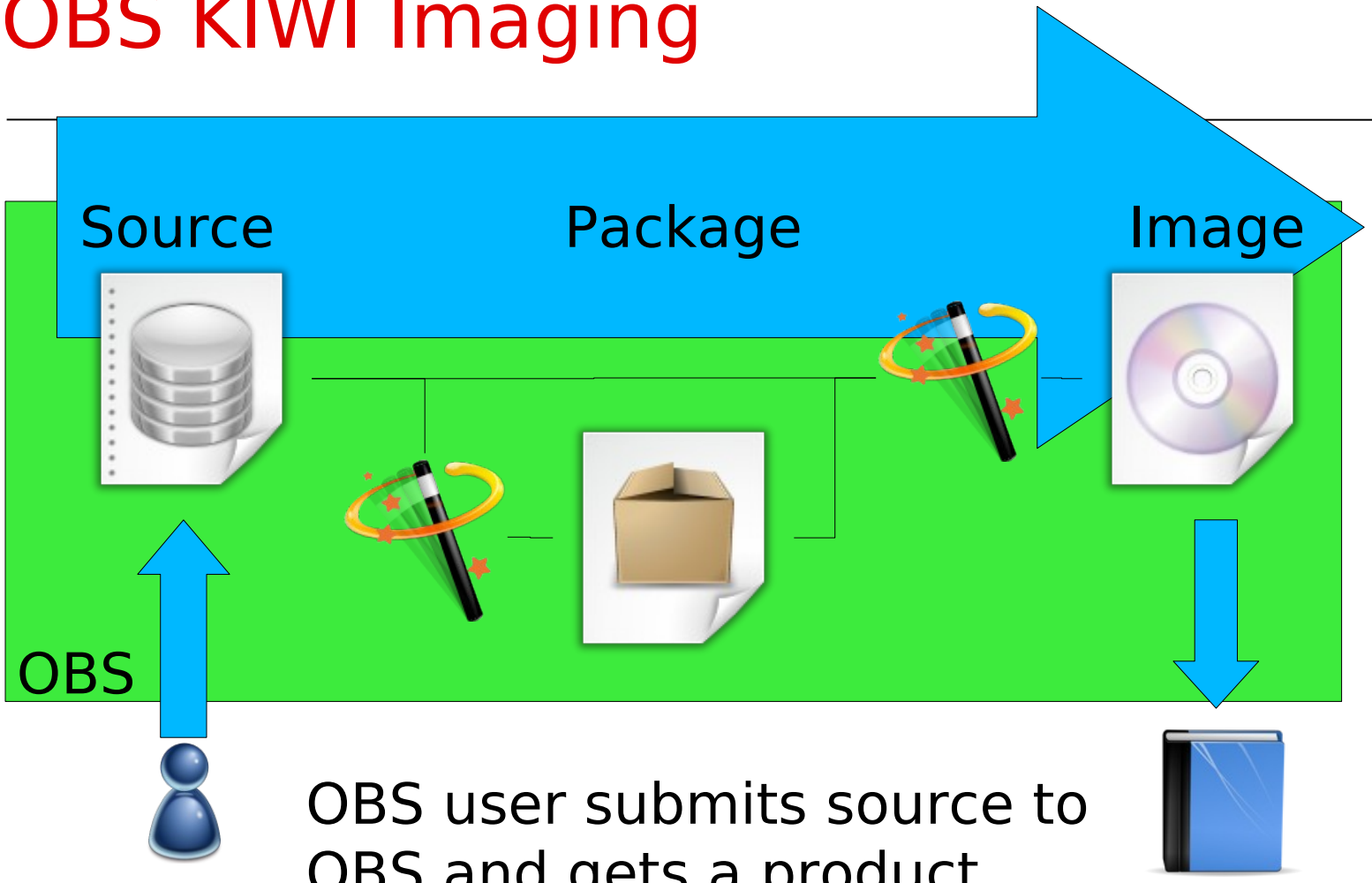
KIWI Imaging with openSUSE Build Service

Adrian Schröter
Project Manager Build Enviroment

Novell

OBS KIWI Imaging



OBS user submits source to OBS and gets a product

OBS Imageing compared to other KIWI solutions 1/2

Running KIWI manually:

- All KIWI functionalities are usable.
- Best way to hack on KIWI.
- Build happens local.

Imaging in Studio:

- For fast and easy image creation.
- Easy and integrated testing of the image.
- Workflow and tool guided image creation.
- Interactive working style.
- Server side image creation

OBS Imageing compared to other KIWI solutions 2/2

Imaging in Build Service:

- Batched processed image building depending on single package build results.
- Currently recommended for product/installation medias.
- Low-Level / Command line interface only.
- Allows usage of modified kiwi tool or kiwi descriptions in own project.
- Server side and local building options.
- Integrating of regular image builds into maintenance process for official products.
- Supports multiple KIWI version per Image (using it from the projects).

