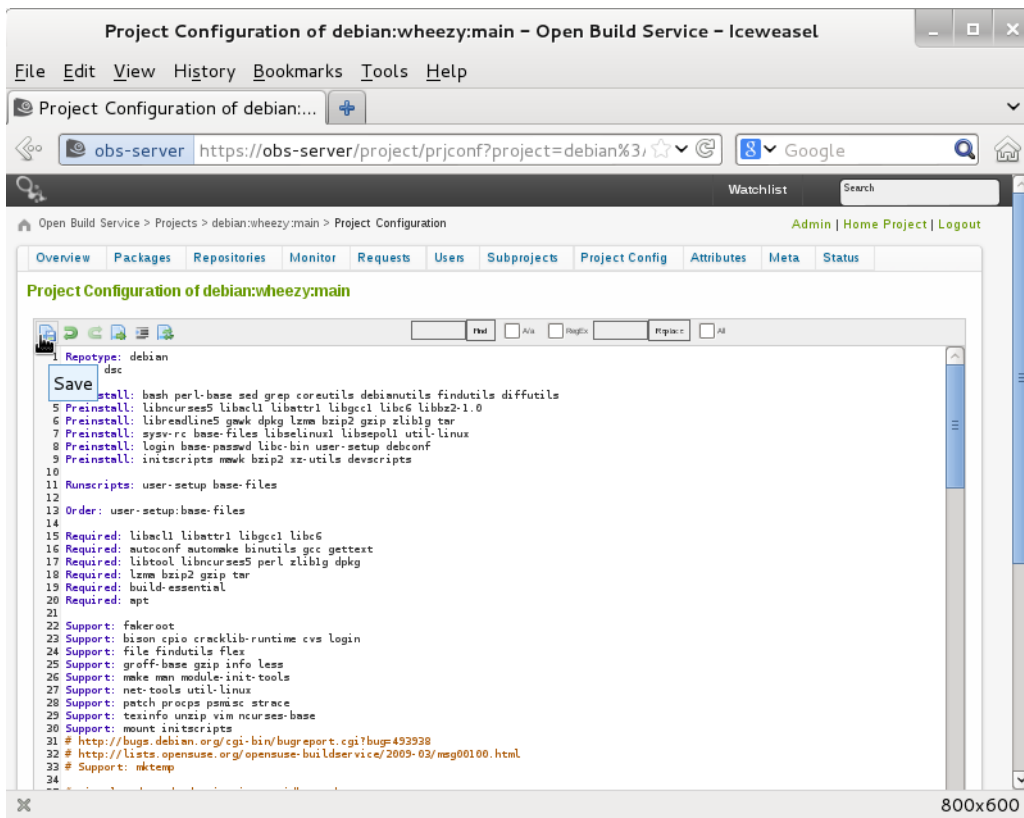
```
<repository name="main">
  <arch>i586</arch>
  <arch>x86_64</arch>
  <arch>armv7hl</arch>
</repository>
```

The meta configuration should be saved as shown in the WebUI:



With this configuration OBS knows about the main repository for the given 3 architectures, it will use `ftp.uk.debian.org/debian` as the bases for all downloads and it will use the provided metafile (Packages) to determine the list of available packages, their dependency information and their location relative to the baseurl.

The Packages metafile is not yet active and must be placed on the filesystem by hand once the project configuration has been completed.



### 3. Add a Project Config

As described in the Project configuration file section the project must be configured for building of deb packages. Example configuration files are provided in the obs-server package in `/usr/share/doc/obs-server/examples/debian-wheezy.prjconf.gz`

The project configuration describes two main parameters to OBS:

- What type of repository it is (Debian/dsc)
- What packages need to get installed for every build (the list of packages necessary for a minimal build environment)

### 4. Set up required directories and files

The build areas, package files and first scan must now be set up. This requires being logged into the OBS server as the root user.

First the build directories must be created using the structure

```
/srv/obs/build/<project name>/<repository>/<architecture>/:full/
```

specifically for this example:

```
# mkdir -p /srv/obs/build/debian:wheezy:main/main/armv7hl/:full/
# mkdir -p /srv/obs/build/debian:wheezy:main/main/x86_64/:full/
# mkdir -p /srv/obs/build/debian:wheezy:main/main/i586/:full/
```

A package file for each architecture must be placed in these directories these can be fetched from the Debian mirror directly and must be uncompressed. For this example the following can

be used:

```
# wget http://ftp.uk.debian.org/debian/dists/wheezy/main/binary-  
i386/Packages.gz -O - | gunzip >  
/srv/obs/build/debian\wheezy\main/main/i586/\:full/Packages  
# wget http://ftp.uk.debian.org/debian/dists/wheezy/main/binary-  
amd64/Packages.gz -O - | gunzip >  
/srv/obs/build/debian\wheezy\main/main/x86_64/\:full/Packages  
# wget http://ftp.uk.debian.org/debian/dists/wheezy/main/binary-  
armhf/Packages.gz -O - | gunzip >  
/srv/obs/build/debian\wheezy\main/main/armv7hl/\:full/Packages
```

It is important that the files in these directories be owned by the obsrun user and group, this is done with the chown command.

```
chown -R obsrun:obsrun /srv/obs/build/debian\wheezy\main/
```

The next step is to restart the scheduler.

```
/etc/init.d/obsscheduler restart
```

