openSUSE - KIWI Image System

System Design

openSUSE - KIWI Image System: System Design

$Date: 2007-02-13 01:54:25 +0100 (Tue, 13 Feb 2007) $

Published October 30, 2007

Draft Version

This book is a draft. There might be errors, inconsistencies, typos and other broken things. If you want to contribute, please look at `http://developer.novell.com/wiki/index.php/OpenSUSE_Build_Service/documentation/` for more information.

Legal Notice

Copyright (c) 2007 Novell, Inc. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with the Invariant Section being Trademark Policy, with the Front-Cover Texts being *Build Service* and *make up something nice and shiny*. A copy of the license is included in Appendix A, *GNU Free Documentation License*.

Disclaimer. All information found in this book has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Neither Novell, Inc., SUSE LINUX Products GmbH, the authors, nor the translators shall be held liable for possible errors or the consequences thereof.

Trademark Policy. For Novell trademarks, see the Novell Trademark and Service Mark list `http://www.novell.com/company/legal/trademarks/tmlist.html`. Linux is a registered trademark of Linus Torvalds. All other third party trademarks are the property of their respective owners. A trademark symbol (®, ™ etc.) denotes a Novell trademark; an asterisk (*) denotes a third party trademark.

Table of Contents

1. Image Building with KIWI 1
1.1. Introduction 1
1.2. Creating Operating System Images 4
1.3. The KIWI Image Description 4
1.4. Activating an Image 12
1.5. Maintaining Operating System Images 19
1.6. Real-Life Scenarios - A Tutorial 21
A. GNU Free Documentation License 33

List of Figures

1.1. Image Serving Architecture 2
1.2. The Boot Process of a Netboot Client 13
1.3. Image Maintenance Scenarios 20

List of Examples

1.1. Basic kiwi Commands 25
1.2. Example USB Configuration File 26
1.3. How to Retrieve dd status 29
1.4. Basic kiwi Commands for OEM 29

Chapter 1. Image Building with KIWI

The openSUSE KIWI Image System provides a complete operating system image solution for Linux-supported hardware platforms as well as for virtualization systems such as Xen and VMware. KIWI is an ideal solution for creating and imaging thin client devices.

- Section 1.1, “Introduction”
- Section 1.2, “Creating Operating System Images”
- Section 1.3, “The KIWI Image Description”
- Section 1.4, “Activating an Image”
- Section 1.5, “Maintaining Operating System Images”
- Section 1.6, “Real-Life Scenarios - A Tutorial”

1.1. Introduction

KIWI is a complete imaging solution that can be divided in to three distinct stages. The first stage, which is based on a valid *software package source*, creates a *physical extend* as provided by an image description. The second stage creates an operating system image from the physical extend. This image is called a *logical extend* or *operating system image*. In the third stage, you deploy the new image.

- the section called “Physical Extend”
- the section called “Logical Extend”
- the section called “Deployment”

Physical Extend

In this stage, you prepare a directory containing the contents of your new filesystem based on a software package source (such as SUSE Linux Enterprise Desktop 10) and an image description file (`config.xml`). During this stage, you determine which packages are installed on your image and which configuration files are included.

The physical extend requires at least one valid software package source, called a repository, to access the software you want to use to build an image. A repository consists of software packages, organized in a package tree (also called an image description tree), and meta data. Because software repositories exist in many different formats, KIWI uses a package manager to access them. KIWI lets you use either the Smart Package Manager [`http://labix.org/smart`] or Zypper [`http://en.opensuse.org/Zypper`].

To prepare the physical extend, use the following command:

```
kiwi -p /path
```

`/path` is the location of your `config.xml` file.

This command creates the directory `/tmp/kiwi.<random>`, where *`random`* is a random string.

For more detailed information, see Section 1.2, “Creating Operating System Images”.

Logical Extend

This stage, based on the physical extend, creates an operating system image. It takes place without user interaction, so all the necessary information needs to be created prior to the image building process.

You can use the `images.sh` configuration script while creating the logical extend to clean-up your image or to perform additional configuration. This script is called at the beginning of the image creation process.

To create the logical extend, use the following command:

```
kiwi -c /path1 -d /path2
```

/path1 is the path to your previously prepared `/tmp/kiwi.<random>` directory, and */path2* is the destination output path.

This command creates an image file and an associated MD5 checksum.

For more detailed information, see Section 1.3, “The KIWI Image Description”.

