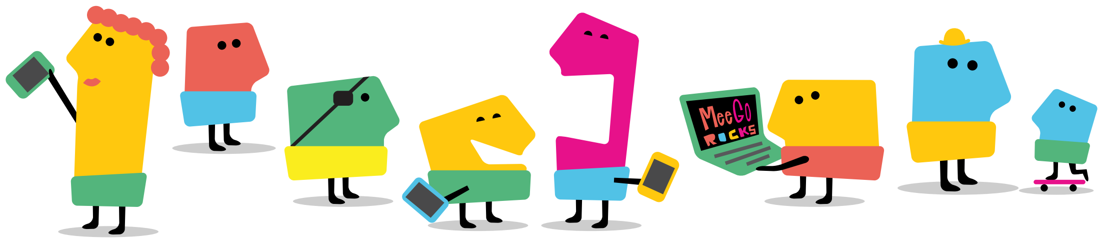
MeeGo™



» The Linux Foundation

Building MeeGo with OBS

An Introduction to the MeeGo Build Infrastructure

By Rudolf Streif, The Linux Foundation

June 2011

A Publication By The Linux Foundation
http://www.linuxfoundation.org

Background

Building a Linux distribution from scratch can be a daunting task. Besides the obligatory Linux kernel itself with its myriad of configuration options, a working Linux distribution requires hundreds if not thousands of additional packages to form a viable platform for desktop, server, mobile or any other type of specialized applications. For embedded Linux solutions, the complexity scales with the need to support multiple and different processor architectures, hardware platforms with limited resources, requirements for cross toolchains, and more.

The MeeGo project leverages the strengths of the Open Build Service (OBS) for distribution and application development. MeeGo has established a 6-months release cadence mounting two releases per year in April and October in addition to maintenance releases in-between. To support this, the MeeGo OBS environment builds bootable MeeGo images for the various MeeGo device categories and user experiences (Handset, Netbook, Tablet PC, In-vehicle Infotainment, and Smart TV) targeting multiple different hardware platforms for IA, ARM, etc. For application development, the MeeGo OBS allows developers to easily build their software packages utilizing the same infrastructure the MeeGo distributions were built with, validating dependencies and providing binary compliance.

In this whitepaper, we will provide a brief introduction to the MeeGo build infrastructure and how it is utilized to support the MeeGo development process. Then we demonstrate the use of the OBS WebUI and the command-line client osc with step-by-step instructions. You may want to follow through the sections with your own computer to gain hands-on experience using OBS.

This whitepaper represents the second paper in a series of three papers on OBS. If you are looking for a general introduction into OBS, its capabilities, features and components then [1] will be the right start. If you are already an OBS user and have experienced the WebUI as well as its command-line client and are looking on how you can deploy your own OBS instance then the third paper of the series [2] will provide you with information and step-by-step instructions.

Getting Involved with MeeGo

MeeGo is a Linux-based platform built for the next-generation of computing devices. Different from other mobile platforms, MeeGo is open source and includes the core operating system, user interface libraries and tools, reference user experiences for multiples devices and applications, a standard set of APIs across all target device types. MeeGo supports a magnitude of mobile client devices also called verticals and includes: Handsets, Smart TVs, In-Vehicle Infotainment , Netbooks and Tablets. It provides choice and flexibility to create and deliver a uniquely differentiated service offering with the flexibility to support proprietary add-ons. Getting involved in MeeGo is as easy as visiting MeeGo.com. Please see below for pointers to various core areas in MeeGo.
MeeGo Project http://www.meego.com and http://www.meego.com/about
Getting a MeeGo.com Account http://meego.com/user/register
MeeGo Wiki http://wiki.meego.com/Main_Page
Developers Resources http://meego.com/developers
Mailing Lists http://meego.com/community/mailing-lists
IRC Discussions http://meego.com/community/irc-channel
MeeGo Forums http://meego.com/community/forum

Getting Involved with MeeGo Cont.

TSG Meetings http://wiki.meego.com/Technical_Steering_Group_meetings
MeeGo Projects http://meego.com/projects
MeeGo Source Code http://meego.gitorious.org/
MeeGo Bugzilla http://bugs.meego.com/
MeeGo Forums http://meego.com/community/forum
Pre-Compiled Images http://meego.com/downloads
MeeGo Releases http://meego.com/downloads/releases
MeeGo SDK http://meego.com/developers/getting-started
MeeGo OBS http://wiki.meego.com/Build_Infrastructure

The MeeGo OBS

MeeGo utilizes two OBS: the *MeeGo Build Service* used for building the MeeGo OS itself and the *MeeGo Community Build Service* used by application developers to build applications for MeeGo.

The MeeGo OBS build farm is hosted by The Linux Foundation at Oregon State University. It is, at the time of writing, running version 2.1.1 serving over 880 users building more than 1700 projects of which about 1150 are user projects and the remainder are MeeGo development branches. These projects contain more than 19,000 packages maintained in over 2,700 repositories. To provide all users a decent experience for building their projects and packages as well as supporting regular weekly builds of MeeGo on different architectures, the MeeGo OBS farm currently deploys 36 build hosts with at least 16 CPU cores and 32 GB of RAM per host. Each host is running 12 workers. These numbers are steadily increasing as new developers join the project, more projects and packages are added and additional target platforms are supported.
The MeeGo OBS regularly builds two releases per week with images for all reference hardware platforms and architectures (Figure 1). Every Friday, a preview release is built and delivered to the QA team to test major changes into the trunk and to provide a preview the weekly release.

The release engineers have the discretion to accept, reject, or queue submitted changes to ensure the usability of the upcoming release for the MeeGo user community. On the following Tuesday, the weekly release is finalized and sent to the QA team for testing. Then on the following Wednesday by 08:00 GMT, the QA report is published and the weekly release is announced by 09:00 GMT.

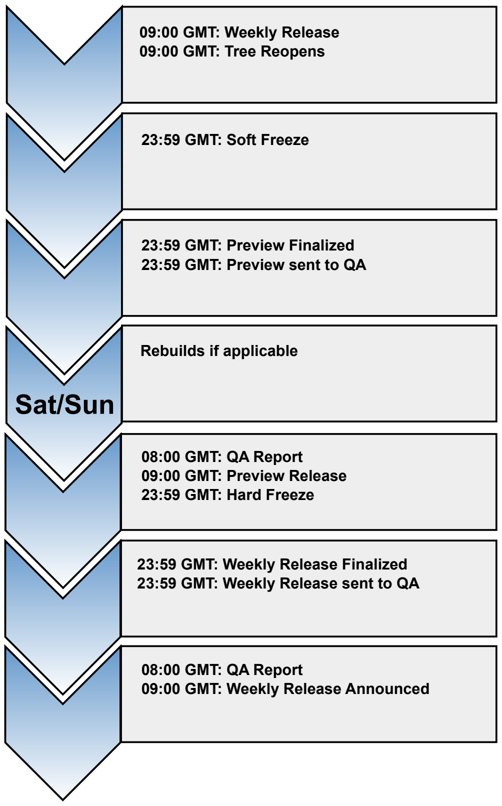

09:00 GMT: Weekly Release
09:00 GMT: Tree Reopens
23:59 GMT: Soft Freeze
23:59 GMT: Preview Finalized
23:59 GMT: Preview sent to QA
Rebuilds if applicable
Sat/Sun
08:00 GMT: QA Report
09:00 GMT: Preview Release
23:59 GMT: Hard Freeze
23:59 GMT: Weekly Release Finalized
23:59 GMT: Weekly Release sent to QA
08:00 GMT: QA Report
09:00 GMT: Weekly Release Announced

Figure 1. MeeGo Weekly Release Cycle

The MeeGo Community Build Service directly links to the MeeGo Build Service providing application developers direct read-only access to all projects and packages contained in the MeeGo Build Service. This allows application developers to directly build and test their work using the latest MeeGo branches.

Getting Started with MeeGo OBS

In the following sections we will demonstrate how to use OBS through the WebUI and the command-line client osc. To follow along you will need an account for either the MeeGo Build Service or to the MeeGo Community Build Service.

For application developers it is recommended to use the Community Build Service. To obtain an account for the Community Build Service follow the instructions on https://build.pub.meego.com/user/register_user.

For all examples in the paper we will be using the MeeGo Build Service. However, the MeeGo Community Build Service works exactly the same way.

OBS Training

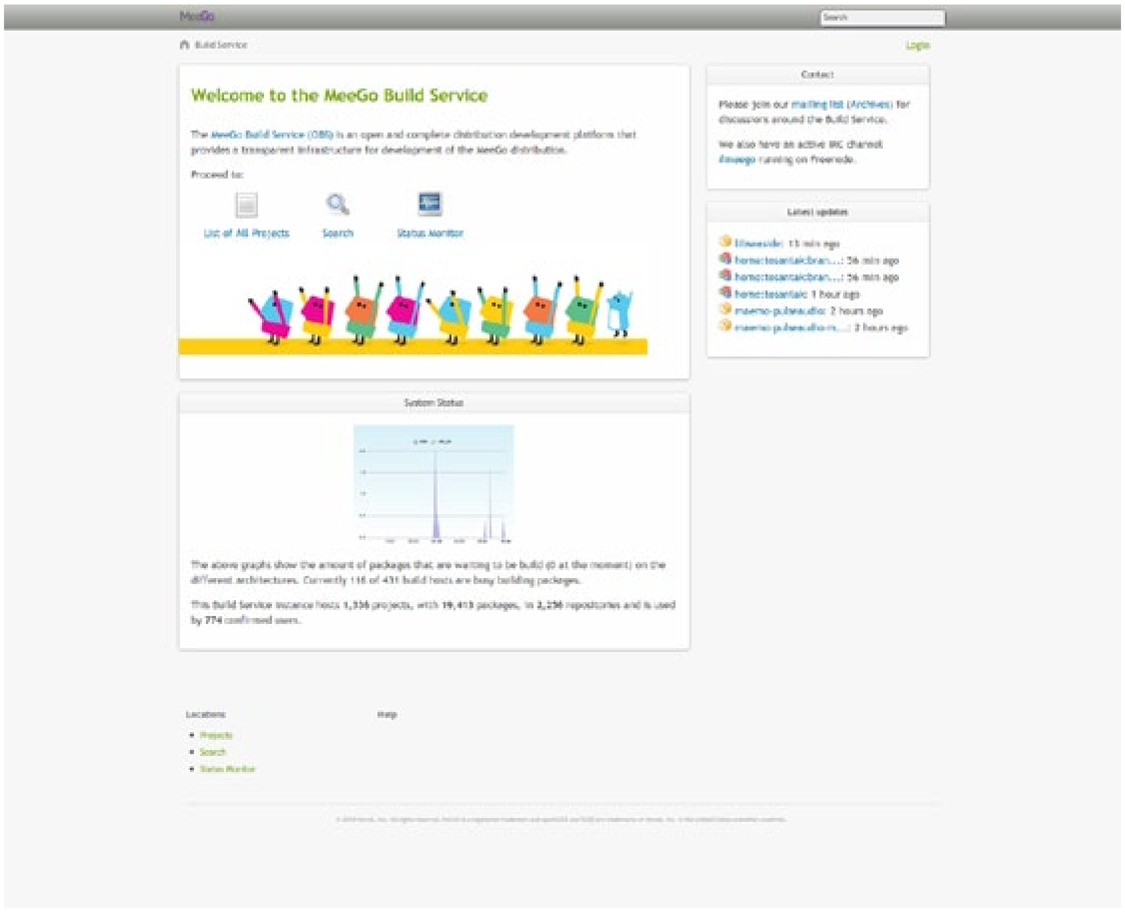
The Linux Foundation offers a hands-on training course on OBS using MeeGo as the target distribution. The course teaches how to build MeeGo for different target hardware with practical exercises. Follow this link to the course description: http://training.linuxfoundation.org/courses/meego/building-meego

Accessing OBS via WebUI

The WebUI is the simplest way to access and use OBS. There are no prerequisites other than an account for OBS and a web browser on your computer. OBS's WebUI has no special demands on the web browser. Any common browser such Chrome, Chromium or Firefox is sufficient.