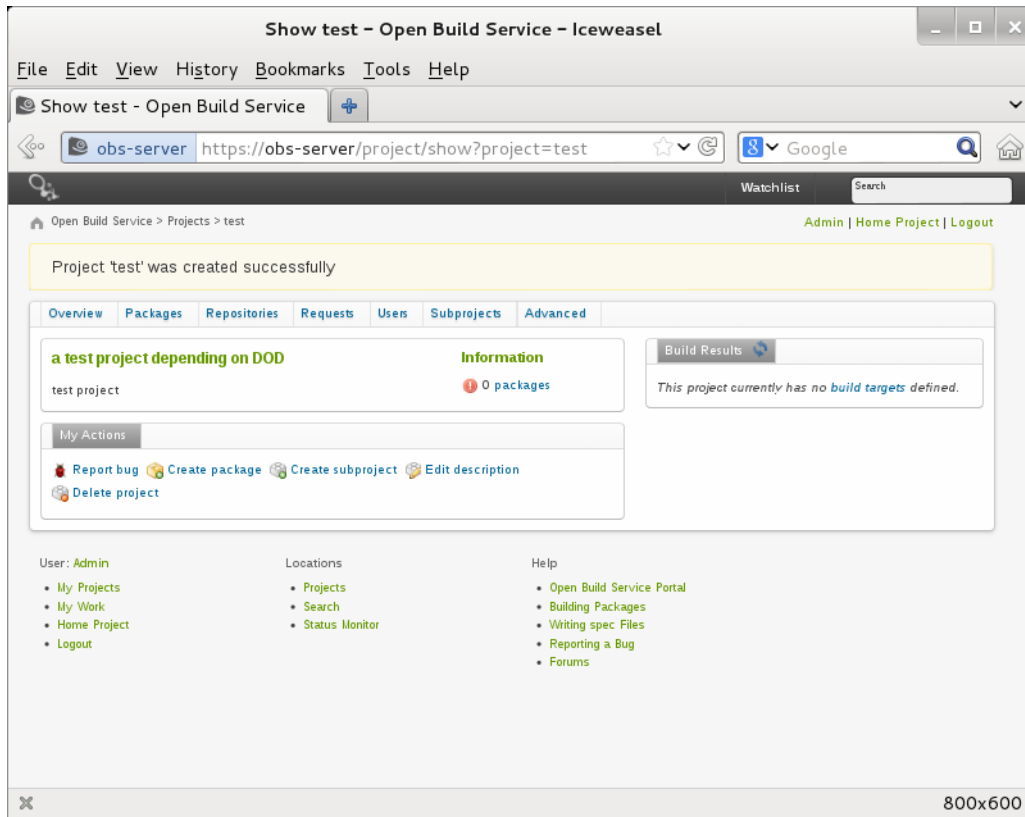
## 5. Check setup

To complete the setup, the obs-dod-check script is used to check consistency and trigger the first scan. The script will compare the Packages file with the dependency database generated by OBS scheduler to see if they match, in which case the DOD is setup correctly.

The expected output from the script is shown here. If there is an error the script will produce obvious error messages.

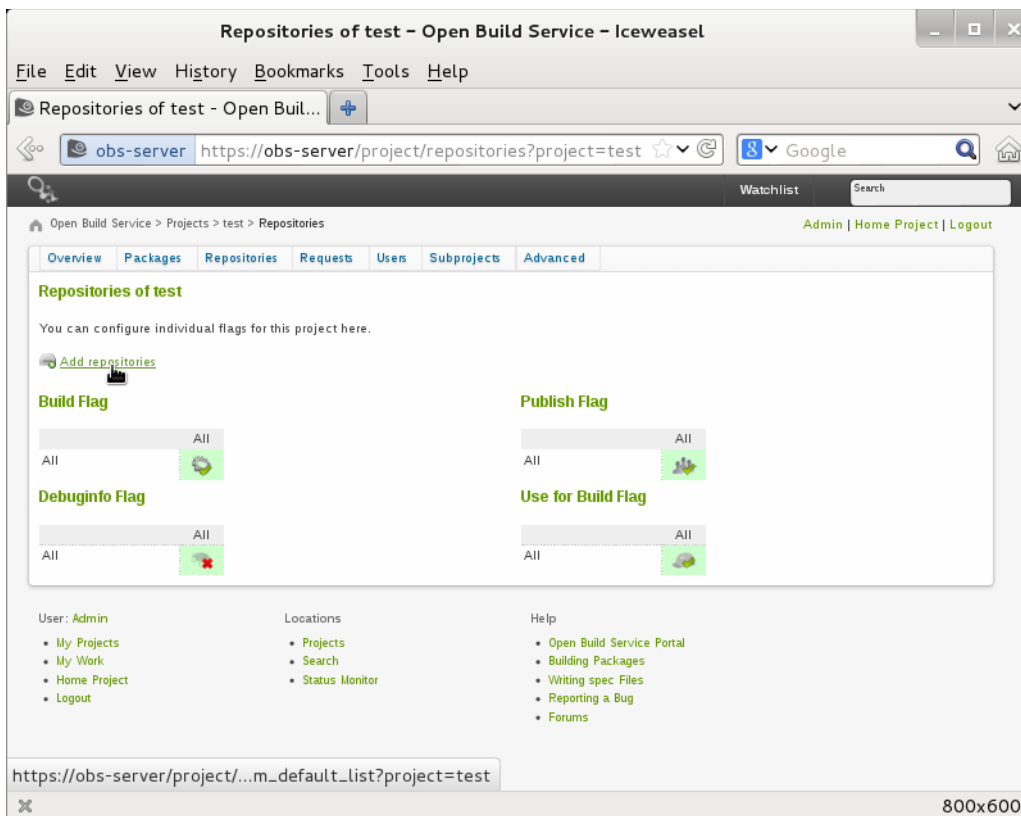
```
# /usr/lib/obs/server/obs-dod-check.pl debian:wheezy:main
running as user: root
Download on Demand enabled: 1
repositories: /srv/obs/build
eventroot: /srv/obs/events
projectsdir: /srv/obs/projects
Project name: debian:wheezy:main
== Checking DoD for debian:wheezy:main main i586 ==
Metafile: Packages
Baseurl: http://ftp.uk.debian.org/debian/
Download type: debmd
Meta file: /srv/obs/build/debian:wheezy:main/main/i586/:full/Packages
Package metadata contains 36098 packages
No solv file found!
Asking the scheduler to recheck
Waiting 5 seconds for the solv file to appear...
solve data contains 36098 packages
Checking metadata vs. solv...
Checking solv vs. metadata...
== Checking DoD for debian:wheezy:main main x86_64 ==
Metafile: Packages
Baseurl: http://ftp.uk.debian.org/debian/
Download type: debmd
Meta file: /srv/obs/build/debian:wheezy:main/main/x86_64/:full/Packages
Package metadata contains 35985 packages
No solv file found!
Asking the scheduler to recheck
Waiting 5 seconds for the solv file to appear...
solve data contains 35985 packages
Checking metadata vs. solv...
Checking solv vs. metadata...
== Checking DoD for debian:wheezy:main main armv7hl ==
Metafile: Packages
Baseurl: http://ftp.uk.debian.org/debian/
Download type: debmd
Meta file: /srv/obs/build/debian:wheezy:main/main/armv7hl/:full/Packages
Package metadata contains 35146 packages
No solv file found!
Asking the scheduler to recheck
Waiting 5 seconds for the solv file to appear...
solve data contains 35146 packages
Checking metadata vs. solv...
Checking solv vs. metadata...
```

Once the DOD project is correctly configured it can be used by other projects as a repository source.