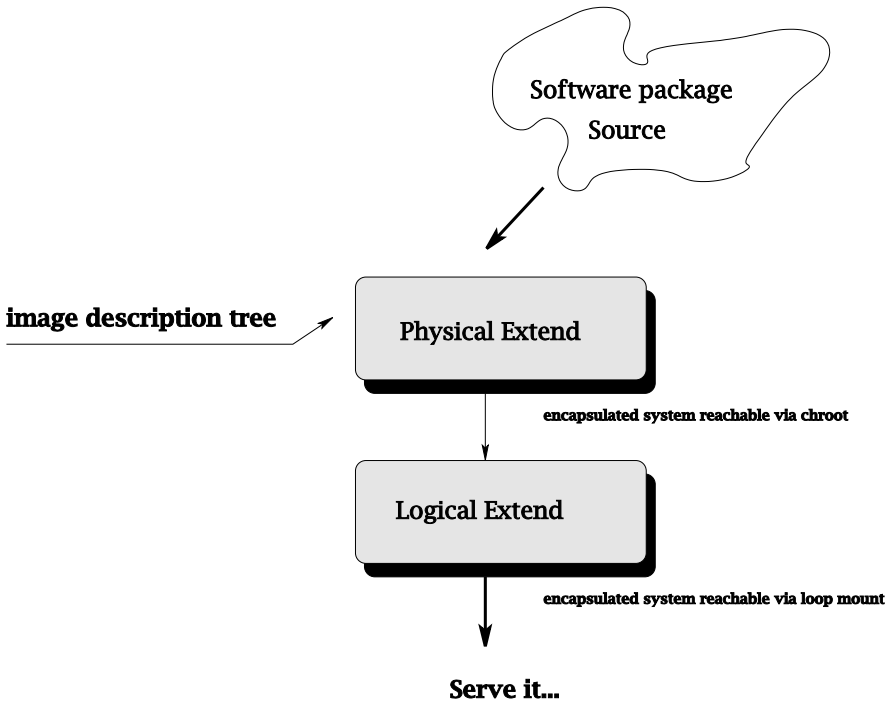
- the section called “Operating System Images ”
- the section called “Supported Image Types”

Operating System Images

A regular installation process starts with an installation image, then installs single pieces of software until the system is complete. This type of installation process is usually interactive, meaning that the user can modify installation settings and configuration options. An operating system image, on the other hand, represents an already completed installation encapsulated in a file. It is a copy of a system and its related data (such as the kernel, file systems, libraries, and programs) at a given point in time. It can even include the configuration for a specific task.

An operating system image starts working as soon as the image is installed on some sort of system storage device, no matter if the device is volatile or nonvolatile. An operating system image is deployed “as is,” and no user interaction is possible.

Figure 1.1. Image Serving Architecture



The process of creating an operating system image takes place without user interaction, so all requirements of the encapsulated system must be fulfilled before the image is created. The image description tree stores all the information needed to create an image.

Supported Image Types

The logical extend is the final result of an image creation process, and it represents an operating system as part of a specific filesystem which could also be covered by the structures of real or virtual hardware layers. KIWI supports the following image types:

Live-system image [ISO]
: The ISO image type (`.iso`) is used to create a live system (an operating system on a CD or DVD). When the system boots, all data is read from the CD/DVD. The system provides write support, but all data is stored in RAM, so as soon as the system shuts down, the data is lost. The generated `.iso` file must be burned on the media.

Virtual hard disk image [VMX]
: The VMX image type is used to create a virtual disk which provides partition information, the boot manager, and all other data found on a real disk. This kind of image can be used as a disk for full-virtual systems such as QEMU or VMware. KIWI also creates the VMware configuration file if requested. The generated image requires virtualization software to be installed. In a full virtualized environment, all components are virtualized, including the storage devices, the processor, and all other parts of the system. To activate a virtual disk system, the user calls the correct “player” application. For QEMU, use `qemu`, and for VMware, use `vmplayer`.

Paravirtual Xen image [Xen]
: The Xen image type is used to create an operating system image based on a given filesystem including a special Xen boot image, the Xen kernel, and the Xen configuration file. If the current system is a Xen hypervisor [`http://en.wikipedia.org/wiki/Hypervisor`], you can create new paravirtual machines with the data provided by KIWI. The generated image requires a Xen hypervisor running on the computer.

USB-stick image [USB]
: The USB image type is used to create an operating system image based on a given filesystem including a special USB boot image. KIWI can install the generated image onto a USB stick. If your system BIOS can boot from a USB stick, you can use the stick as complete operating system. Compared to a live system, a USB stick provides permanent storage of your private data. The generated image can be installed by a subsequent KIWI call.

Network image [PXE]
: The Preboot eXecution Environment (PXE) is used to boot computers using a network interface card independent of available data storage devices (such as a hard disk) or an installed operating system. The image itself is stored on a server, and a boot image downloads and activates it. The generated image requires a DHCP/TFTP network infrastructure running on a server.

Network split image [split]
: The split image type is used to create an operating system image based on two given filesystems. The image is divided into two portions, with the first portion representing the data which requires Read-only access and the second portion representing the Read-write data. The first portion can be part of a compressed Read-only filesystem. This image type does not automatically create a boot image, and it works only with netboot boot images. The generated image requires a DHCP/TFTP network infrastructure running on a server.