Login into OBS User Account

If you have an account for the MeeGo Build Service point your browser to http://build.meego.com. If you are planning on using the MeeGo Community Build Service direct your browser to http://build.pub.meego.com. You will be presented with the OBS start page:



MeeGo

Build Service

Login

Welcome to the MeeGo Build Service

The MeeGo Build Service (OBS) is an open and complete distribution development platform that provides a transparent infrastructure for development of the MeeGo distribution.

Proceed to:

List of All Projects | Search | Status Monitor

Contact

Please join our mailing list (Archives) for discussions around the Build Service.

We also have an active IRC channel #meego running on Freenode.

Latest updates

libseaside: 15 min ago
home:tesantalcbran...: 36 min ago
home:tesantalcbran...: 36 min ago
home:tesantalc: 1 hour ago
maemo-pulseaudio: 2 hours ago
maemo-pulseaudio-m...: 2 hours ago

System Status

The above graphs show the amount of packages that are waiting to be build (0 at the moment) on the different architectures. Currently 116 of 431 build hosts are busy building packages.

This Build Service instance hosts **1,336** projects, with **19,413** packages, in **2,256** repositories and is used by **774** confirmed users.

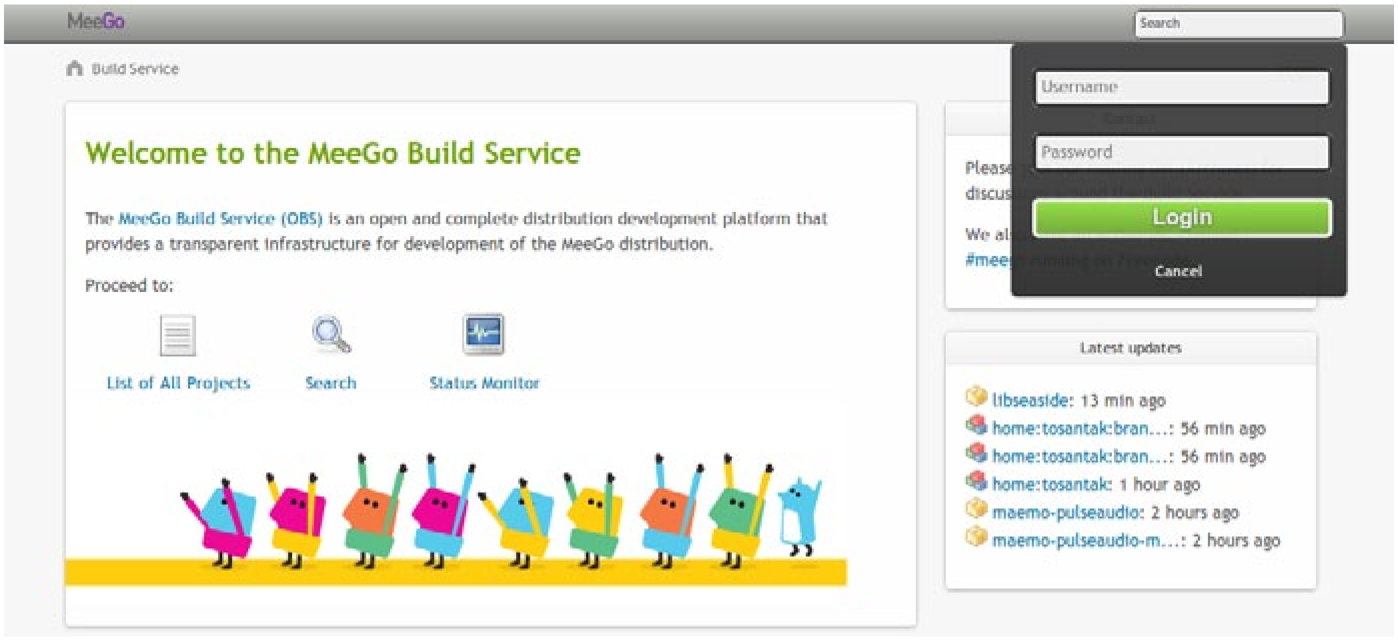
Locations:

- Projects
- Search
- Status Monitor

Help

Figure 2. MeeGo Build Service WebUI

If you click on Login in the upper right corner of the screen you will be able to enter your login credentials.



MeeGo

Search

Build Service

Welcome to the MeeGo Build Service

The MeeGo Build Service (OBS) is an open and complete distribution development platform that provides a transparent infrastructure for development of the MeeGo distribution.

Proceed to:

List of All Projects | Search | Status Monitor

Please
discus

We al
#mee

Username

Password

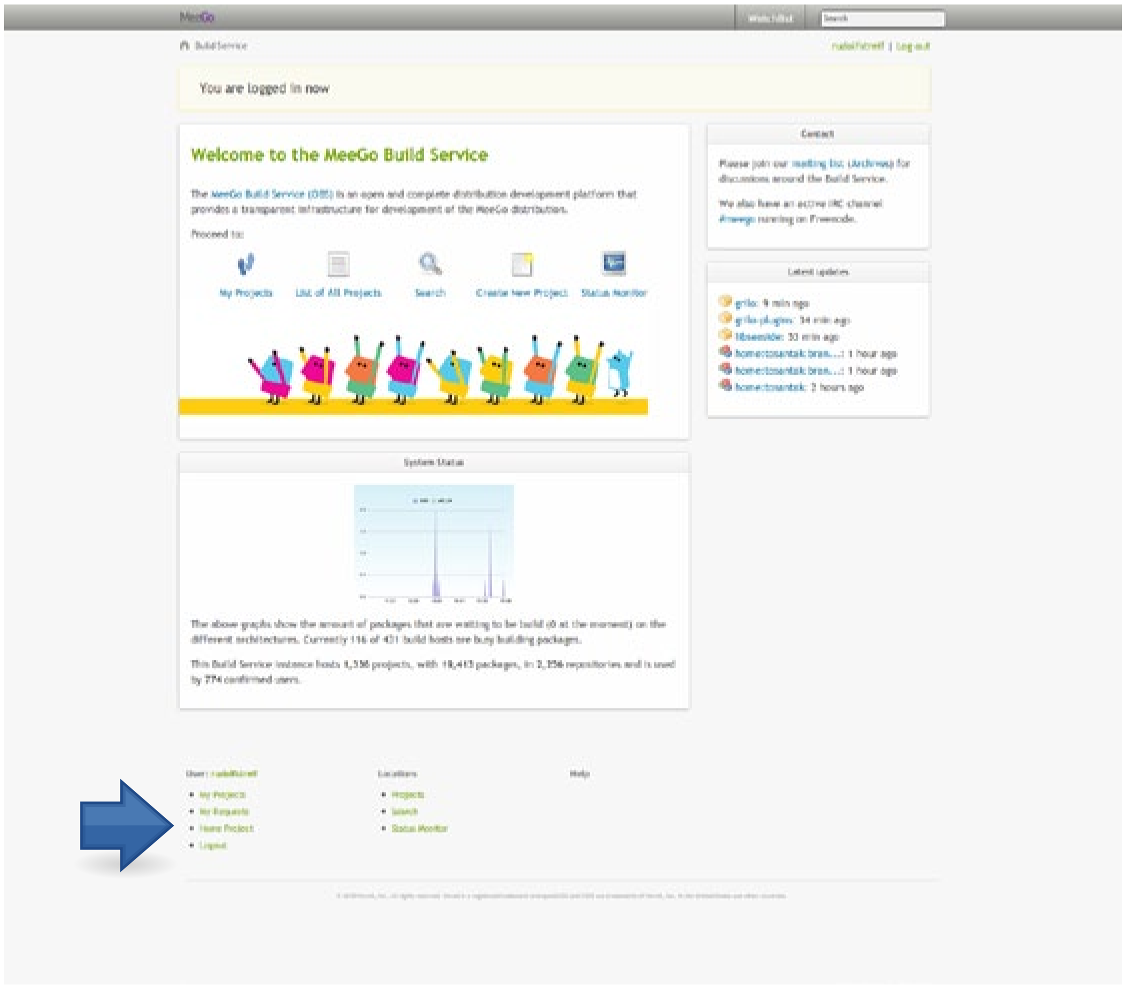
Login

Cancel

Latest updates

- libseaside: 13 min ago
- home:tosantak:bran...: 56 min ago
- home:tosantak:bran...: 56 min ago
- home:tosantak: 1 hour ago
- maemo-pulseaudio: 2 hours ago
- maemo-pulseaudio-m...: 2 hours ago

Figure 3. Login



MeeGo

Watchlist

Search

Build Service

rudolfstreif | Log out

You are logged in now

Welcome to the MeeGo Build Service

The MeeGo Build Service (OBS) is an open and complete distribution development platform that provides a transparent infrastructure for development of the MeeGo distribution.

Proceed to:

My Projects | List of All Projects | Search | Create New Project | Status Monitor

Contact

Please join our mailing list (Archives) for discussions around the Build Service.

We also have an active IRC channel #meego running on Freenode.

Latest updates

- grilo: 9 min ago
- grilo-plugins: 34 min ago
- libseaside: 30 min ago
- home:tosantak:bran...: 1 hour ago
- home:tosantak:bran...: 1 hour ago
- home:tosantak: 2 hours ago

System Status

The above graphs show the amount of packages that are waiting to be build (0 at the moment) on the different architectures. Currently 116 of 431 build hosts are busy building packages.

This Build Service instance hosts 1,538 projects, with 18,413 packages, in 2,356 repositories and is used by 774 confirmed users.