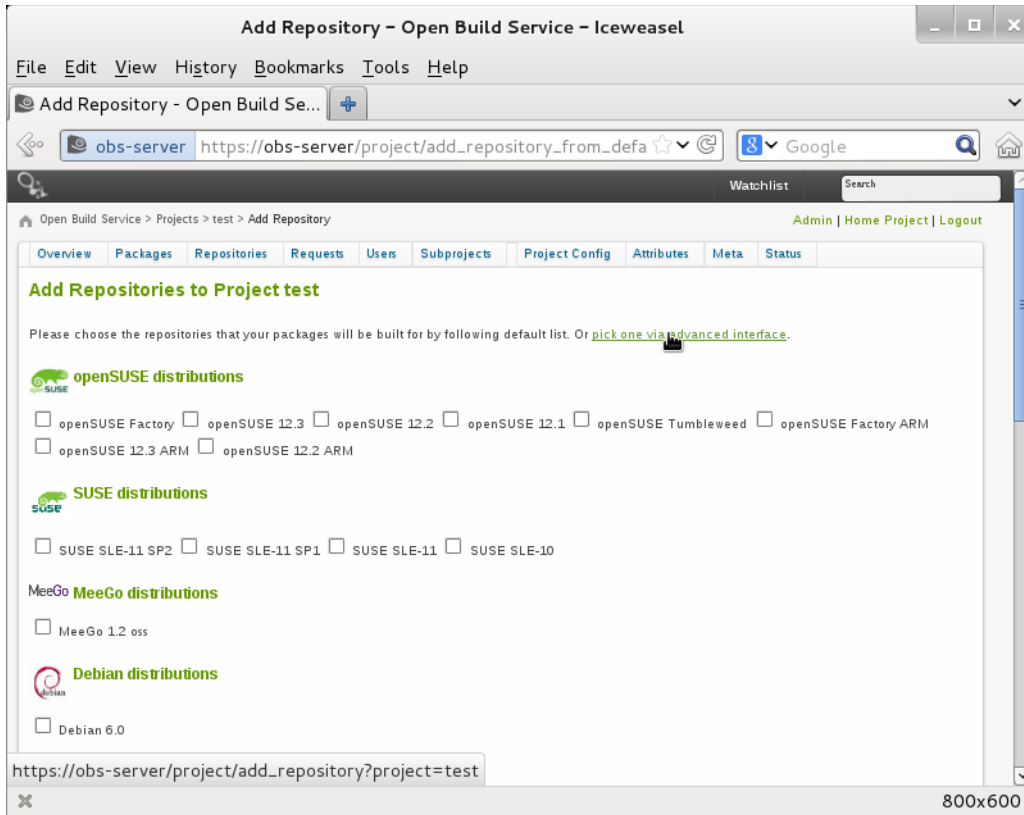


## 6. Add dependent project

A new project should be created (e.g. debian:wheezy:test) and add repository selected



Select "add a new repository".

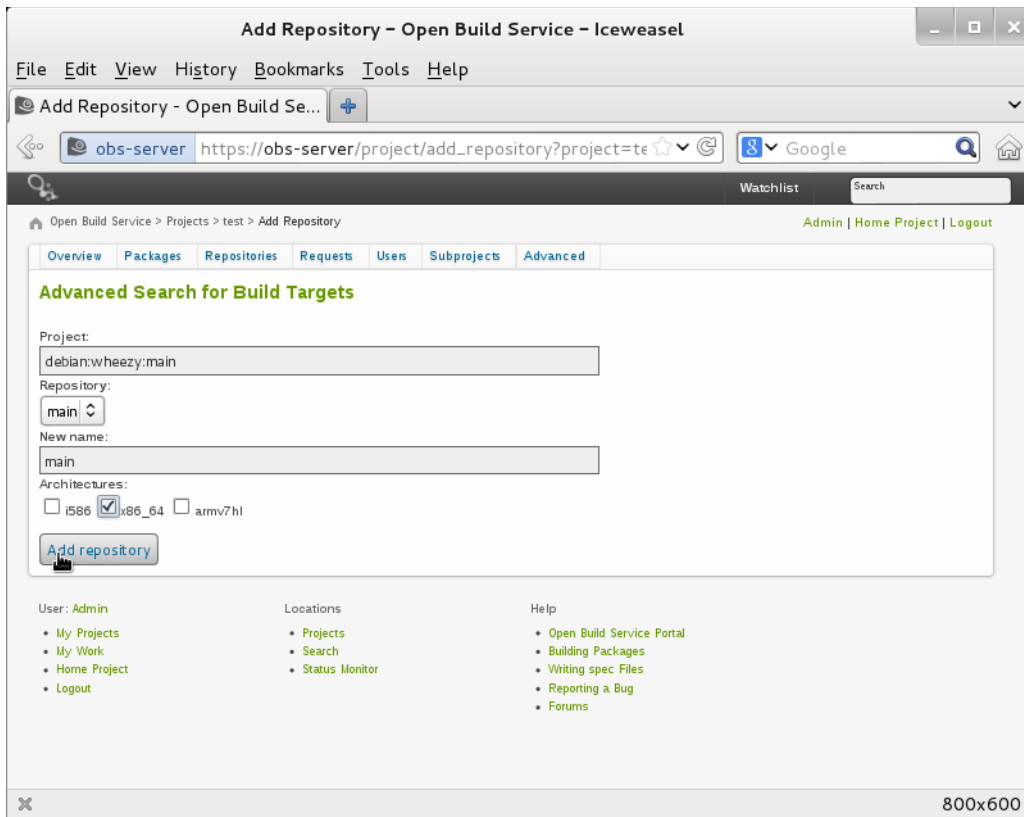


Select the "pick one with advanced interface" to add the repository information.

Enter the details:



```
project: debian:wheezy:main
repository: main
new name: main
```



Then select the the architectures this project should build packages for and select “Add repository”.

Packages can now be added to this project and will be built using the Debian wheezy packages to satisfy dependencies which will be downloaded as required for the build.

If workers for the selected architecture are available the packages will be built.

### Binary import

Binary import is the most raw and resource-heavy way of bootstrapping a project. The process is similar to DOD in terms of using server resources, but now it needs to copy all binary packages to the repository.

- Define your architectures
- Define the name of your repository
- Access the server OBS directory
- Create the directories needed `<Project>/<Repository>/<Arch>/:full`
- Copy the binary packages from the download server. ALL of them.
- Restart the schedule.

For a more real-world example, let's assume that we want a set of i586 Ubuntu packages, from <http://archive.ubuntu.com>, our project will be called UbuntuPrecise, repository will be called

standard, and we are administrator of machine.