The image base type is referenced in the main image configuration file `config.xml` and has several mandatory/optional parameters. The parameters influence the operating system environment like the used filesystem but the result is still an image made for the purpose described by the base type.

Deployment

You can use any of the following methods to deploy your new image:

- **Netboot:** Advanced Trivial File Transfer Protocol (ATFTP) is used to deploy the newly created image via PXE. You can use KIWI to create a special kernel and initrd for the purpose of deploying a KIWI based image. You can load the image into volatile RAM or onto non-volatile, persistent storage such as flash memory or a hard drive. KIWI checks to see if a KIWI image is installed on the target system. If the version installed on the target system is older than the version being served, the image on the target system is updated.

- **Locally Accessible Hard Disk:** The image can be installed by dumping (`dd`) the image file on a previously created partition on this disk. To activate the system, use a boot manager such as GRUB or LILO.
- **Paravirtualized:** In case of a paravirtualized target system such as Xen, install the image by copying the image file on the target system. To activate the virtual system, a configuration must be provided which points to the image in some way (for example, you can use a loop mounted location).
- **Fully Virtualized:** In case of a full virtualized target system such as VMware, the image represents a virtual disk as a file which can be played by the virtualization system.

For more information, see Section 1.4, “Activating an Image”.

1.2. Creating Operating System Images

To create an operating system image with KIWI, you must first create a *system image description*. This description is a directory containing at least one file named `config.xml`. Among other things,.the `config.xml` file lists the packages and patterns used to make your image, the source repositories the data should be obtained from, and the image types that can be generated from this description (see Section 1.3, “The KIWI Image Description” for more information).

You can use an existing system image description as a template. An image description is normally provided as an architecture-independent package. For your reference, you can download `kiwi-desc-livesystem` from `http://download.opensuse.org/repositories/openSUSE:/Tools`. This package contains descriptions for openSUSE live systems, which can serve as a basis for other image types.

The process of creating an image starts with a `prepare` command, which requires at least the path to your system image description and an optional destination directory as arguments.

```
kiwi --prepare /path_to_description --root /tmp/myroot
```

The result of this step is an operating system which has its root directory below the given destination directory (`/tmp/myroot` in this example). After this step, the new root directory serves as the data source for creating the requested operating system image. This step is called the *create* step, and requires, at minimum, the path to your previously created new root environment and a destination directory for the image files.

```
kiwi --create /tmp/myroot --destdir /tmp/myimages
```

The destination directory (`/tmp/myimages` in this example) must exist before you run the command. KIWI stores all files for the requested image type below this directory. KIWI lets you specify *defaultroot* and *defaultdestination* attributes as part of `config.xml`. If these attributes are present, and the *--root* and/or *--destdir* options are not specified, the information from the *default* attributes is used.

In addition to creating your system image, KIWI also provides a subsystem for image deployment. Each system image needs to be activated in some manner. For example, the system image for a network client must be downloaded from a server, installed, and a disk must be partitioned. The steps that must be performed before a system image is activated are done in the so called *boot image*. KIWI provides boot images for all of its supported image types. KIWI creates them automatically when the `create` command is run.

The KIWI boot images are special initrd images that contain the image type they should activate. The boot images are implemented as normal KIWI image descriptions, and are stored in a subdirectory below `/usr/share/kiwi/image`. The subdirectories use the naming scheme *`type`*`boot`. For example, the boot images for network images are stored in the directory `netboot/`, and the boot images for virtual disk images are stored in `vmxboot/`. Boot images should not require any modifications by the user. For more information on boot images, see the section called “ `config` ”.

1.3. The KIWI Image Description

An image description tree is used to create an operating system image from a physical extend. It contains the set of files required to generate an image, and is arranged in the following order:

|

```
 |- config.xml
 |- config [optional]
 |   |- package configuration scripts
 |
 |- cdboot [optional]
 |   |- isolinux.cfg
 |   |- isolinux.msg
 |   |- isolinux.sh -> ../../suse-isolinux
 |
 |- config.sh [optional]
 |- images.sh [optional]
 |- config-yast.xml [optional]
 |- config-cdroot.tgz [optional]
 |- root [optional]
     |- root tree files/directories
```

config.xml

`config.xml` is the main configuration file used to define the image type, base name, repositories, profiles, options, and the package/pattern list.

All values entered in the `image`, `preferences`, and `drivers` elements of the `config.xml` file are also stored in a file called `.profile`. This file is created before the execution of an image script such as `config.sh` or `images.sh`, or a "package script." This means that the `config.xml` file parameters are available as variables that can be sourced via script. This kind of script should follow this template (replace *`name`* with the actual image name):

```
#!/bin/sh
test -f /.kconfig && . /.kconfig test -f /.profile
test -f /.profile && . /.profile test -f /.profile
echo "Configure image: [$name]..."
...
exit 0
```

This kind of script is called within the image environment, which means that is not possible to damage the host system with this script even if you are using absolute paths.