User: rudolfstreif
- My Projects
- My Requests
- Home Project
- Logout

Locations
- Projects
- Search
- Status Monitor

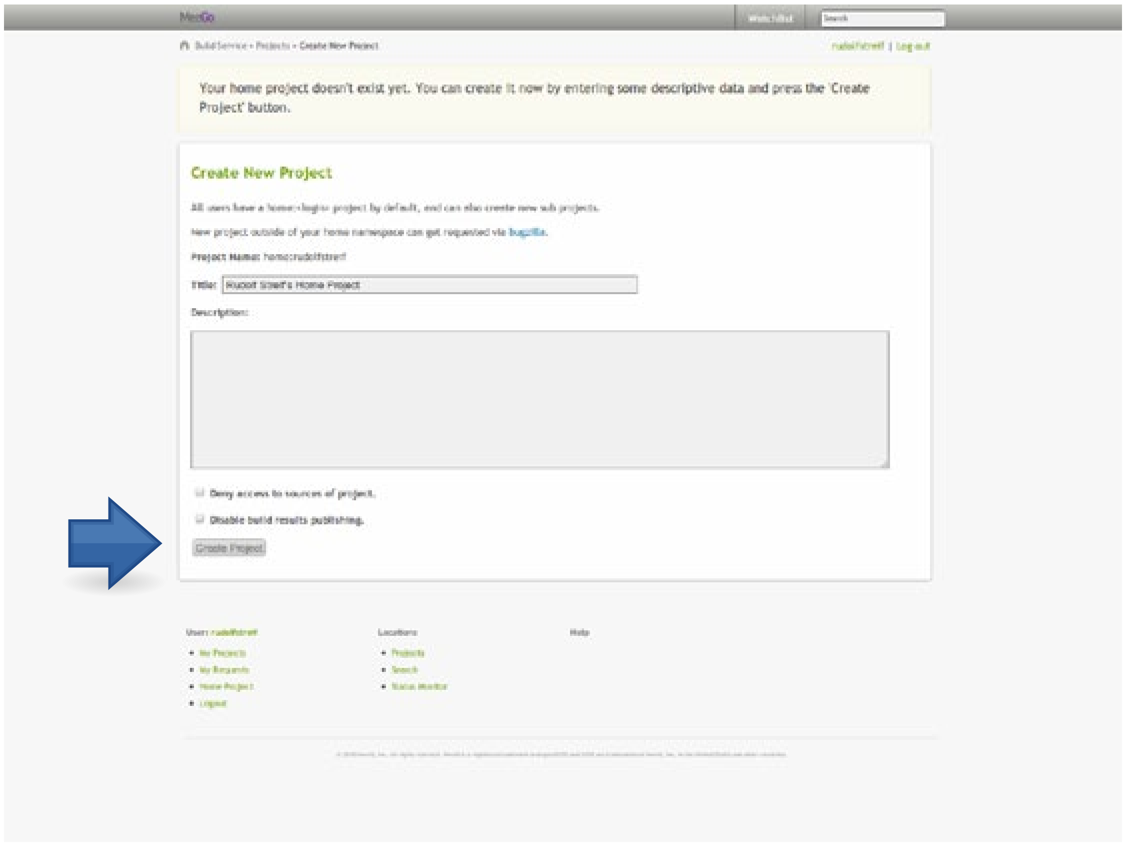
Help

[illegible]

Figure 4. After Login

Create Home Project

At the bottom of the screen you will find a menu with several items. Click on Home Project to launch the page allowing you to setup your home project. OBS assigns every user a home project as a sandbox for the user to add, modify and build packages.



Your home project doesn't exist yet. You can create it now by entering some descriptive data and press the 'Create Project' button.

Create New Project

All users have a home:<login> project by default, and can also create new sub projects.

New project outside of your home namespace can get requested via bugzilla.

Project Name: home:rudolfstreif

Title: Rudolf Streif's Home Project

Description:

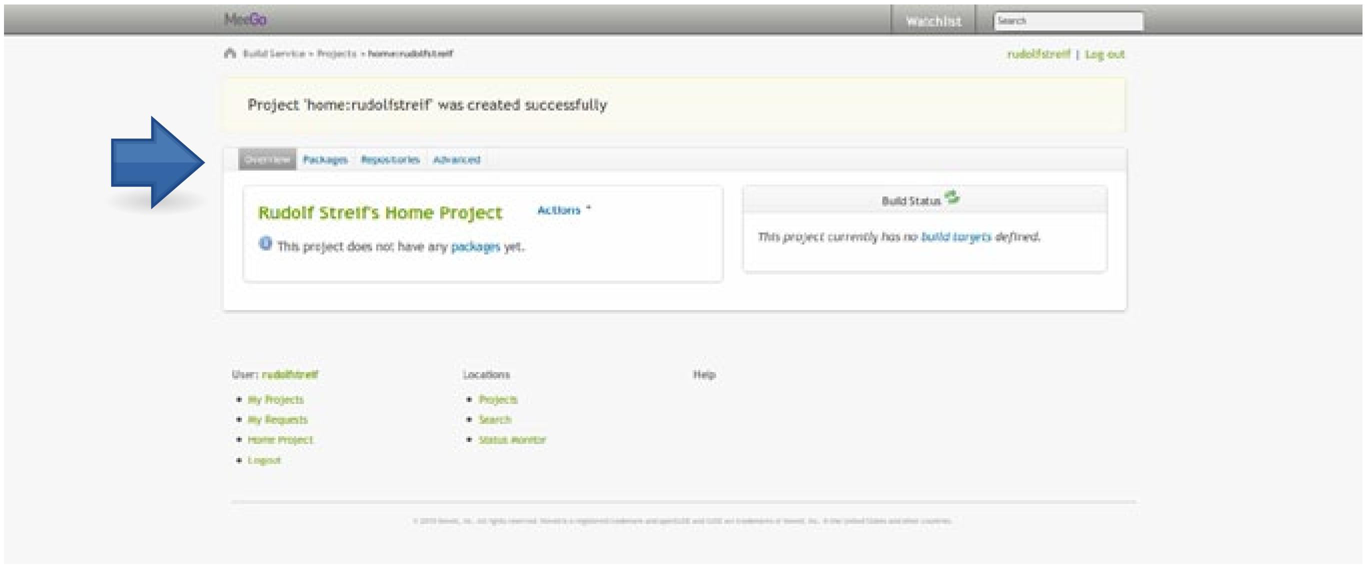
Deny access to sources of project.

Disable build results publishing.

Figure 5. Create Home Project

By default other users can access the sources in your home project unless you check *Deny access to sources of project*. OBS also publishes build results for your home project. If you do not wish others to see the build results of your home project check *Disable build results publishing*. Create your home project by clicking on *Create Project*.

Now you will see the status page of your home projects which is currently empty since it does not have any packages yet and it has no build targets defined.


Build Service › Projects › home:rudolfstreif

rudolfstreif | Log out

Project 'home:rudolfstreif' was created successfully

Overview | Packages | Repositories | Advanced

Rudolf Streif's Home Project — Actions

This project does not have any packages yet.

Build Status

This project currently has no build targets defined.

User: rudolfstreif
- My Projects
- My Requests
- Home Project
- Logout

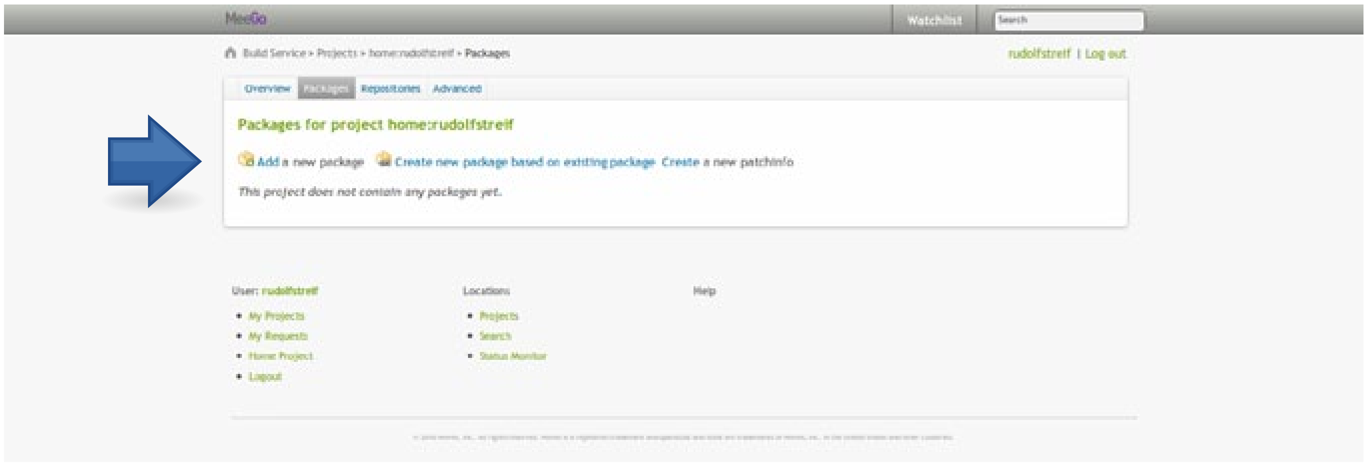
Locations
- Projects
- Search
- Status Monitor

Help

Figure 6. Empty Home Project

Adding Packages to the Home Project

Click on the *Packages* tab to display the list with packages in your home project (which is obviously empty at this point).


Build Service › Projects › home:rudolfstreif › Packages

rudolfstreif | Log out

Overview | Packages | Repositories | Advanced

Packages for project home:rudolfstreif

Add a new package   Create new package based on existing package   Create a new patchinfo

This project does not contain any packages yet.