First, we access the server and go to the OBS build directory. The standard is `/srv/obs/build`

- Stop the scheduler. Make sure it is really stopped by looking at the log.
- Create the project directory with repository, architecture and the OBS standard repository directory, `:full` (if not created).
- Enter the directory and copy all pool packages from remote pool. This will be difficult since regular Debian pool repositories have all deb version in same place. Usually, a faster way is to parse the Packages file and scan for required deb files. A rough script for doing this is provided below.
  - If you are forking a local project, there is also available an OBS native tool for cloning repository binaries:

```
# /usr/lib/obs/server/bs_admin --clone-repository <distro A> <repo> <distro B> <repo>
```

- Then, include sources, using `osc copypac`:

```
# osc copypac <distro A> <package> <distro B> -m "Branch from A to B"
```

- Restart the scheduler.

```
# /etc/init.d/obsscheduler restart
```

If you are doing a binary import from an upstream distribution, find the following example to bootstrap the initial distribution.

```
# cd /srv/obs/build
# rcobsscheduler stop ; tail /srv/obs/log/scheduler_*.log ;
# mkdir -p UbuntuPrecise/standard/i586/\:full
# cd UbuntuPrecise/standard/i586/\:full
# wget http://archive.ubuntu.com/ubuntu/dists/precise/main/binary-i386/Packages.bz2 && bunzip2 Packages.bz2
# for package in `cat Packages |grep -i filename | sed -e "s,Filename:,,g"`; do
#   wget http://archive.ubuntu.com/ubuntu/$package
# done
# rm Packages
# chown -R obsrun. /srv/obs/build/UbuntuPrecise
# rcobsscheduler restart
```

## 6.1.2 Managing the scheduler

When dealing with OBS projects stopping the scheduler at the right point is important as they do various jobs, in this context the important ones are:

- Create new repositories
- Schedule packages for building
- Maintain the build results in the repositories

Before we can clone the binaries the repositories must be created and because repository

creation is scheduled by changing the configuration the schedulers must be running at that point.

After repository configuration source and binary import commences. If the schedulers are running at this point then depending on order of operation one of the following happens:

- If the source is imported first builds are scheduled for binaries that are going to be imported.
- If the binaries are imported first they will be removed again as there is no source package to back them.

To avoid these cases the schedulers must be off during the whole import process.

## 7 Managing repositories with reprepro

Reprepro<sup>5</sup> is a tool to handle local repositories of Debian packages. Collabora has replaced OBS tools that handle Debian repositories (*dpkg-scanpackages*), because those tools do not scale well when projects have lots of packages and repositories created are flat<sup>6</sup> repositories, while Debian repositories have their own layout<sup>7</sup> and those are required when standard Debian tools are used. However, from hooking up reprepro to OBS, there are some new benefits as well as new limitations.

### 7.1 Using reprepro along with OBS

By default, OBS projects are set to use OBS style (flat) repositories. To get reprepro enabled you need to specify in the project configuration file the following information:

```
Repotype: debian
type: dsc
```

Every time a new project is created, in order to have Debian repository format, new configuration needs to be created under */srv/obs/repos/shared/<distro-local>/*

- Edit *conf/distributions*, add repository configuration data<sup>8</sup>. As an example:

```
Origin: Distro Local
Label: Distro local repository
Codename: suiteA
AlsoAcceptFor: suiteA
Architectures: amd64 i386 armhf source
Components: core
UdebComponent: core
Description: Distro local repository
SignWith: <signing_key>
Contents: .bz2
DebIndices: Packages Release . .gz
Tracking: all includechanges keepsources
Log: reprepro.log
```

More blocks can be added to *conf/distributions* file if repository needs to hold more than one suite.

- Proceed creating indexes and symlinks for new repository

```
# sudo -u obsrun reprepro export
# sudo -u obsrun reprepro createsymlinks
```

- Debian format repositories are now created and managed by reprepro but OBS still needs to know about those, editing */etc/obs/BSCConfig.pm*, add new block towards end of file:

```
our $reprepository = {
    "distro-local:suiteA:componentA/repository" =>
        { "repository" => "shared/distro-local",
          "codename" => "suiteA",
          "component" => "componentA" },
};
```

5 <http://mirrorer.alioth.debian.org/>

6 [https://wiki.debian.org/RepositoryFormat#Flat\\_Repository\\_Format](https://wiki.debian.org/RepositoryFormat#Flat_Repository_Format)

7 [https://wiki.debian.org/RepositoryFormat#Debian\\_Repository\\_Format](https://wiki.debian.org/RepositoryFormat#Debian_Repository_Format)

8 See *reprepro* manual page

```
1;
```

- Add as many blocks as you need.
- Restart the publisher so it reloads its configuration
- Check the publisher log at `/srv/obs/log/publish.log` and see that sources are included into shared repository via including its dsc file and binaries are getting included via changes file.

```
# service obspublisher restart
# tailf /srv/obs/log/publish.log
```

## 7.2 Architecture all packages

We build architecture all packages on all architectures afterwards these get put into a Debian repository in reprepro (once per project). When this publishing happens it is decided which architecture all package makes it into the final project repository (which is why it's important to have global unique versioning for all our packages). In case a project is forked new repositories are created in the OBS instance, in this case it can occur that a different architecture all package is selected (e.g. armhf instead of i386). When that happens reprepro gets upset as the checksum of an already existing architecture all package changed.

Collabora reprepro has been modified to include packages that their architecture all packages are already included in repository (because of those entered from other architecture previously).

## 7.3 Removing/adding packages from shared repository

When a package is removed from OBS, Debian shared repositories do not automatically delete such package, that needs to be done manually server side. Note that all binaries (for all architectures) and sources will be deleted.

```
# cd /srv/obs/repos/shared/<distro-local>
# reprepro removesrc <suiteA-local> <package>
```

If a package is not found in the Debian shared repository, it can be deleted from OBS package publisher directory and next publisher run, the package will be included in the Debian shared repository. If it does not get included, some issue might be happening, please check the OBS publisher log which should contain information about what is going on.

```
# cd /srv/obs/repos/<distro-local>/<repository>/
# rm -rf <package>* ## will delete source packages

## will delete binaries for given architecture
# dcmd rm <arch>/<package>*changes

## trigger publisher which will include packages in Debian repository
```

Version	Date
v0.3	2014-06-16

```
# /usr/lib/obs/server/bs_admin -publish-repository <distro-local>  
<repository>
```

## 8 Configure signing keys

### 8.1 Assumptions & Requirements

This document assumes an OBS instance has already been deployed as documented, with all Collabora patches applied. It is also assumed that OBS resides in a private server, with its repositories copied to a second server that offers public access.

This setup will provide signing and authentication for:

- Flat .deb repositories as created by the OBS Publisher
- Debian-style poolized repositories created by reprepro using the flat repositories as sources.
- Public reprepro repositories on the repository server, signed by a release key and verified against the OBS key.

### 8.2 GnuPG Keys Creation

First of all, we will create two pairs of GnuPG keys, one for the OBS server, and another one for the public repositories. The latter will be published via a keyring package that APT will use to verify repository contents against forgery.

#### 1.Note about security

It is important to keep the GnuPG secret keys safe. Potentially they should never be available on a device that is connected to a public network, or accessible to anyone who isn't in charge of guarding them. If the keys are compromised, the course of action should be to publish a revocation certificate to mark them invalid.