Image Builds from OBS POV

OBS knows currently these types of packages:

- rpm/spec builds
- deb/dsc builds
- KIWI Image (aka known as appliance image)
- KIWI Product Image (aka Installation Media)

Planned:

- QA builds
- MS Windows builds

→ Image builds are just another “package” build for the Build Service.

Limitations of Image builds within OBS

The OBS has as highest goal a clean and reproducable image build, as soon as possible (eg. not waiting for OpenOffice build when not needed). As a result we have the following limitations compared to plain KIWI usage:

- Only OBS repositories can be used.
- Own/modified boot description templates needs to get packaged.
- Used packages must be unambiguous !
- Currently no pattern support.
- Server may wait for building packages and does not start immediately.
  → Local *osc build* works at any time.
- Non-ISO build results are stored in tar ball, extended with Build number.

How to setup a KIWI repo

- Create a repository in a project.
  - Enable wanted architectures
  - No other repository needed in project config. KIWI's xml is specifing it.
- Create project config, setting this repository to
  - Type: kiwi
  - Repotype: none
- Create a package
- Submit adapted KIWI config files.

What needs to be changed in KIWI configs for OBS ?

- The config.xml needs to be suffixed as .kiwi
- Repositories needs to be specified as obs://$PROJECT/$REPOSITORY
  - obs:// refers always to the used build service.
  - Example: obs://openSUSE:11.1/standard
- Content of root directory needs to get packaged as root.tar or root.tar.bz2
- In case of expansion error “have choice” just select a package and add it to your package list.

Examples

openSUSE Factory Live CD in
→ openSUSE:Factory:Live Project

KDE:Media Live CDs in
→ KDE:Medias Project

OBS worker images (netboot deployment) in
→ openSUSE:Tools Project

Future Plans

- Support patterns
- Integrate into QA system for testing a produced build automatically (NOT interactive).
- Connect to SUSE Studio somehow for kiwi config exchange

Installation Media Creation (aka Product Creation)

What are Products ?

- Products are SUSE specific.
- Products are medias with plain rpm packages, to be handled via YaST or zypper.
- The Media may be bootable.
- Medias can be CD iso files, DVD iso files or FTP trees.
- The media may support multiple architectures.
- Examples are the openSUSE 11.1 DVD or the Non-OSS FTP tree Add-On.

A Product from KIWI POV

- A product KIWI config looks complete different to a system image. (Own section)
- No automatic dependency solving between packages.
- It works only with local rpm repositories currently.
- KIWI needs to deal with
  - RPM package which are used for installation
  - Meta packages (get extracted on the media)
  - Generate meta data

A Product In Detail

A typical product media consist of:

- An rpm repository
- Meta data
  - Patterns (prepared package selections)
  - Bootable initrd starting YaST for installation
  - Theming
  - EULA / License Information

A product may consist of multiple product medias !

Example Product

OpenSUSE 11.1 comes as:

- DVD5 for i586, x86_64 and ppc each
- DVD9 for i586 and x86_64 together
- FTP tree for i586 and x86_64 together
- FTP tree for ppc and ppc64 together
- NET boot media i586, x86_64 and ppc each

OpenSUSE 11.1 Non-OSS comes as:

- CD for i586, x86_64 and ppc each
- FTP tree for i586 and x86_64 together
- FTP tree for ppc

The Problem

Each product media needs

- An own kiwi config
- An own release flavor package
- Meta packages to be put one the media.

This means in each of them is some data which needs to be kept in sync. Like package lists or the Beta/RC version.

The Solution

- We have product configs in Build Service, specifing all medias for a Product.
- Multiple Products from one code stream can share definitions(eg SLE-11 or openSUSE:11.1).
- The OBS product converter creates
  - All kiwi config files
  - A spec file for release packages, including all flavors.
  - Patterns on media (in future)
- Product definitions are stored in “_product” package, all resulting sources gets generated as “_product:....” packages on checkin time.

Nice New Features

- KIWI allows to collect automatically all required source and debug packages.
- Not Yet: One place to maintain package lists for products and patterns.
- Not Yet: Automatic dependency solving for products optional.

Examples and Documentation

- openSUSE 11.1 was the first product using this.
- Product Definition wiki pages
- And of course the general KIWI documentation describing how to create an installation source manually.

Future

- Adapt KIWI after PDB migration
  - Obsolete some meta packages
  - Obsolete some autobuild tools with native implementation

    → Significant speed up hopefully
- Support Driver Update Disks in KIWI
- Support pattern generation based on product config
- Code/return value cleanup
- Media overflow handling ?
- Optional package dependency resolving ?
- KIWI remote repository support ?

Show real life product and kiwi configs

Novell®

General Disclaimer

This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. Novell, Inc., makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc., reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All Novell marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.

Novell