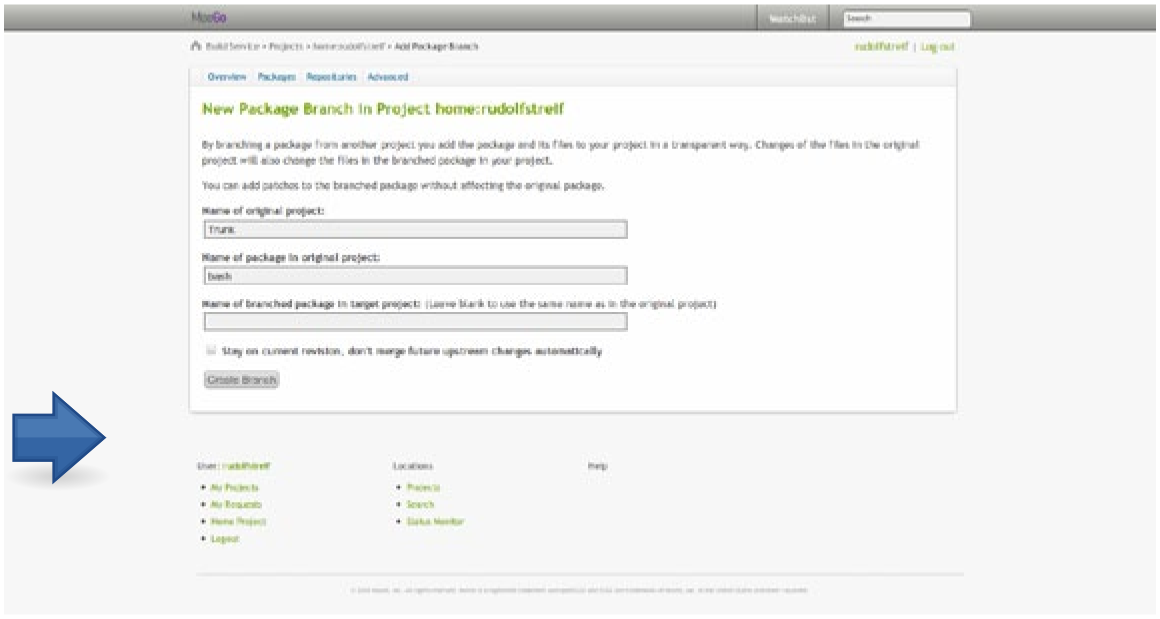
User: rudolfstreif
- My Projects
- My Requests
- Home Project
- Logout

Locations
- Projects
- Search
- Status Monitor

Help

Figure 7. Packages in Home Project

You can either add a new package by clicking on *Add a new package* or branch a package from an existing one by clicking on *Create new package based on existing package*. If you are adding a new package you will have to upload and add all source and metadata files. For this exercise we will be branching an existing package from another project. In this case all source and metadata files will be copied from the existing package into your home project. You can then make changes and build it.


Build Service › Projects › home:rudolfstreif › Add Package Branch

rudolfstreif | Log out

Overview Packages Repositories Advanced

New Package Branch in Project home:rudolfstreif

By branching a package from another project you add the package and its files to your project in a transparent way. Changes of the files in the original project will also change the files in the branched package in your project.

You can add patches to the branched package without affecting the original package.

Name of original project:
Trunk

Name of package in original project:
bash

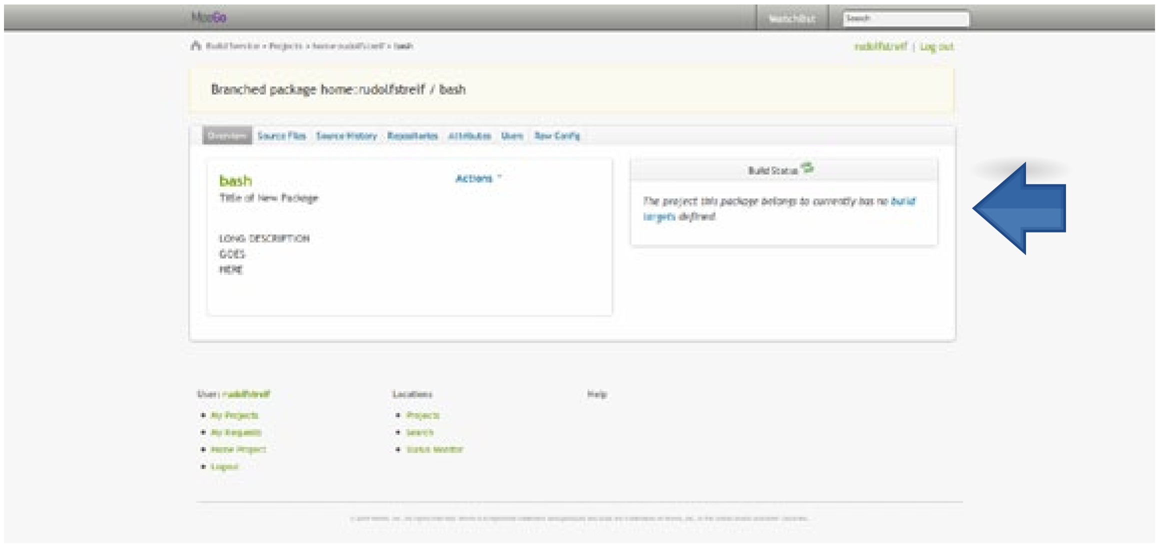
Name of branched package in target project: (Leave blank to use the same name as in the original project)

☐ Stay on current revision, don't merge future upstream changes automatically

Create Branch

Figure 8. Packages in Home Project

The package we are branching is the *bash* package (containing the sources for the Bourne-Again-Shell) located in the *Trunk* project (which is the current MeeGo development branch). If you enter *Trunk* in the field Name of *original project* the WebUI's auto-completion feature will show you a list of all available projects matching the characters you type into the field. Now start entering *bash* in the field *Name of package in original project* on the WebUI will present you with a list of all packages in the Trunk project that match your entry. In general it is a good idea to leave the field *Name of branched package in target project* empty to instruct OBS to use the same name as the original package. If you need to change the name you can change it here. Typically, OBS will automatically merge all changes made to the original package after you created your branch into your branch unless you check *Stay on current revision, don't merge future upstream changes automatically*. Finally, click on *Create Branch* to add the package to your home project.


Build Service › Projects › home:rudolfstreif › bash

rudolfstreif | Log out

Branched package home:rudolfstreif / bash

Overview Source Files Source History Repositories Attributes Users Raw Config

bash
Title of New Package

LONG DESCRIPTION
GOES
HERE

Actions

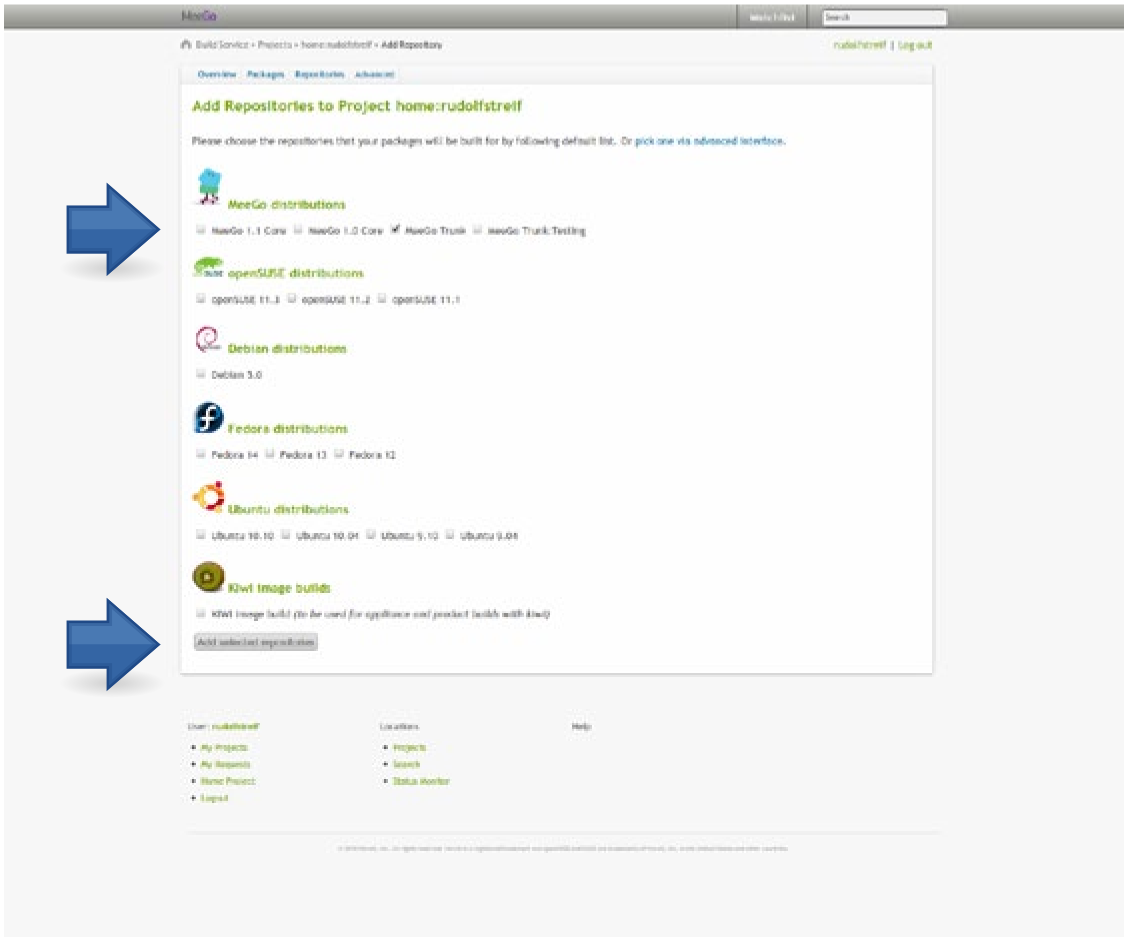
Build Status

The project this package belongs to currently has no build targets defined

Figure 9. Branched Package

Adding Build Targets

The next step is adding build targets to the project. Your project will be built against these build targets resolving any dependencies using the targets. To add build targets click on the *build targets* link in the *Build Status* box.



Build Service › Projects › home:rudolfstreif › Add Repository

Add Repositories to Project home:rudolfstreif

Please choose the repositories that your packages will be built for by following default list. Or pick one via advanced interface.