#### 2.OBS GnuPG Key

We will work on an alternative gnupg home to avoid mixing already existing keys.

```
$ mkdir gnupghome
$ cd gnupghome
$ gpg --homedir . --gen-key
gpg: WARNING: unsafe permissions on homedir `.'
gpg (GnuPG) 1.4.12; Copyright (C) 2012 Free Software Foundation,
Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
```

```
Requested keysize is 2048 bits
Please specify how long the key should be valid.
  0 = key does not expire
  <n> = key expires in n days
  <n>w = key expires in n weeks
  <n>m = key expires in n months
  <n>y = key expires in n years
Key is valid for? (0) 2013-11-01
Key expires at dv 01 nov 2013 00:00:00 CET
Is this correct? (y/N) y
```

```
You need a user ID to identify your key; the software constructs
the user ID
from the Real Name, Comment and Email Address in this form:
  "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"
```

```
Real name: OBS Automatic Signing Key
Email address: example@example.com
Comment:
You selected this USER-ID:
  "OBS Automatic Signing Key <example@example.com>"
```

```
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key.
```

```
You don't want a passphrase - this is probably a *bad* idea!
I will do it anyway. You can change your passphrase at any time,
using this program with the option "--edit-key".
gpg: key 01234567 marked as ultimately trusted
public and secret key created and signed.
```

After a few seconds, the GnuPG key will have been created.

We now need to edit it and add some subkeys. We will add a DSA signing key, and a ElGamal encryption key.

```
$ gpg --homedir . --edit-key 01234567
gpg (GnuPG) 1.4.12; Copyright (C) 2012 Free Software Foundation,
Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Secret key is available.

pub 2048R/01234567  created: 2012-11-05  expires: 2013-10-31
usage: SC
                                trust: ultimate      validity: ultimate
sub 2048R/89ABCDEF  created: 2012-11-05  expires: 2013-10-31
usage: E
[ultimate] (1). OBS Automatic Signing Key <example@example.com>

gpg> addkey
This key is not protected.
```



```

Please select what kind of key you want:
  (3) DSA (sign only)
  (4) RSA (sign only)
  (5) Elgamal (encrypt only)
  (6) RSA (encrypt only)
Your selection? 3
DSA keys may be between 1024 and 3072 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits
Please specify how long the key should be valid.
  0 = key does not expire
  <n> = key expires in n days
  <n>w = key expires in n weeks
  <n>m = key expires in n months
  <n>y = key expires in n years
Key is valid for? (0) 2013-11-01
Key expires at dv 01 nov 2013 00:00:00 CET
Is this correct? (y/N) y
Really create? (y/N) y

pub 2048R/01234567  created: 2012-11-05  expires: 2013-10-31
usage: SC
                                trust: ultimate      validity: ultimate
sub 2048R/89ABCDEF  created: 2012-11-05  expires: 2013-10-31
usage: E
sub 2048D/DEADCODE  created: 2012-11-05  expires: 2013-10-31
usage: S
[ultimate] (1). OBS Automatic Signing Key <example@example.com>

gpg> addkey
This key is not protected.
Please select what kind of key you want:
  (3) DSA (sign only)
  (4) RSA (sign only)
  (5) Elgamal (encrypt only)
  (6) RSA (encrypt only)
Your selection? 5
ELG-E keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits
Please specify how long the key should be valid.
  0 = key does not expire
  <n> = key expires in n days
  <n>w = key expires in n weeks
  <n>m = key expires in n months
  <n>y = key expires in n years
Key is valid for? (0) 2013-11-01
Key expires at dv 01 nov 2013 00:00:00 CET
Is this correct? (y/N) y
Really create? (y/N) y

pub 2048R/01234567  created: 2012-11-05  expires: 2013-10-31
usage: SC
                                trust: ultimate      validity: ultimate
sub 2048R/89ABCDEF  created: 2012-11-05  expires: 2013-10-31
usage: E
sub 2048D/DEADCODE  created: 2012-11-05  expires: 2013-10-31

```

```
usage: S
sub 2048g/C0FFEE51 created: 2012-11-05 expires: 2013-10-31
usage: E
[ultimate] (1). OBS Automatic Signing Key <example@example.com>
save
```

### 3.Public repository GnuPG key

We will repeat the same process to create a second key, for the public repository. A suggested name would be Distro Archive Signing Key (codename). This key will be renewed every time a new release is about to be published, and the collabora-archive-keyring will be updated to include it.

### 4.Exporting the keys

Once both keys are created, we will export the keys using `gpg --homedir . --export -a -o pubkeys.asc` and `gpg --homedir . --export-secret-keys -a -o seckey.asc`.

Additionally, we will create a revocation certificate for both keys, using `gpg --homedir . --gen-revoke keyid`. It is wise to store this certificate in very safe external media, which isn't permanently plugged into our servers or workstations.

## 8.3 Adding keys to the keyring package

After creating the key pairs, we will need to have their public keys updated in the distro-archive-keyring, so as soon as the build infrastructure starts using newly generated keys, all the devices using the archive will get the new key in advance, avoiding validation errors.

Updating the public keys in the keyring package is a matter of replacing the file `archive.asc`, which contains the current keys to be exported to the devices, with another file of the same name with additional or replaced keys.

In our example above, copying the contents of `pubkeys.asc` would replace its previous contents with the two generated keys, for example.

Once `archive.asc` contains all the desired keys, the package should be rebuilt (using a bumped version) and pushed to the archive so all the archive users have access to the new signing key.

## 8.4 Enabling signatures for OBS-created repositories

Our next step will be to enable signing at the very first step of the package creation process. OBS creates dumb "flat" Debian repositories via its publisher service. These repositories are then massaged to a format more ready for consumption via some scripts that mirror them in "pool" format using the reprepo repository manager. In order to get OBS to sign these initial repositories, we will enable its Signer service, and deploy our secret key into the OBS server.

### 8.4.1 Setting up GnuPG

On the OBS server, we need to make the secret OBS gnupg key available to the user that runs

all OBS services (obsrun). While the correct place to import the GnuPG key in `/usr/lib/.gnupg`, OBS does look for keys in `/obs/gnupg`, so we will symlink `/usr/lib/.gnupg` to `/obs/gnupg`. We will then import the OBS secret key:

```
# sudo -u obsrun gpg --import obsseckey.asc
```

As our key has no passphrase, we will not need to add the passphrase to `/obs/gnupg/phrases`.

We will also place a copy of the exported ascii-armored key in `/srv/obs/gnupg/public_obs_key.asc`

## 8.4.2 Setting up the OBS Signer service

We will make sure the sign binary is available, or will install the `obs-signd` package if it isn't. To enable the OBS Signer service, we will edit the `/etc/obs/BSConfig.pm` configuration file, adding the following parametres:

```
OBS signer changes
our $gpg_standard_key = "/srv/obs/gnupg/public_obs_key.asc";
# The above should be something like "/etc/keyfile.asc".
our $sign = '/usr/bin/sign';
# Extend sign call with project name as argument "--project $NAME"
our $sign_project = 0;
# Global sign key
our $keyfile = '/srv/obs/gnupg/public_obs_key.asc';
# Create a key by default for new projects, if top level have not
one
#our $forceprojectkeys = 1;
```

We will then edit `/etc/sign.conf` to make it look like this:

```
user: example@example.com
server: 127.0.0.1
allowuser: obsrun
allow: 127.0.0.1
phrases: /srv/obs/gnupg/phrases
hash: sha256
```

The hash parameter is important, otherwise the Signer might be unable to use our generated key.