The following is a sample `config.xml` file:

```
<image name="Name" inherit="optional path" schemeversion="1.4">❶
  <description type="boot|system">❷
    <author>Author</author>
    <contact>Contact</contact>
    <specification>Specification</specification>
  </description>
  <preferences>❸
    <type primary="true" boot="..." filesystem="..." flags="..."
          bootprofile="..." baseroot="..." checkprebuilt="true|false">Type</type>
    <version>Version</version>
    <size unit="Unit">Size</size>
    <packagemanager>Name</packagemanager>
    <rpm-check-signatures>True|False</rpm-check-signatures>
    <rpm-force>True|False</rpm-force>
    <keytable>Name</keytable>
    <timezone>Name</timezone>
    <locale>Name</locale>
    <defaultdestination>Path</defaultdestination>
    <defaultroot>Path</defaultroot>
    <defaultbaseroot>Path</defaultbaseroot>
```

```
    <compressed>Yes|No</compressed>
  </preferences>
  <profiles>❹
    <profile name="Name"
          description="Description"/>
  </profiles>
  <users group="Groupname">❺
    <user name="User"
          pwd="Password" home="Homedirectory"/>
  </users>
  <drivers type="Type"
    profiles="Name">❻
    <file name="Filename"/>
  </drivers>
  <repository type="Type">❼
    <source path="Url"/>
  </repository>
  </deploy server="IP address"
    blocksize="Size">❽
    <partitions device="Devicename">
      <partition type="Type"
          number="Number" size="Size"/>
    </partitions>
    <union rw="RW-Device" ro="RO-Device" type="aufs|unionfs"/>
    <configuration source="Source"
          dest="Destination"/>
  </deploy>
  <packages type="Type"
          patternType="onlyRequired|plusSuggested|plusRecommended"
          patternPackageType="onlyRequired|plusSuggested|plusRecommended"
          profiles="Name">❾
    <package name="Packagename"
          arch="Arch"/>
    <opensusePattern name="Patternname"/>
    <ignore name="Packagename"/>
  </packages>
</image>
```

❶ The `image` element contains the `name` attribute, which specifies the base name of the image. The version number and current date are added automatically, with the version number extracted from the directory in which the description files for this image are located. The `inherit` attribute is optional, and if you specify a path to another KIWI description, this description is prepended to the current description. The `schemeversion` attribute is mandatory, and allows only version <emphasis>1.4</emphasis> at the moment.

❷ The `description` element contains the `author`, `contact`, and `specification` attributes. `Author` is the name of the person responsible for this image. `Contact` is a valid e-mail address for the responsible person. `Specification` is a description of the image.

❸ The `preferences` element contains the following information needed to create the logical extend:

type
: The image type of the logical extend. When specifying multiple entries, the additional attribute `primary`, which specifies the primary type, must be filled in, or the first entry of the `type` value is used.

  The following values are valid for `type`:

  1. ext2, ext3, reiserfs, squashfs, cpio
  2. iso, split, usb, vmx, xen, pxe

The second group of types requires additional attributes:

- **primary:** Specifies the primary type. The `KIWI` option `--type` lets you choose the type.

- **boot:** Specifies the boot image (`initrd`) which should be used and created for this system image description. The boot images for KIWI are stored in `/usr/share/kiwi/image`, and are grouped by function into the following directories: `isoboot`, `netboot`, `usbboot`, `vmxboot`, and `enboot`. The attribute value is the path relative to `/usr/share/kiwi/image`.

- **flags:** Specifies flags for the image type. Currently, the only available flag is `compressed` for the `iso` type. This flag causes live media to be based on a squashfs compressed file system.

- **filesystem:** Specifies the filesystem, which can be any of the following:`ext2`, `ext3`, `reiserfs`, `squashfs`, or `cpio`.

- **bootprofile:** Specifies an addon boot profile. In `config.xml`, you can bind specific packages/drivers into a namespace. This is called a profile. The information in a profile is used if the `KIWI` option `--add-profile` is used, or if the profile name is included as a value in the `bootprofile` attribute. This feature is used with the netboot image to distinguish between different kernels (for example, between the default kernel and the Xen kernel).

- **baseroot:** attribute to specify the path to a pre created base root system. With KIWI it's possible to prepare and create an image based on a non empty root directory, the so called base root. KIWI will setup the image based on the contents of this directory. This could speedup the build process a lot. Depending on the output type KIWI will call itself in order to be able to create the boot image required for the output type. To make this boot image aware of a possible base root path the attribute *baseroot* is used.

- **checkprebuilt:** Specifies whether or not kiwi should search for a prebuilt boot image. The prebuilt boot images must be stored in a directory named `/usr/share/kiwi/image/*boot/*-prebuilt`.