MeeGo distributions

☐ MeeGo 1.1 Core ☐ MeeGo 1.0 Core ☑ MeeGo Trunk ☐ MeeGo Trunk:Testing

openSUSE distributions

☐ openSUSE 11.3 ☐ openSUSE 11.2 ☐ openSUSE 11.1

Debian distributions

☐ Debian 5.0

Fedora distributions

☐ Fedora 14 ☐ Fedora 13 ☐ Fedora 12

Ubuntu distributions

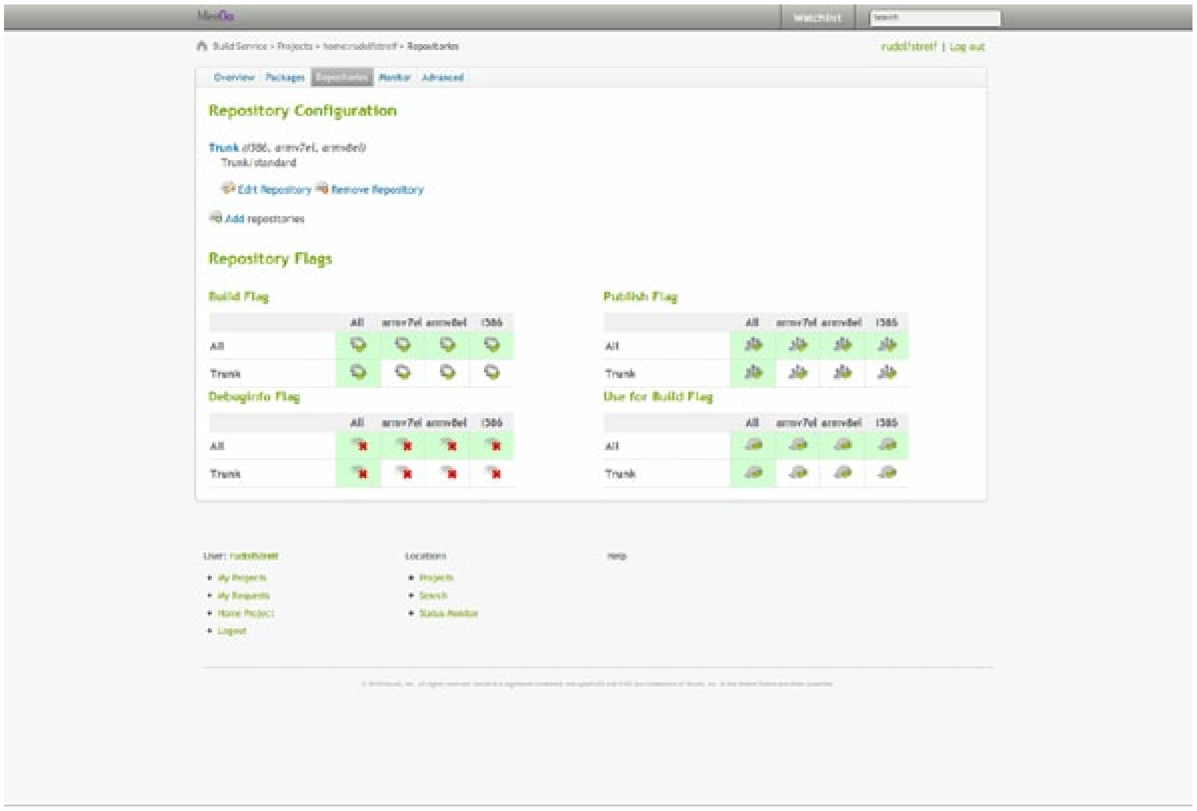
☐ Ubuntu 10.10 ☐ Ubuntu 10.04 ☐ Ubuntu 9.10 ☐ Ubuntu 9.04

Kiwi image builds

☐ KIWI image build *(to be used for appliance and product builds with kiwi)*

Figure 10. Add Repositories

The easiest way of adding build targets is to choose one or more of the pre-configured repositories. The MeeGo Build Service does not only allow you to build against the MeeGo repositories but also against other Linux distributions such as openSUSE, Debian, Fedora and Ubuntu. For this exercise we want to build for the *MeeGo Trunk*. Clicking on *Add selected repositories* takes you to the repository configuration page.



Repository Configuration

Trunk (i586, armv7el, armv8el)
Trunk/standard

Edit Repository Remove Repository

Add repositories

Repository Flags

Build Flag

	All	armv7el	armv8el	i586
All				
Trunk				

Debuginfo Flag

	All	armv7el	armv8el	i586
All				
Trunk				

Publish Flag

	All	armv7el	armv8el	i586
All				
Trunk				

Use for Build Flag

	All	armv7el	armv8el	i586
All				
Trunk				

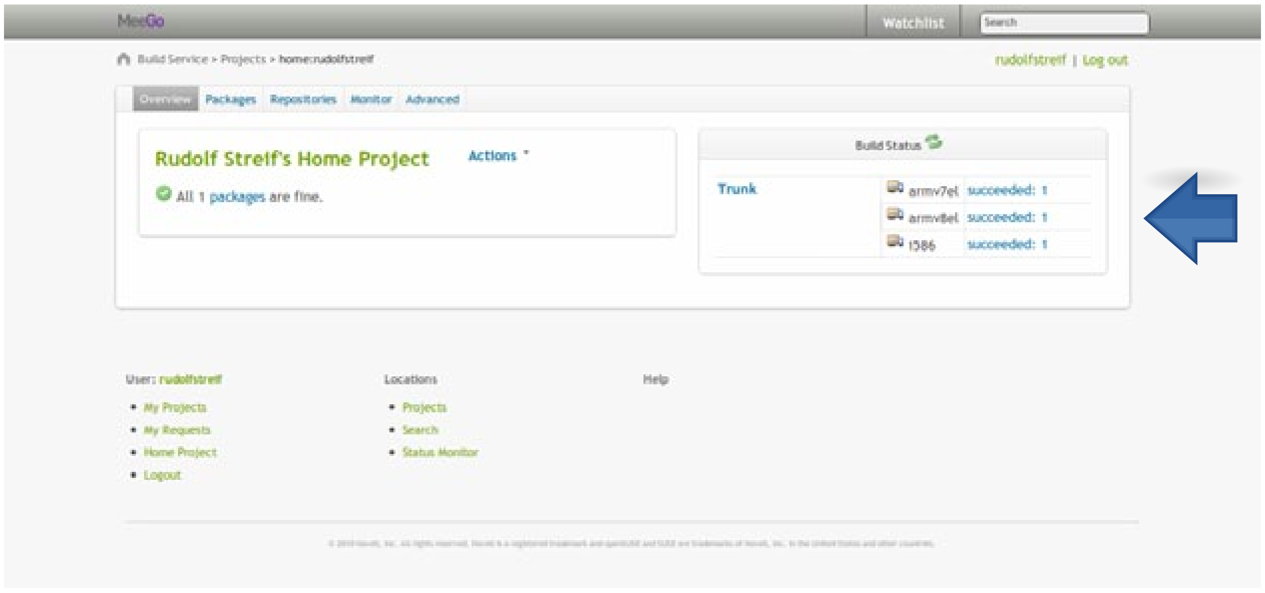
Figure 11. Repository Configuration

Repository configuration lets you fine-tune the build for your project:

- **Build Flag** - Select for which architecture you would like to build your project.
- **Debuginfo Flag** - Select for which architecture you would like to build the debug information.
- **Publish Flag** - Select for which architectures you would like to publish the build results.
- **Use for Build Flag** - Select for which architecture you would like to use prebuilt binaries.

Building

The moment you selected the repositories you would like to build your home project against you already submitted the packages in your home project to OBS for building. If you click on the Overview tab you will return to the status page for your home project.



Rudolf Streif's Home Project

All 1 packages are fine.

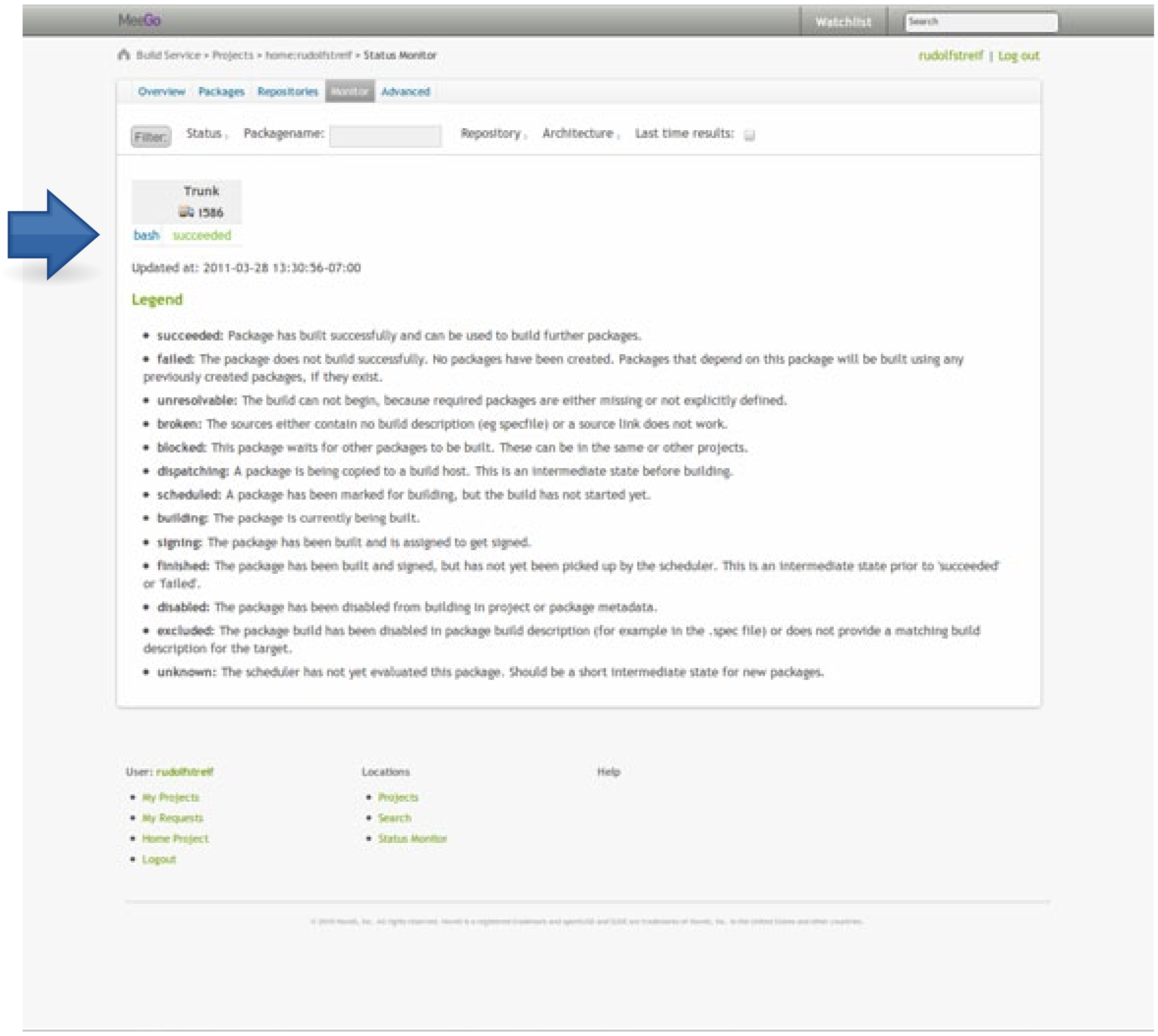
Build Status

Trunk	armv7el	succeeded: 1
	armv8el	succeeded: 1
	i586	succeeded: 1

Figure 12. Project Overview

Verifying Build Results

Bash is a rather small package and dependent on the capacity of the build farm you may already see in the *Build Status box* that OBS has built your home project for all architectures. Click on one of the status links for any of the architectures such as *succeeded* (or other such as *scheduled*, *building*, etc if the build is not yet complete) to go to the build status page for that architecture.