After the configuration is in place, it's time to start the Signer service:

```
# insserv obssignd obssigner
# rcobssignd start
# rcobssigner start
```

Restarting the Publisher service is also recommended so it can pick up or previous changes:

```
# rcobspublisher restart
```

After doing this, the Signer service should appear as enabled in the OBS monitor at the web interface, as all other OBS services.

Newly or updated repositories, projects or subprojects in OBS should start gaining Release.gpg files, which will indicate signing is working as expected.

### ***8.5 Creating and verifying signatures on Debian-style reprepro repositories***

Once OBS has started to sign its generated repositories, we will move forward to the next step in the verification chain. We will make sure the poolized Debian repositories created by bs\_reprepro validate the cryptographic signature from the OBS flat repositories, and also signs these repositories again, so they can keep being verified down the chain.

We will edit the distribution file in `/srv/obs/repos/shared/$dist/conf/`.

The distributions generated config file should have a SignWith directive, with the key identifier of our OBS key.

The result should look like this:

```
Codename: ${codename}
#Suite: ${projid}
AlsoAcceptFor: ${alsoaccept}
Version: 0.1
Origin: ${distro}
Description: Debian repository ${_reproid} for project $
${_projid}
Architectures: ${arches}
Components: ${component}
#UDebComponents: main
DebIndices: Packages Release . .gz
#UDebIndices: Packages Release . .gz
SignWith: 01234567
Tracking: all includechanges keepsources
Log: reprepro.log
Update: - up-from-${hostname}
```

Force `./dists/$dist/Release.gpg` file creation issuing

```
$ sudo -u obsrun -Vb /srv/obs/repos/shared/$dist export
```

Once this happens, the repositories in `/srv/obs/repos` will slowly start generating Release.gpg files as well as activity starts happening in each individual repository. The generation of the signatures can be forced at any time by running `sudo -u obsrun reprepro export` on each repository.

Version	Date
v0.3	2014-06-16

## ***8.6 Configure BZ hooks***

TBD – Document BZ hooks?

## 9 Configure merge-o-matic (mom)

Merge-o-matic<sup>9</sup> is a tool to handle automated merge of packages from two different distributions. It also generates useful reports to aid manual merges of packages that cannot be merged automatically. Status web pages display merged packages as well as packages with outstanding merges that might need human intervention.

The tool was initially created by Ubuntu to automatically merge changes from Debian source packages. Collabora has enhanced the tool to be able to handle OBS repositories, however, there is an assumption for merge-o-matic to work, which needs to follow the convention on OBS project naming: `<distro_name>:<suite>:<component>`.

### 9.1 Install and configure merge-o-matic

In order to use merge-o-matic (mom) we need to set it up and configure it.

```
# apt-get install merge-o-matic
```

- Edit `/etc/merge-o-matic/momsettings.py` as needed
  - note: distro and component names must match current names for the project. Make sure to update first entry on DISTROS and its components. Also update `defineDist` function calls.
  - note: to use the external collabora servers, use `https://user:password@repositories.<PROJECT>.collabora.co.uk/<DISTRO>/` in the `DISTRO_SOURCES` and `DISTROS` sections. An example follows,

```
DISTROS = {
    "distro-local": {
        "obs": {
            "url": "https://%s:444" % _OBS_SERVER,
            "web": "https://%s" % _OBS_SERVER,
            "project": "distro-local",
        },
        "mirror": "http://%s:82/shared/distro-local/" % _OBS_SERVER,
        "dists": [ "suite-localA", "suite-localB", "suite-localC" ],
        "components": [ "component-localA", "component-localB" ],
        "expire": True,
    },
    "distro-upstream": {
        "mirror": "https://user:password@repositories.distro-
upstream.collabora.co.uk/<DISTRO>/",
        "dists": [ "suite-upstreamA", "suite-upstreamB", "suite-upstreamC" ],
        "components": [ "component-upstreamA", "component-upstreamB" ],
        "expire": True,
    }
}

DISTRO_SOURCES = {
    "suiteA+updates": [
        {"distro": "distro-upstream", "dist": "suite-upstreamA"},
    ],
    "suiteB+updates": [
        {"distro": "distro-upstream", "dist": "suite-upstreamB"},
    ],
    "suiteC+updates": [
```

<sup>9</sup> <https://wiki.ubuntu.com/MergeOMatic>

Version	Date
v0.3	2014-06-16

```

    {"distro": "distro-upstream", "dist": "suite-upstreamC"},
  ]
}

```

```

defineDist('distro-local', 'suite-localA', 'suiteA+updates', False)
defineDist('distro-local', 'suite-localB', 'suiteB+updates', False)
defineDist('distro-local', 'suite-localC', 'suiteC+updates', False)

```

Finally, teach merge-o-matic about which suffix is used on packages with local changes, so merge-o-matic knows that a package with such suffix contains local modifications that need to be merged when newer upstream package is available from upstream distribution. Collabora uses 'co1' suffix, so packages modified by Collabora have a revision number like:

- **<version>-Oco1**: new upstream version package (only in our local distribution)
- **<version>-1co1**: upstream distribution package versioned with <version>-1 adds some modifications done by Collabora.
- **<version>-1.1ubuntu5co3**: upstream distribution, Debian has <version>-1.1, Ubuntu added 5 changes on top of Debian version, then Collabora added 3 changes on top Ubuntu.

```
LOCAL_SUFFIX = "co1"
```

- Create an OBS account for use with merge o-matic, go to [https://obs-server/user/register\\_user](https://obs-server/user/register_user) and fill in the fields:
  - Login: merge-o-matic
  - Email: mom@obs-server
  - Real Name: Merge-o-Matic
  - Password: <password of your choosing>
  - Password Confirmation: <same as above>
- Logout of OBS and log back in as the Admin user
- Add merge-o-matic to the DISTRO:RELEASE\_CODENAME:COMPONENT project on [https://obs-server/project/add\\_person?](https://obs-server/project/add_person?project=DISTRO:RELEASE_CODENAME:COMPONENT) project=DISTRO:RELEASE\_CODENAME:COMPONENT as maintainer
- As root on the obs server, change user to mom and initialise it's osc configuration by running the following commands, filling in the username and password as used for merge-o-matic in OBS and permanently accepting the ssl certificate:

```

# su - mom -s /bin/bash
$ osc -A https://obs:444 ls
osc will now prompt for a configuration