The following is a list of boot image (`initrd`) types. KIWI automatically creates the specified *...boot* boot image during the creation of the system image. The description for the boot image must exist in `/usr/share/kiwi/image/...boot/suse-...`

- **<type boot="isoboot/suse-..." flags="unified">iso</type>:** If the optional `flags` attribute is set to `compressed`, squashfs compression is used for the read-only portion of the ISO image. If flags is set to `unified`, the squashfs compressed read-only portion is used in combination with the union filesystem named `aufs`, which allows for a complete read-write system as long as the system runs.

- **<type boot="usbboot/suse-..." filesystem="ext3">usb</type>:** The `filesystem` attribute can be set to any of the following: `ext2`, `ext3`, `squashfs`, or `reiserfs`. The created system and boot images are suitable to run on a USB stick, using the KIWI `--bootstick / --bootstick-system` options.

- **<type boot="vmxboot/suse-..." filesystem="ext3" format="...">vmx</type>:** The `filesystem` attribute can be set to any of the following: `ext2`, `ext3`, or `reiserfs`. The final result is one file appearing as a virtual disk, including the system and boot images, the disk geometry partition information, and the boot manager. The `format` attribute can be set to any of the following: `vvfat`, `vpc`, `bochs`, `dmg`, `cloop`, `vmdk`, `qcow`, `cow`, or `iso`. With the exception of the iso format, these are virtual disk formats supported by the QEMU disk tool qemu-convert. The iso format should be use only in combination with an *oemboot* boot image. Using this format generates an install CD/DVD from the created virtual disk image. To install the virtual disk system from a CD onto a real computer hard disk, you must use the oemboot boot image. The process creates a bootable ISO image, including the virtual disk image and the oemboot boot image. The oemboot image first installs the virtual disk image file onto the real hard disk using the `dd` command. Because the virtual hard disk contains a boot manager, the newly installed system can boot after the installation. The oemboot

image then re-partitions the disk according to its real disk geometry, installs the distribution default initrd, and boots into the final system.

- **<type boot="xenboot/suse-..." filesystem="ext3">xen</type>:** The `filesystem` attribute can be set to any of the following: `ext2`, `ext3`, or `reiserfs`. The created system and boot images, as well as the xen configuration file, can be used to run a Xen virtual machine using the `xm` program.

- **<type boot="netboot/suse-..." filesystem="ext3">pxe</type>:** The `filesystem` attribute can be set to any of the following: `ext2`, `ext3`, or `reiserfs`. The value of `boot` attribute contains the path to a netboot boot image description. The created system and boot images are suitable to drive a network client station. An appropriate network infrastructure that includes a DHCP and TFTP server is required in order to activate a network client.

- **<type boot="netboot/suse-..." filesystem="*type-rw*,*type-ro*">split</type>:** The `filesystem` attribute specifies a filesystem pair in which *`type-rw`* specifies `ext2`, `ext3`, or `reiserfs`, and *`type-ro`* specifies `ext2`, `ext3`, `reiserfs`, `cramfs`, or `squashfs`. The ability to booting split images is supported only by the netboot boot images (which must be created with a separate KIWI run).

version
: The three-part version number with the following format:

  `Major``.`*`Minor`*`.`*`Release`*

  The following rules should be applied when incrementing the version number:

  - Increment the `Release` number on minor modifications if no packages have been added or removed.

  - Increment the `Minor` number and reset the `Release` number to `0` if packages have been added or removed.

  - Increment the `Major` number if the size of the image changed.

size
: Specifies the size of an image with a numerical value in Megabytes or Gigabytes. Use the `unit` attribute to assign the unit `M` for Megabytes or `G` for Gigabytes.

  KIWI extends the image size automatically if the specified value is too small. If the actual size is more than 100MB larger than the specified size, KIWI aborts with an error message. KIWI does not automatically reduce the image size if the specified value is too large, because the extra space might be needed to, for example, run custom scripts. If no size is specified, KIWI uses the required size plus approximately 10% free space.

packagemanager
: Specifies the packagemanager to be used for package installation. The default packagemanager is `smart`, but `zypper` is also supported.

compressed
: Specifies whether the system image should be compressed or not. This affects only the KIWI output file itself and has no influence on the operating system contained in the image. For network clients, download the image as a compressed image.

keytable
: Specifies the name of the console keymap to use. The value corresponds to a map file in `/usr/share/kbd/keymaps`. The `KEYTABLE` variable in `/etc/sysconfig/keyboard` file is set according to the keyboard mapping.

timezone
: Specifies the time zone. Available time zones are located in the `/usr/share/zoneinfo` directory. Specify the attribute value relative to `/usr/share/zoneinfo`. For example, specify `Europe/`

`Berlin` for `/usr/share/zoneinfo/Europe/Berlin`. KIWI uses this value to configure the timezone in `/etc/localtime` for the image.

locale
: Specifies the name of the locale to use, which defines the contents of the `RC_LANG` system environment variable in `/etc/sysconfig/language`.

defaultdestination
: (Optional) Used if the `destdir` option is not specified when calling KIWI.

defaultroot
: Optional element `defaultroot` is used if the option `root` is not specified while calling KIWI.

defaultbaseroot
: Optional element `defaulbasetroot` is used if the option `--base-root` is not specified while calling KIWI. It's possible to prepare and create an image using a predefined non empty root directory as base information. This could speedup the build process a lot if the base root path already contains most of the image data.

defaultroot
: (Optional) Used used if the `root` option is not specified when calling KIWI.