Updated at: 2011-03-28 13:30:56-07:00

Legend

- **succeeded:** Package has built successfully and can be used to build further packages.
- **failed:** The package does not build successfully. No packages have been created. Packages that depend on this package will be built using any previously created packages, if they exist.
- **unresolvable:** The build can not begin, because required packages are either missing or not explicitly defined.
- **broken:** The sources either contain no build description (eg specfile) or a source link does not work.
- **blocked:** This package waits for other packages to be built. These can be in the same or other projects.
- **dispatching:** A package is being copied to a build host. This is an intermediate state before building.
- **scheduled:** A package has been marked for building, but the build has not started yet.
- **building:** The package is currently being built.
- **signing:** The package has been built and is assigned to get signed.
- **finished:** The package has been built and signed, but has not yet been picked up by the scheduler. This is an intermediate state prior to 'succeeded' or 'failed'.
- **disabled:** The package has been disabled from building in project or package metadata.
- **excluded:** The package build has been disabled in package build description (for example in the .spec file) or does not provide a matching build description for the target.
- **unknown:** The scheduler has not yet evaluated this package. Should be a short intermediate state for new packages.

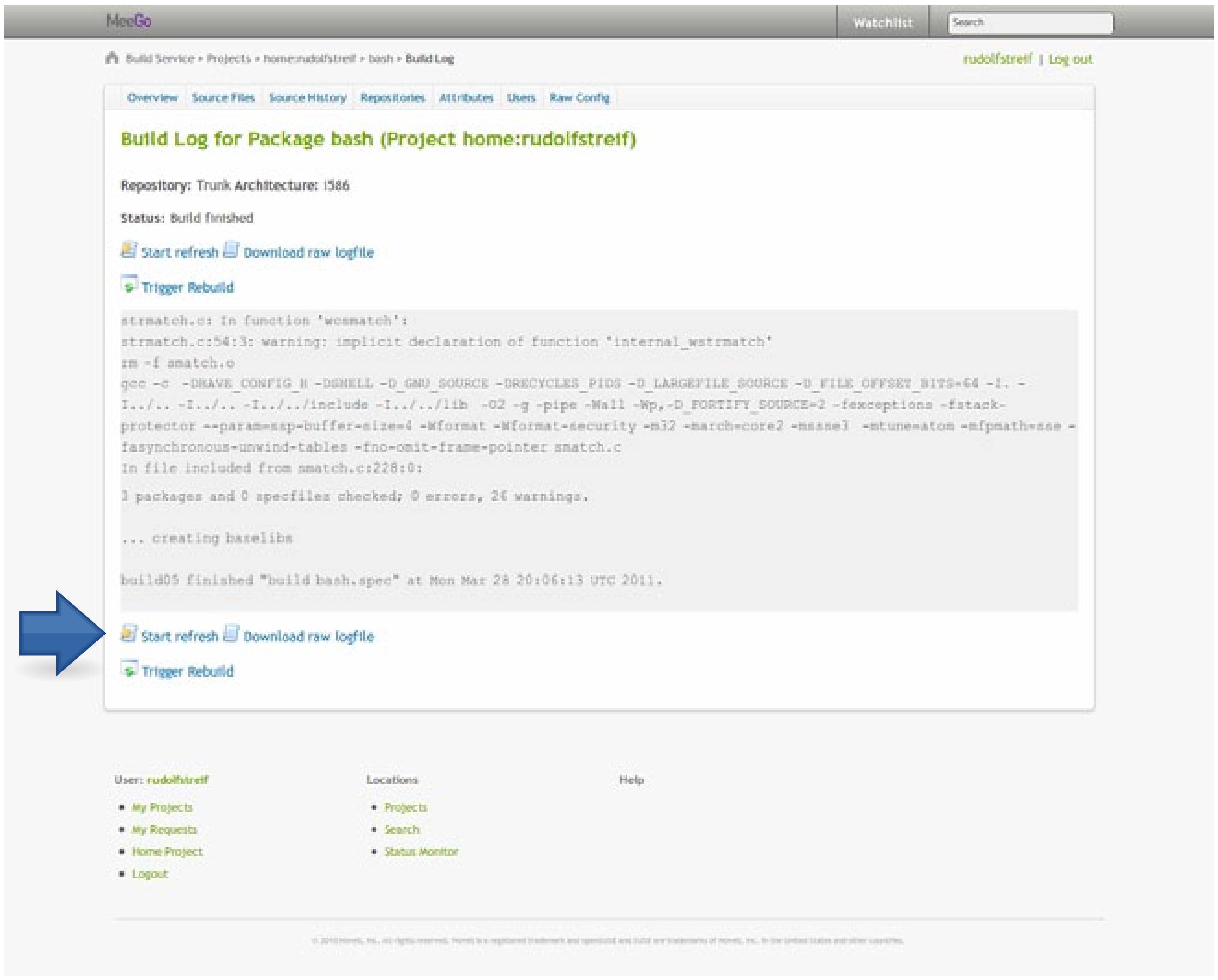
Figure 13. Build Status

The build status page gives you an overview over the current build status of all packages in your home project for that architecture (alternatively you could click on the Monitor tab and it will show you the status for all architectures). Currently, OBS distinguishes 13 different states:

- **succeeded** - Package has built successfully and can be used to build further packages.
- **failed** - he package does not build successfully. No packages have been created. Packages that depend on this package will be built using any previously created packages, if they exist.
- **unresolvable** - The build can not begin, because required packages are either missing or not explicitly defined.
- **broken** - The sources either contain no build description (eg specfile) or a source link does not work.
- **blocked** - This package waits for other packages to be built. These can be in the same or other projects.
- **dispatching** - A package is being copied to a build host. This is an intermediate state before building.
- **scheduled** - A package has been marked for building, but the build has not started yet.

- **building** - The package is currently being built.
- **signing** - The package has been built and is assigned to get signed.
- **finished** - The package has been built and signed, but has not yet been picked up by the scheduler. This is an intermediate state prior to ‘succeeded’ or ‘failed’.
- **disabled** - The package has been disabled from building in project or package metadata.
- **excluded** - The package build has been disabled in package build description (for example in the .spec file) or does not provide a matching build description for the target.
- **unknown** - The scheduler has not yet evaluated this package. Should be a short intermediate state for new packages.

Clicking on the status label next to the package names directs you to the detailed build output.



Build Log for Package bash (Project home:rudolfstreif)

Repository: Trunk Architecture: i586

Status: Build finished

Start refresh Download raw logfile

Trigger Rebuild

```
strmatch.c: In function 'wcsmatch':
strmatch.c:54:3: warning: implicit declaration of function 'internal_wstrmatch'
rm -f smatch.o
gcc -c  -DHAVE_CONFIG_H -DSHELL -D_GNU_SOURCE -DRECYCLES_PIDS -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -I. -
I../.. -I../.. -I../../include -I../../lib  -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-
protector --param=ssp-buffer-size=4 -Wformat -Wformat-security -m32 -march=core2 -mssse3  -mtune=atom -mfpmath=sse -
fasynchronous-unwind-tables -fno-omit-frame-pointer smatch.c
In file included from smatch.c:228:0:
3 packages and 0 specfiles checked; 0 errors, 26 warnings.

... creating baselibs

build05 finished "build bash.spec" at Mon Mar 28 20:06:13 UTC 2011.
```

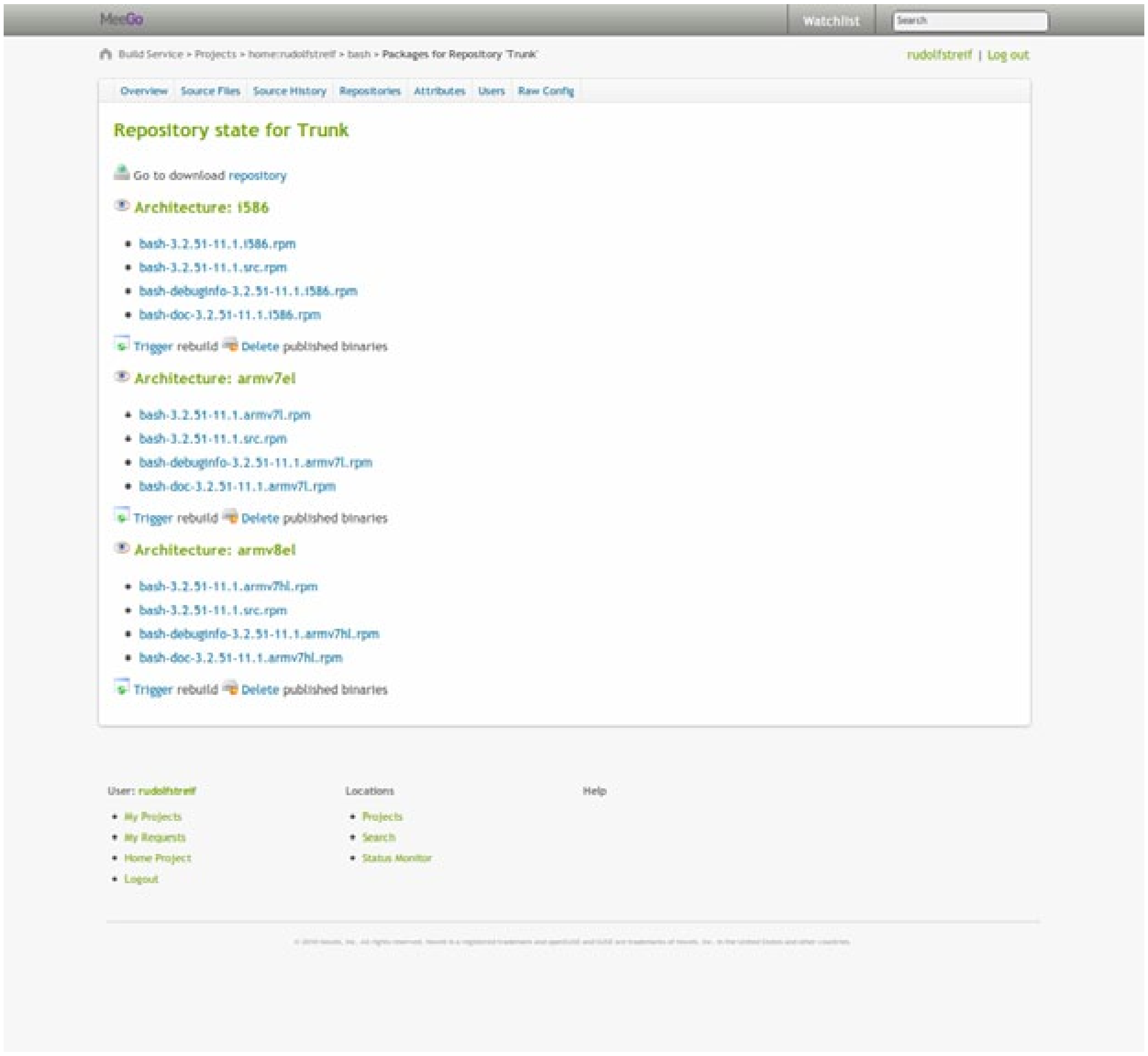
Start refresh Download raw logfile

Trigger Rebuild

Figure 14. Build Output

The build output is essentially the log that you would be seeing if you were running the build, typically with *make*, on your own system. If you like to view or download the raw log file to your system simply click on *Download raw logfile*. Click on *Trigger rebuild* to instruct OBS to build your package again. If you go back to the status page and stay on it you can follow the build process.