```

```
$ exit
```

- If the previous step goes well a list of all packages will be shown in the output, at this point an initial manual merge-o-matic run can be done (Note that this will take a long time):

```
# cd /srv/obs/merge-o-matic/
# sudo -u mom /usr/lib/merge-o-matic/main.py
```

- Be sure a mom user account exists in the system running merge-o-matic service.
- Set a crontask for daily merge-o-matic pass (*/etc/cron.d/merge-o-matic*):

```
## crontab for merge-o-matic runs
MAILTO="sysadmin@domain.host"
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
LOG="/srv/obs/log/mom.log"

# min      hour      day mon wday user      cmd
01         11        * * *    mom      (cd /srv/obs/merge-o-matic/ &&
/usr/lib/merge-o-matic/main.py) > "$LOG" 2>&1
```

- The results of the last merge-o-matic run can be found on: <http://obs-server:83/merges/SUITE-COMPONENT.html>
- If there are new updates, *merge-o-matic* will merge changes and try a test build at *home:mom:branches:DISTRO:RELEASE\_CODENAME:COMPONENT* and submit a merge request to project for a project maintainer to review and accept into main project.

## 9.2 Using merge-o-matic

*/usr/lib/merge-o-matic/main.py* is the main Merge-o-Matic executable. Called without arguments, it will attempt to sync the entire OBS project. With a *-p packagename* argument, it will only sync the specified Debian package.

If necessary, the individual phases of a Merge-o-Matic run can also be called manually by separate commands:

- **update\_pool.py**: updates (or checks out) the OBS project using osc and generates a Debian source repository from it in */srv/obs/merge-o-matic/pool*. Downloads those packages from the upstream distribution's Debian source repository that were found in the OBS project into another subdirectory of */srv/obs/merge-o-matic/pool*.
- **update\_sources.py**: updates Sources files in the pools. These are required for all further phases.
- **generate\_diffs.py**: generate diffs between successive versions of packages in a distro, and store them in */srv/obs/merge-o-matic/diffs*. These can be useful if a manual merge is required.
- **generate\_dpaches.py**: extract Debian patches from source packages. These can be useful if a manual merge is required.
- **publish\_patches.py**: generate web reports for the above in */srv/obs/merge-o-*



*matic/published.*

- **produce\_merges.py**: the main step. If possible, merge the latest version from the upstream repository with the changes in the package from OBS.
- **commit\_merges.py**: commit packages that were successfully merged to OBS.
- **stats.py, stats\_graphs.py**: generate merge statistics.
- **merge\_status.py**: generate the main report file (e.g. */srv/obs/merge-o-matic/merges/distro-standard.html*)
- **expire\_pool.py**: punt unneeded files from the pool.

### 9.3 Configuration files

- */etc/merge-o-matic/momsettings.py* : main configuration file.
- */etc/apache2/vhosts.d/mom.conf* : Apache virtual host configuration; allows MOM's output files to be accessible via a browser.
- */etc/cron.daily/merge-o-matic* : the main cron job.
- */srv/obs/merge-o-matic/sync-blacklist.txt* : one-per-line list of Debian package names that MOM will ignore.
- */srv/obs/merge-o-matic/comments.txt* : per-package comments, will be integrated into the main MOM report file.

### 9.4 Output files

- */srv/obs/merge-o-matic/merges/distro-core.html*: the main Merge-o-Matic report for the distro core repository. Packages that will need to be merged manually by a human are listed in the "Outstanding Merges" section. Packages that were merged and committed into OBS automatically by MOM will be listed in the "Committed Merges" section. Other reports are located in that same directory.
- */srv/obs/merge-o-matic/merges/\*/\*/REPORT* : per-package merge reports.
- */srv/obs/log/mom.log* : default log file for the MOM cron job.
- */srv/obs/merge-o-matic/stats.txt* : merge statistics.

## 10 OBS worker installation

The host machine should be prepared as described in the Host Operating System section.

### *10.1 The worker package should be installed with apt*

```
# apt-get install obs-worker
```

The worker is configured by editing the `/etc/default/obsworker` file.

The main parameters that should be set are:

- Enable worker
- Source and repository server IP address and ports
- Number of worker instances
- Number of parallel jobs
- Worker build directory
- OBS master name

Ensure to dimension the worker accordingly to machine specification.

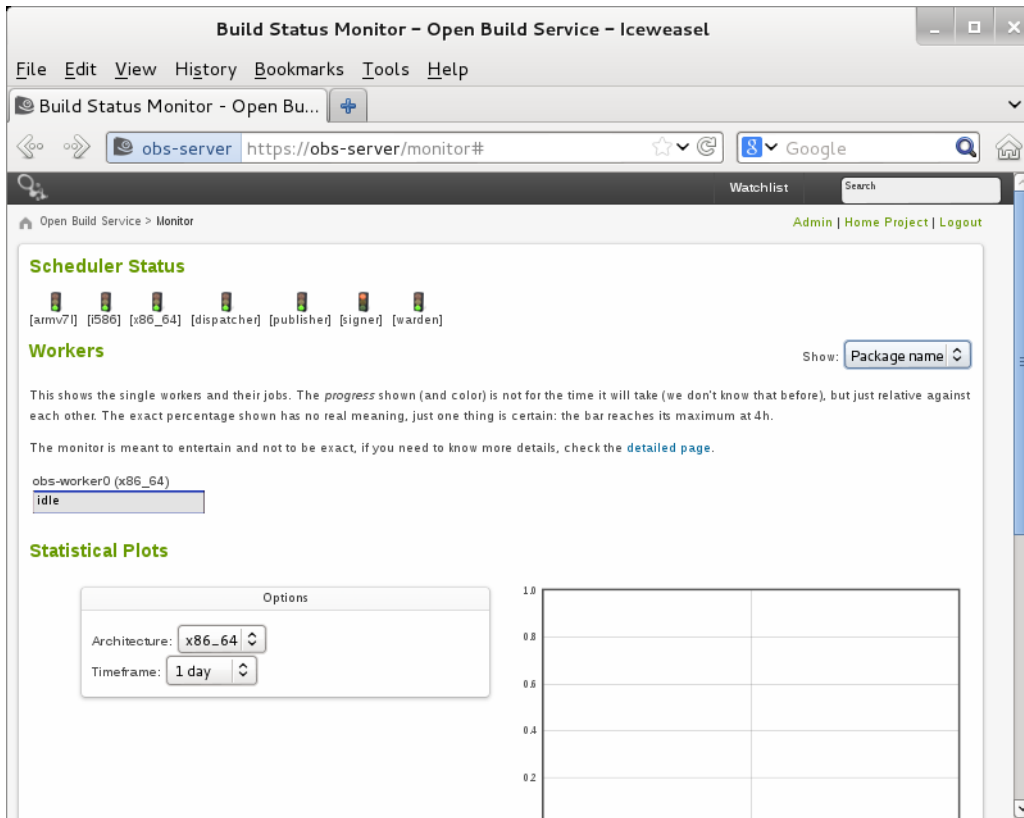
Verify worker is able to resolve the master OBS instance name to an IP address and proceed with starting the worker:

```
# /etc/init.d/obsworker start
Run 6 obsworker using /srv/obs/worker
Type of obsworker is chroot
Fetching initial worker code from http://obs-server:5252
Fetched worker code from server.
```

If everything goes as expected, code is fetched from the OBS server and worker can be seen using the WebUI on the monitor page<sup>10</sup>.