❹ The optional `profiles` element lets you maintain one image description while allowing for variation in the set of packages and drivers that are included. A separate `profile` element must be specified for each variation. The `profile` child element, which has `name` and `description` attributes, specifies an alias name used to mark sections as belonging to a profile, and a detailed description explaining what this profile does. To mark a set of packages or drivers as belonging to a profile, simply annotate them with the `profiles` attribute. If a `packages` tag does not have a `profiles` attribute, it is assumed to be present for all profiles. This information also applies to the `drivers` element.

❺ The optional `users` element specifies the users to be added to the image. The `group` attribute specifies the group the users belong to. If this group does not exist, it is created. A `user` element must be specified for each group. The user child element specifies the users belonging to a group, and the `name`, `pwd`, and `home` attributes specify the username, password, and path to the home directory.

❻ The optional `drivers` element contains driver file names. The names are interpreted as general driver names and used if they are contained in the kernel tree. The `type` attribute specifies one of the following driver types:

`netdrivers`
: Each file is specified relative to the `/lib/modules/<Version>/kernel/drivers/net` directory.

`usbdrivers`
: Each file is specified relative to the `/lib/modules/<Version>/kernel/drivers/usb` directory.

`drivers`
: Each file is specified relative to the `/lib/modules/<Version>/kernel` directory.

❼ The `repository` element specifies the source path and type used by the package manager. The `type` attribute specifies the type of the repository, for example, <code>type="yast2"</code>. The `source` child element contains the `path` attribute, which specifies the location of the repository (for example, <code>source="/image/CDs/full-i386"</code>). The path specification can be any of the following, and can include the %arch macro if needed:

- A local path starting with `/`
- `this://` relative path name, which is relative to the image description being referenced
- `http://` or `ftp://` Network-Location
- opensuse://Project-Name

Multiple repository tags are allowed. For information on how to setup a smart source, see `http://labix.org/smart`.

❽ The `deploy` element is used only for network clients. It specifies the location of the system image, how the client disk should be partitioned, and the configuration files that should be included. The `server` and `blocksize` attributes specify the TFTP server which controls the download. The `partitions` tag specifies the partitions for one disk device (`device`). Each partition is specified by one `partition` subtag named which defines the type (see `sfdisk --list-type`), partition number, size, optional mountpoint, and optional information on if this partition is the system image target partition.

With the KIWI netboot image, the first partition is always the swap partition, while the second partition is used, by default, for the system image. With the optional `target` flag, you can specify a partition other than the second partition to install the system image on. If `size` is set to "image," KIWI calculates the required size for this partition in order to have enough space for the later image. The optional `union` tag is used if the system image is based on a read-only filesystem such as squashfs. In this case, KIWI creates an additional write partition, then combines both partitions with the given overlay filesystem. Currently, there are two such filesystems: unionfs and aufs (aufs is the preferred file system). The partition that holds the read-only system image must be set as the `ro` attribute value, and the partition that serves as the write partition must be set the `rw` attribute value.

The optional `configuration` tag can be used to integrate a network client's configuration files which are stored remotely on the server. The `source` attribute specifies the path on the server used by a TFTP client program to download the file, and the `dest` attribute specifies the target relative to the root (/) of the network client.

❾ The `packages` element specifies the list of packages and pattern names to be used with the image. There are five different types of package sets or patterns, specified with the `type` attribute:

image
: List of packages used to finish the image installation. All packages which make up the image are listed here.

boot
: List of packages used to start creating a new operating system root tree. Basic components which are required to `chroot` into that system, such as `glibc`, are listed here.

xen
: List of packages used when the image needs support for Xen-based virtualization. The `memory` and `disk` attributes define how much memory the virtual system requires and the kind of device the disk should appear as in the virtual instance. This information is used to create the appropriate Xen configuration file.

vmware
: List of packages used when the image needs support for VMware-based virtualization. The `memory` and `disk` attributes define how much memory the virtual system requires and the kind of device the disk should appear as in the virtual instance. This information is used to create the appropriate VMware configuration file.

delete
: List of packages stored for later deletion. The package names are available in the `$delete` variable of the `.profile` file created by KIWI. The `baseGetPackagesForDeletion()`function returns the contents of this environment variable, and can be used to delete the packages while ignoring requirements or dependencies.