Click on the *Overview* tab to return to the status page for the bash project and then click on the *Trunk* label in the *Build Status* box. OBS will direct you to the repository state page for the binary packages in your project.



Build Service › Projects › home:rudolfstreif › bash › Packages for Repository 'Trunk'

rudolfstreif | Log out

Overview Source Files Source History Repositories Attributes Users Raw Config

Repository state for Trunk

Go to download repository

Architecture: i586

- bash-3.2.51-11.1.i586.rpm
- bash-3.2.51-11.1.src.rpm
- bash-debuginfo-3.2.51-11.1.i586.rpm
- bash-doc-3.2.51-11.1.i586.rpm

Trigger rebuild Delete published binaries

Architecture: armv7el

- bash-3.2.51-11.1.armv7l.rpm
- bash-3.2.51-11.1.src.rpm
- bash-debuginfo-3.2.51-11.1.armv7l.rpm
- bash-doc-3.2.51-11.1.armv7l.rpm

Trigger rebuild Delete published binaries

Architecture: armv8el

- bash-3.2.51-11.1.armv7hl.rpm
- bash-3.2.51-11.1.src.rpm
- bash-debuginfo-3.2.51-11.1.armv7hl.rpm
- bash-doc-3.2.51-11.1.armv7hl.rpm

Trigger rebuild Delete published binaries

User: rudolfstreif
- My Projects
- My Requests
- Home Project
- Logout

Locations
- Projects
- Search
- Status Monitor

Help

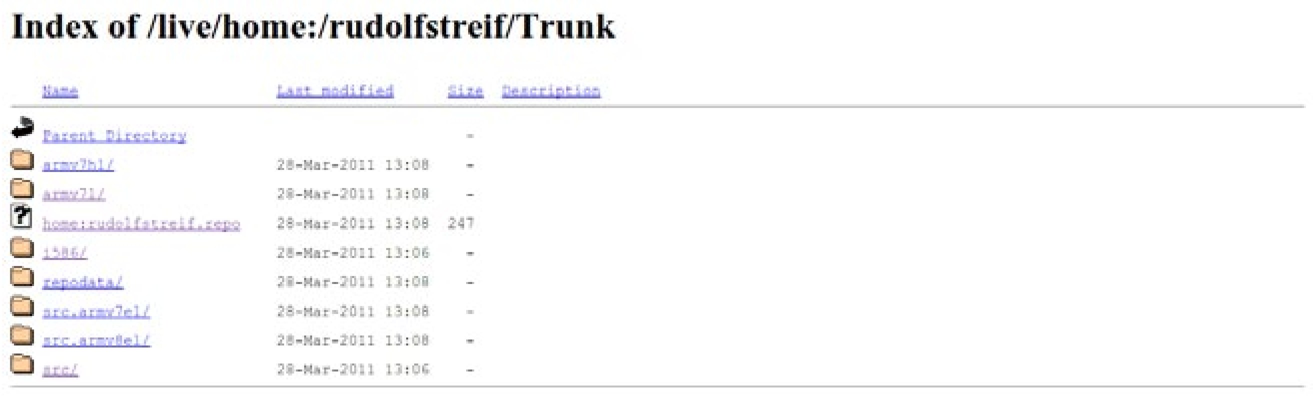
[illegible]

Figure 15. Repository State

This page allows you to manage the binary packages after they have been built by OBS. Clicking on the respective package names will show you detail information about the packages. For binary packages that includes the runtime dependencies.

Downloading Packages

Finally, you will probably want to download the binary packages to your system to install and run them. Click on Go to *download repository* to open a download page through which you can download your packages using http.



Index of /live/home:/rudolfstreif/Trunk

```
    Name                       Last modified      Size  Description
-----------------------------------------------------------------
    Parent Directory                               -
    armv7hl/                   28-Mar-2011 13:08   -
    armv7l/                    28-Mar-2011 13:08   -
    home:rudolfstreif.repo     28-Mar-2011 13:08  247
    i586/                      28-Mar-2011 13:06   -
    repodata/                  28-Mar-2011 13:08   -
    src.armv7el/               28-Mar-2011 13:08   -
    src.armv8el/               28-Mar-2011 13:08   -
    src/                       28-Mar-2011 13:06   -
-----------------------------------------------------------------
```

Figure 16. Repository Download

Summary

This concludes the section on working with OBS through the WebGUI. To summarize the steps we went through:

- Login into OBS account.
- Setup user's home project.
- Branch a package from MeeGo Trunk.
- Build the package against MeeGo Trunk for three different architectures.
- Verify the build results.
- Download the built packages to the local system.

The OBS WebUI offers more functionality such as a status monitor to view the current status of system including all the Workers, a page to view all projects, a search function, etc. These functions are self-explanatory and you are invited to explore them on your own.

Working with the OBS Command-line Interface OSC

The OBS command-line client osc is a set of Python scripts that will run on any Linux distribution. The osc command-line client interfaces directly with the OBS API and uses Subversion-like commands. Developers familiar with the Subversion command-line client *svn* will find osc rather similar but even novices to any source control system will find osc straightforward to use.

Users can extend osc via plugins to perform additional tasks when certain commands are executed or even add new commands to the tool.

MeeGo uses a slightly modified version of the openSUSE osc which has been extended via plugins allowing to add additional tags etc. MeeGo provides repositories for MeeGo osc and other MeeGo developer tools for all major distributions.

Installing OSC

Obtaining the tools and keeping them up to date is simplified by adding the repository to your distribution's package manager repository list. To find the proper URL for your distribution go to http://repo.meego.com/MeeGo/tools/repos/.

We are using Ubuntu 10.04 as an example here. Other distributions work similar except that the package manager may be Yum/Zypper instead of Synaptic.

1. Add package source
   Add the following line to the end of /etc/apt/sources.list
   ```
   deb http://repo.meego.com/MeeGo/tools/repos/ubuntu/10.04 /
   ```
2. Update repository list
   ```
   sudo apt-get update
   ```
   You should see the following error "GPG error: http://repo.meego.com Release: The following signatures could not be verified because the public key is not available: NO_PUBKEY 0BC7BEC479FC1F8A."
   Add the repository public key with:
   ```
   gpg –keyserver subkeys.pgp.net –recv 0BC7BEC479FC1F8A
   gpg –export –armor 0BC7BEC479FC1F8A | sudo apt-key add -
   ```
   If your system is located behind a corporate firewall access to the GPG keyserver may be blocked. Download the public key from the repository and install it manually:
   ```
   wget http://repo.meego.com/MeeGo/tools/repos/ubuntu/10.04/Release.key
   ```
   Open *System* > *Administrator* > *Software Sources* and import the key under the *Authentication tab*. Now, run again:
   ```
   sudo apt-get update
   ```
3. Install osc
   ```
   sudo apt-get install osc
   ```
4. Verify installation
   ```
   osc
   ```
   If osc installed correctly you should see the following output (truncated):
   ```
   Usage: osc [GLOBALOPTS] SUBCOMMAND [OPTS] [ARGS...]
   or: osc help SUBCOMMAND
   ```

Setting up OSC

Osc requires you to specify the URL of the API of the OBS server you would like to use for your work. This is done via the `-A<apiurl> or --apiurl=<apiurl>` parameter. To simplify use it is recommended to use an alias:
For the MeeGo Core OBS: `alias meosc='osc -Ahttps://api.meego.com'`
For the MeeGo Community OBS: `alias mcosc='osc -Ahttps://api.pub.meego.com'`
It is advisable to add the alias to your ~/.bashrc or to /etc/profile. For all of the following examples we will be using the MeeGo Core OBS and the meosc alias. If you are using the MeeGo Community OBS please substitute mcosc or meosc.

Running OSC for the First Time

If you now type meosc it will ask you for your user account credentials for the target OBS:
```
Your user account / password are not configured yet.
You will be asked for them below, and they will be stored in
/home/<username>/.oscrc for future use.
```

```
Creating osc configuration file /home/<username>/.oscrc ...
Username: <login>
Password: <password>
done
*** certificate verify failed at depth 0
Subject:  /O=build.linux.com/OU=Domain Control Validated/CN=build.linux.com
Issuer:   /C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com, ¬    Inc./OU=http://
certificates.godaddy.com/repository/CN=Go Daddy ¬
     Secure Certification Authority/serialNumber=07969287
Valid:  May 14 21:46:46 2010 GMT - Apr 26 21:10:19 2012 GMT
Fingerprint(MD5):   C13D91AB12008D2F9FD901A8FADDFC75
Fingerprint(SHA1):  144018EC455C20F395F18879DFD75E5B500B84F0
Reason: unable to get local issuer certificate
Reason: certificate not trusted
Reason: unable to verify the first certificate

The server certificate failed verification

Would you like to
0 - quit (default)
1 - continue anyways
2 - trust the server certificate permanently
9 - review the server certificate

Enter choice [0129]: 2

Usage: osc [GLOBALOPTS] SUBCOMMAND [OPTS] [ARGS...]
or: osc help SUBCOMMAND
...
```

Since we did not specify a command osc will just print out the standard help information. Osc has now created .oscrc in your home directory. It uses this file for its configuration settings unless you explicitly specify a different file with the `-c <file>` or `--config=<file>` parameter. You can tune osc's behavior by adjusting the parameters in .oscrc. We will be addressing some of the parameters in the context of the following sections.
After this initial setup your ~/.oscrc file will also store the URL to the OBS server. Hence, for all the following examples you may simply use osc instead of *meosc* or *mcosc*.

Basic OSC Commands

There are a couple of osc commands that you can use without being in the context of a project or working copy:

- `osc help` - print the overview help information.
- `osc help <command>` - print specific help information on <command>, for instance 'osc help checkout' prints information on the checkout command.
- `osc list` - list all projects of that OBS (alternatively you can use ls or ll instead of list).

- `osc list <project>` - list all packages in a project, for instance 'osc list home:<username>' list all packages in the home project of <username>.
- `osc list <project> <package>` – list source files of <package> of <project>.
- `osc -b <project>` - list all binaries of <project>.
- `osc -b <project> -a <architecture>` – list binaries of <project> for <architecture>.
- `osc my <type>` - show projects (<type> = projects), packages (<type> = packages), or requests (<type> = requests or submitrequests) involving yourself.
- `osc search <search term>` - search projects and packages that contain <search term> in their name, title or description.
- `osc config <section> <option>` - get configuration option (if no configuration file is explicitly specified the option in ~/.oscrc will be returned). <section> is either general or a URL delimiting a section for a specific OBS instance. For example 'osc config general apiurl' retrieves the current setting for apiurl in ~/.oscrc or 'osc config general su-wrapper' retrieves the wrapper called by osc to execute commands as root. The command 'osc config https://api.meego.com trusted_prj' returns the setting for the trusted projects option for the MeeGo Core OBS.
- `osc config <section> <option> <value>` - set configuration option (if a configuration file is specified explicitly the corresponding option in the section of that file will be changed otherwise the ~/.oscrc will be modified). The 'osc config' command is useful to set and get configuration options on the fly when embedding osc commands in shell scripts.