---

<sup>10</sup> <https://obs/monitor>



The screenshot shows a web browser window titled "Build Status Monitor - Open Build Service - Iceweasel". The address bar shows "https://obs-server/monitor#". The page content includes:

- Scheduler Status:** A row of six small colored icons representing different scheduler components: [armv7l], [i586], [x86\_64], [dispatcher], [publisher], [signer], and [warden].
- Workers:** A section with a "Show:" dropdown menu set to "Package name". Below it, a text description explains that the progress bars are relative and reach their maximum at 4h. A specific worker "obs-worker0 (x86\_64)" is shown with an "idle" status.
- Statistical Plots:** A section with an "Options" box containing "Architecture: x86\_64" and "Timeframe: 1 day". To the right is a large empty grid for a chart with a y-axis ranging from 0.2 to 1.0.

Version	Date
V0.3	2014-06-16

## 11 OSC API client tool

At the moment there is no custom Collabora OSC documentation, please for now refer to OSC<sup>11</sup> main site.

---

<sup>11</sup> <http://en.opensuse.org/openSUSE:OSC>

## Appendix A – Files

### *Generic Project configuration (prjconf) for Debian repository type*

```

Repotype: debian
type: dsc
release: ba<B_CNT>

# Need devscripts for binary build numbering
Preinstall: devscripts

# assumed by some packages even though it's not essential
Preinstall: lockfile-progs

Preinstall: apt base-files mawk base-passwd bash binutils bsutils
initscripts build-essential
Preinstall: bzip2 coreutils cpp cpp-4.6 dash debconf debconf-i18n
debianutils diffutils
Preinstall: libc-bin libc-dev-bin libgcc1 libc6 libc6-dev e2fslibs
e2fsprogs fakeroot findutils g++ g++-4.6 gcc
Preinstall: gcc-4.6 gcc-4.6-base gnupg gpgv grep gzip hostname
insserv
Preinstall: libacl1 libattr1 libblkid1 libbz2-1.0
Preinstall: dpkg dpkg-dev autotools-dev
Preinstall: libclass-isa-perl libcomerr2 libselinux1 libdb5.1
libdpkg-perl libgdbm3
Preinstall: libgmp10 libgomp1 liblocale-gettext-perl liblzma5
libmount1 libmpc2 libmpfr4
Preinstall: libncurses5 libpam-modules libpam-modules-bin libpam-
runtime libpam0g
Preinstall: libreadline6 libsepol1 libslang2 libss2 libstdc++6
Preinstall: libswitch-perl libtext-charwidth-perl libtext-iconv-
perl libtext-wrapi18n-perl
Preinstall: libtimedate-perl libtinfo5 libusb-0.1-4 libuuid1 linux-
libc-dev login lsb-base
Preinstall: make mount multiarch-support ncurses-base ncurses-bin
passwd patch perl
Preinstall: perl-base perl-modules readline-common sed sensible-
utils sysv-rc
Preinstall: tar tzdata ucf util-linux xz-utils zlib1g libsigsegv2
libssl1.0.0
Preinstall: dash bash perl perl-base sed grep coreutils debianutils
Preinstall: libncurses5 libacl1 libattr1 libpcre3 libreadline5
Preinstall: lzma liblzma5 libstdc++6 passwd libdbus-1-3 libdbus-1-
dev
Preinstall: sharutils gettext docutils-common
Preinstall: zlib1g libbz2-1.0 libtinfo5 adduser
Preinstall: g++-4.6-multilib g++-multilib
Preinstall: libffi6 libglib2.0-0 python
%ifarch armv7l
Preinstall: libc6-armel libc6-dev-armel
Order: libc6-armel:libc6-dev-armel
%endif
%ifarch i586
Preinstall: libc6-amd64 libc6-dev-amd64
%endif
Order: libc6:libc-bin

```

```

Order: libc6:libgcc1
Order: xz-utils:libc6
Order: xz-utils:liblzma5
Order: base-files:glibc
Order: dpkg:libc6

Runscripts: base-files base-passwd passwd tzdata sysv-rc

VMinstall: binutils libblkid1 libuuid1 mount

Required: autoconf automake autotools-dev binutils bzip2 gcc libc6
Required: libtool libncurses5 perl zlib1g pkg-config
Required: build-essential
Order: automake:autotools-dev

Support: fakeroot
Support: bison cpio cvs login
Support: file findutils flex diffutils
Support: groff-base gzip hostname info less
Support: make man kmod
Support: net-tools autotools-dev
Support: patch procps psmisc
Support: texinfo unzip ncurses-base
Support: locales
Support: mount mime-support

# circular dependendencies in openjdk stack
Order: openjdk-6-jre-lib:openjdk-6-jre-headless
Order: openjdk-6-jre-headless:ca-certificates-java

# Workaround for missing prerequires:
Preinstall: initscripts

Prefer: libdb-dev
Prefer: xorg-x11-libs libpng fam mozilla mozilla-nss xorg-x11-Mesa
Prefer: unixODBC libsoup glitz java-1_4_2-sun gnome-panel
Prefer: desktop-data-SuSE gnome2-SuSE mono-nunit gecko-sharp2
Prefer: apache2-mpm-prefork openmotif-libs ghostscript-mini gtk-
sharp
Prefer: glib-sharp libzypp-zmd-backend mDNSResponder
Prefer: mawk libjpeg-dev
Prefer: xz-lzma ruby

Prefer: gnome-sharp2:art-sharp2 gnome-sharp:art-sharp
Prefer: ifolder3:gnome-sharp2 ifolder3:gconf-sharp2
Prefer: nautilus-ifolder3:gnome-sharp2
Prefer: gconf-sharp2:glade-sharp2 gconf-sharp:glade-sharp
Prefer: tomboy:gconf-sharp tomboy:gnome-sharp
Prefer: zmd:libzypp-zmd-backend
Prefer: yast2-packagemanager-devel:yast2-packagemanager
Prefer: default-jdk
Prefer: libzephyr4
Prefer: ant ant-optional libpg-java
Prefer: libasound2-dev aspell-dictionary myspell-en-us
Prefer: libblas3gf liblapack3gf

```