Using a pattern name enhances the package list with a number of additional packages belonging to this pattern. Support for patterns is SUSE-specific, and available with openSUSE 10.2 or later. The optional `patternType` and `patternPackageType` attributes specify which pattern references or packages should be used in a given pattern. The value can be set to *onlyRequired*, which incorporates only patterns and packages that are required by the given pattern, *plusSuggested*, which incorporates patterns and packages that are required and suggested by the given pattern, or *plusRecommended*, which incorporates patterns and

packages that are required and recommended by the given pattern. By default, only required patterns and packages are used.

You can specify different values for `patternType` and `patternPackageType`. If a pattern contains unwanted packages, you can use the `ignore` element to specify an ignore list, with the `name` attribute containing the package name. To restrict a package to a specific architecture, use the `arch` attribute to specify a comma separated list of allowed architectures.

config

`config` is an optional subdirectory that contains shell scripts that are executed after all packages are installed. For example, you can include scripts here that remove parts of a package that are not needed for the operating system. The name of the bash script must resemble the package name listed in the `config.xml` file.

cdboot

`cdboot` is an optional directory used to create a bootable CD. This directory contains the files required by the isolinux boot loader, including the `isolinux.cfg` and `isolinux.msg` configuration files, and the `isolinux.sh` build script. The `cdboot` directory creates an ISO image from a prebuilt CD tree, based on the configuration information found in the `isolinux.cfg` file.

config.sh

`config.sh` is an optional configuration script used during the creation of the physical extend. This script is executed at the end of the installation after the `chroot` command is used to switch to the operating system image but before the package scripts paced in the config directory (see the section called “ `config` ”) are run. This script can be used, for example, to configure the image system by activating or deactivating services.

images.sh

`images.sh` is an optional configuration script executed at the beginning of the image creation process during the creation of the logical extend. This script removes the programs and files only needed while the physical extend exists.

config-yast.xml

`config-yast.xml` is an optional AutoYaST configuration file that creates a ready-to-use profile (`/root/autoinst.xml`) that can be used to create a clone of a particular installation. To generate this file, select *Clone This System for AutoYaST* during the installation of openSUSE. To create an autoinstallation file from scratch, or to edit an existing autoinstallation file, use the *Autoinstallation* YaST module.

To use `/root/autoinst.xml`, copy the file to the image description tree, then change the name to `config-yast.xml`. KIWI processes the file and sets up your image as follows:

- When the image is booted, YaST automatically starts in AutoYaST mode.
- YaST configures the system by applying the rules from `config-yast.xml`.
- If the process finishes successfully, the environment is cleaned, and AutoYaST will not be started with the next reboot.

config-cdroot.tgz

`config-cdroot.tgz` is an optional compressed tar archive used only for live systems. The data in the archive is uncompressed and stored in the CD/DVD root directory. This archive can be used, for example, to integrate a license file or readme information on the CD or DVD.

`root`

The `root` subdirectory contains special files, directories, and scripts used to adapt the image environment after the installation of all packages. The entire directory is copied into the root of the image tree using the `cp -a` command. The data in the `root` directory lets you customize your image with data that doesn't exist in the form of a package.

1.4. Activating an Image

After you create a logical extend (an image) from a physical extend, you can use any of the following methods to activate or deploy the image:

- On a *local system*, you can install the image by using the `dd` command to dump the image file on a previously created partition onto a local hard disk. To activate the system, use a boot manager such as GRUB or LILO. A USB stick system is also considered to be a local system, which KIWI supports with the `--bootstick` option.

- For a *network enabled system* (netboot client), use a special boot image to install the image. The boot image, which serves as the initial ramdisk (`initrd`), and the appropriate kernel are downloaded from a network service. The Linux kernel then automatically calls a program named `linuxrc`, which downloads and installs the system image. The installation can be done persistently on disk, or temporarily into the RAM of the machine. PXE is the network boot protocol supported by KIWI.

- For a *paravirtualized target system* such as Xen, install the image by copying the image file and the KIWI-created Xen configuration file onto the target system. To activate the virtual system, use the `xm create -c <xen_config_file>` command.

- For a *full virtualized target system* such as VMware or QEMU, the image represents a virtual disk which can be <quote>played</quote> by the virtualization system.

With all of these activation methods, a special image, called a boot image, controls the activation or deployment process. The following sections explain the KIWI boot images in more detail.