Checkout and Build a Package Locally

With OBS and osc you can easily work with projects and packages on your local development system. Since the OBS repositories contain the compilers and all other tools of the toolchain there is no need for you to install any tools, other than osc of course, on your system.

First you would want to create a workspace on your system for your work with obs. A workspace is simply a directory locates anywhere on your system. We will use the home directory:

```
mkdir ~/obs
cd ~/obs
```

If you followed along the previous section on using the OBS WebUI then your home project already contains the *bash* package. You can simply checkout your entire home project with:

```
osc co home:<your login>
```

Alternatively, you can just checkout the *bash* package from your home project:

```
osc co home:<your login> bash
```

In case you did not follow the section on using the OBS WebUI you can easily branch the bash package from the Trunk project into your home project:

```
osc bco Trunk bash home:<your login>
```

Osc will launch an editor asking you to enter a comment for creating the branch.

While osc is checking out the source files into your workspace it will print messages similar to A home:<your login>/bash

```
A    home:<your login>/bash/Makefile
A    home:<your login>/bash/bash-2.02-security.patch
A    home:<your login>/bash/bash-2.03-paths.patch
A    home:<your login>/bash/bash-2.03-profile.patch
…
A    home:<your login>/bash/dot-bashrc
At revision a3c10bd652f609d9801355f1fa9867e2.
```

The revision number will may be different.

Now it is time to build the package. Change to the package directory

```
cd ~/obs/home:<your login>/bash
```

Before we actually start building you may want to adjust where osc sets up the build environment on your system. The target directory can be specified through the *build-root* variable in ~/.oscrc. You can either edit the file with an editor or use *osc config* to set the variable:

```
osc config general build-root "/home/<your home>/obs/build-root/%(project)
s-%(arch)s"
```

You will have to specify and absolute path. Shell substitutions such as ~ for your home directory will not work. The %(project)s-%(arch)s directive with its placeholders will tell osc to create a build environment separate for project and architecture. This is useful and keeps projects and architectures separate since the build environment is actually a chroot that you can use for local debugging. You may also add the placeholder %(package)s to the *build-root* path to additionally separate by package name.

Packages are always built against a *repository* and for an *architecture*:

```
osc build Trunk i586
```

builds the bash package against the *Trunk* repository for i586 architecture. Osc will now start downloading all dependencies necessary to build the package from the OBS server, create a *chroot* environment and build the package. Dependent on the number of dependencies and the bandwidth of the connection to the OBS server this may take a while. However, downloading dependencies is only executed on the initial build and if dependencies have changed.
OBS can optionally sign binary packages and by default osc will verify the signatures. However, if the OBS you are using is not signing the packages osc will abort building with an error message:

```
BuildService API error: can't verify packages due to lack of GPG keys
```

In that case run *osc build* with the *--no-verify* option:

```
osc build --no-verify Trunk i586
```

This time osc will find the binary packages in its cache and will not download them again. Since osc uses a chroot environment for building you will be asked to enter your password to execute chroot as root.

You can now use

```
osc chroot Trunk i586
```

to chroot into the build environment. Within the chroot build environment you can execute, debug etc. your package. Type exit in the chroot environment to exit the chroot.
The example has built the bash package for i586 architecture. Most likely your development system is an i86 system and hence the architecture of your development system matches the architecture of the build target. However, osc equally simply lets you build for a different target architecture:

```
osc build --no-verify Trunk armv7el
```

Since this is a new architecture osc will first download all dependencies for that architecture, then start set up the build environment and start building the bash package for armv7el. After building has completed, executing

```
osc chroot Trunk armv7el
```

will take you to the chroot environment where you can execute, debug etc. your package in an emulated environment.

Preparing, Building and Checking-in a Source Package

The typical job of a packager is to obtain sources for a project from a *maintainer/developer* and package it into a format for building it with OBS. OBS uses source RPMs for importing packages. The packager would typically have to create a source RPM with all the necessary source files including a *.spec* file for RPM.

The intrinsics of RPM packaging are beyond the scope of this paper hence we will be using the pre-packaged source from the Nano Editor project (http://www.nano-editor.org). Simply download the file *nano-2.2.6.tar.gz* from the Nano website and place it into your ~/obs directory (the version of Nano may vary; any version is suitable).

First we need to create a new package under version control in the home project:

```
cd ~/obs/home:<your login>
osc mkpac nano
```

Now we will need to copy the source archive to the package directory and extract the .spec file (the source archive nano-2.2.6.tar.gz already contains a .spec file):

```
cd ~/obs/home:<your login>/nano
cp ~/obs/nano-2.2.6.tar.gz .
tar xvfz nano-2.2.6.tar.gz nano-2.2.6/nano.spec
cp nano-2.2.6/nano.spec .
rm -r nano-2.2.6
```

Before we are going to check in the package to the OBS server we build it locally against the Trunk repository for i586 architecture to make sure that it builds correctly:

```
osc build --local-package --no-verify Trunk i586
```

You may see an error message from RPM build saying:

```
RPM build errors:
Installed (but unpackaged) file(s) found:
/usr/share/info/dir
```

To correct it simply add *rm -rf %buildroot/usr/share/info/dir* to the end of the *%install* section of *nano.spec* and execute:

```
osc wipebinaries --all home:<your login> nano
osc build --local-package --no-verify Trunk i586
```

The first command removes the binaries built during the previous attempt and the second rebuilds the package. Now the Nano package should locally build without any problems. You may use *osc chroot Trunk i586* to browse the build environment.

Since the package now builds correctly locally it is time to check it in to the OBS server (executed from within the ~/obs/home:<your login>/nano directory):

```
osc st
```

will show the status of the files in the local working copy:

```
?     nano-2.2.6.tar.gz
?     nano.spec
```

The ? indicates that the items are not under version control. The status indicators are:

```
<blank>    No modifications (typically suppressed unless you specify -v)
A          Added
C          Conflicted (changes to local copy conflict with file on the
           server)
D          Deleted
M          Modified
?          Item is not under version control
!          Item is missing (removed by non-osc command) or incomplete
```

To add them and mark them for local deletion execute:

```
osc addremove
```

To check the package into the server execute:

```
osc ci -m"Initial import of Nano package"
```

The *-m* parameter specifies the check-in message. If you do not specify it osc will launch an editor for you to enter the message. Using check-in message is strongly encouraged.

Osc will now create the package on the OBS server, upload the project files and the meta data. Under some circumstances you may see an error message: *Package 'nano' does not exist*. In that case the package has not been created and the meta data was not uploaded. You can correct this by creating the package meta data manually:

```
    osc meta pkg home:<your login> nano -e
```

Checking in a package or changes to a package to the OBS server will automatically trigger a build of the package on the server (as well as all packages depending on this package). You can verify the build progress on the server by either using the WebUI or executing *osc bl <repository> <arch>* from the package directory:

```
    osc bl Trunk i586
```

From anywhere else on your system you can use osc *rbl <project> <package> <repository> <arch>*:

```
    osc rbl home:<your login> nano Trunk i586
```

To check the build status use osc r from within the working directory. To verify the build status of all packages within a project execute osc pr <project> for instance for your home project:

```
    osc pr home:<your login>
```

Summary

This concludes the section on working with OBS using the command-line client osc. To summarize the steps we went through:

- Obtaining and installing osc.
- Setting up osc and running it for the first time.
- Basic osc commands.
- Checking out and building a package locally.
- Preparing, building and checking-in a source package.

These cover all the essential osc commands for a developer or packager to work with projects and packages inside their own home project. Eventually a developer or package will be part of a team requiring to submit changes to other projects. For this purpose osc offers a set of commands for creating, viewing, modifying etc. requests to other users. Also if developers or packagers are maintainers of packages they will receive request to integrate changes from other team members into the projects they are responsible for.

Conclusion

The Open Build Service (OBS) is a powerful and complete distribution development platform. It provides the necessary infrastructure to easily create, package and release open source software for different Linux distributions including MeeGo. Users, software developers and distributors can benefit from OBS's capabilities likewise. The MeeGo ecosystem relies on OBS as its backbone for building the releases for the different MeeGo verticals as well as the applications for various platforms and architectures.

MeeGo and OBS Training

The Linux Foundation offers many training courses around Linux: http://training.linuxfoundation.org/. For MeeGo and OBS the following courses provide in-depth information and hands-on experience:

- LF363 Portable Application Development for MeeGo Devices
- LF361 Hands-on Approach to MeeGo Internals and OBS Build System
- LF273 MeeGo Training For Non-Developers

Training courses are offered in various formats including Classroom Training, Online Training, On-Site Corporate Training and Custom Corporate Training. Please contact the Linux Foundation (http://training.linuxfoundation.org/contact/solutions) for more information.

References

[1] An introduction to the to Open Build Service (OBS), Rudolf Streif, The Linux Foundation

[2] OBS Setup and Administration – Step-by-step Instructions to your own OBS Instance, Jan-Simon Moeller and Rudolf Streif, The Linux Foundation

[3] Open Build Service Wiki: http://en.opensuse.org/Portal:Build_Service

[4] Open Build Service Instance: https://build.opensuse.org/

[5] MeeGo Build Infrastructure: http://wiki.meego.com/Build_Infrastructure

[6] MeeGo Build Service Instance: http://build.meego.com/

[7] MeeGo Community Build Service Instance: https://build.pub.meego.com/

About the Author

Rudolf Streif manages The Linux Foundation's initiatives for embedded solutions working with the community to provide environments and platforms for embedded Linux systems.

About MeeGo

The MeeGo project is the open source software platform for the next generation of computing devices. MeeGo combines Intel's Moblin™ and Nokia's Maemo projects into one Linux-based platform.

MeeGo will be deployed across many computing device types - including pocketable mobile computers, netbooks, tablets, mediaphones, connected TVs and in-vehicle infotainment systems, and brings together the leaders in computing and mobile communications as the project's backers. MeeGo is designed for cross-device, cross-architecture computing and is built from the ground up, for a new class of powerful computing devices.

The Linux Foundation's MeeGo workgroup welcomes contributors and encourages developers and others to get involved at the site or find out more about joining the Linux Foundation at http://www.linuxfoundation.org/meego. Moblin and Maemo contributors are encouraged to participate at MeeGo.com.