- the section called “The KIWI Netboot Image”

- the section called “The KIWI isoboot Image”

- the section called “The KIWI vmxboot Image”

- the section called “The KIWI oemboot Image”

- the section called “The KIWI xenboot Image”

- the section called “The KIWI usbboot Image”

The KIWI Netboot Image

The KIWI netboot boot image is used to install an operating system image on a network client. To establish communication with the client, a boot server infrastructure with the following services is required:

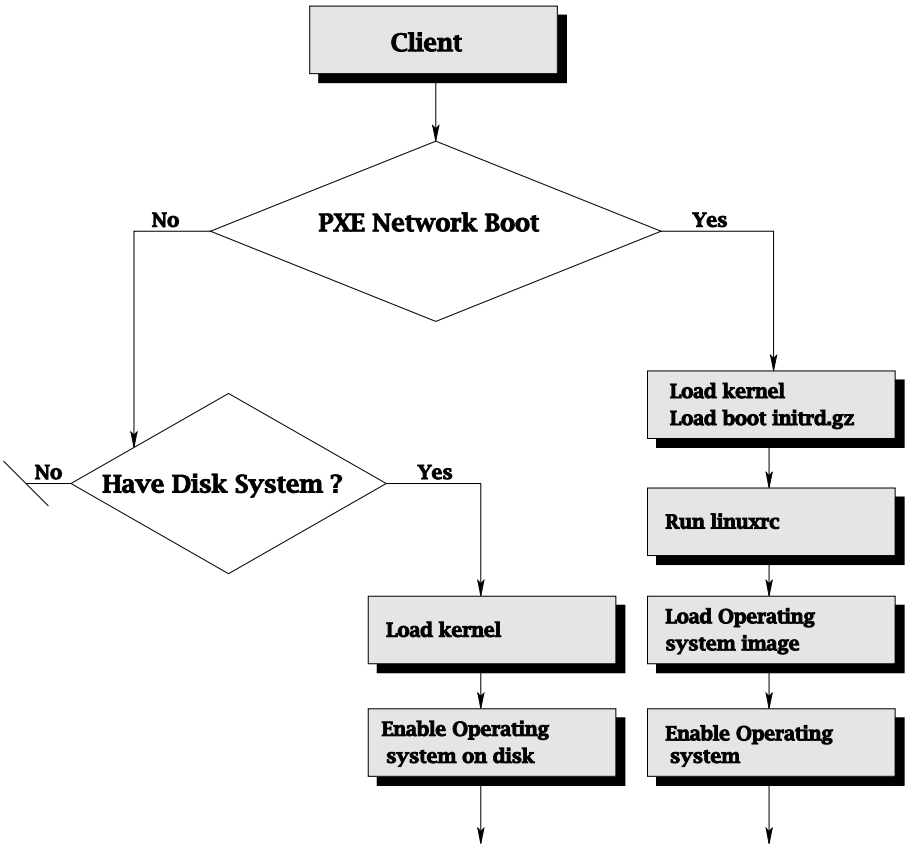
- A DHCP server to provide an IP address for the client.

- A TFTP server to allow file transfers to and from the client.

The Netboot System Boot Process

The following figure illustrates a simplified netboot client boot process:

Figure 1.2. The Boot Process of a Netboot Client



If the system can boot via a network, it loads the kernel and the compressed boot image from the network. The "brain" of the boot image is the `linuxrc` script, which does everything controlled by an image configuration file (also obtained from the network), including downloading and activating the operating system image. The boot image is exchanged for the operating system image to be activated.

The following overview describes the steps that take place when the netboot client is booted:

1. The client boots the initrd (`initrd.gz`), which is served by the TFTP server, via PXE network boot or boot manager (GRUB).

   If a PXE boot is not possible, the client tries to boot from a local hard disk.

2. The `linuxrc` command is run, which starts the following process:

   a. The file systems required to receive system data are mounted (for example, the `/proc` file system).

   b. The kernel parameters are imported. If there is an IMAGE parameter, it is assumed the system boots locally, and the `LOCAL_BOOT` variable is set.

   c. If `LOCAL_BOOT` is not set, network support is activated. The network card is probed using the `hwinfo` command. The appropriate module is loaded using `modprobe`. Any dependencies to other modules is resolved.

   d. If `LOCAL_BOOT` is not set, the network interface is set up via DHCP. After the interface has been established, the DHCP variables are exported into the `/var/lib/dhcpcd/dhcpcd-eth0.info` file, and the contents of DOMAIN and DNS are used to generate a `/etc/resolv.conf` file.

   e. If `LOCAL_BOOT` is not set, the TFTP server address is acquired. During this step, a check is made to see if the kiwitftp `kernel` parameter is set. If it is not set, a check is made to determine if the host name tftp.$DOMAIN can be resolved. If both tests fail, the DHCP server is used as the TFTP server. For more information about the TFTP server structure, see the section called "The TFTP Server Structure".

f. If LOCAL_BOOT is not set, the configuration file is loaded from the /srv/tftpboot/KIWI server directory via TFTP. At this point, the client expects the config.<*MAC Address*> file to be available. If this file is not available, a search is made for config.<*HEX-IP*>. *HEX-IP* is the IP address of the client, with each part of an IPv4 address converted into a hexadecimal value. If this file is not available, the IP address search checks only three out of four IP tokens, then two out of four, and so on. If still no file is found, a final search is made for a file named config.default. If this file is not available, it is assumed that the client did not existed before and can therefore be immediately registered by uploading a control file to the /srv/tftpboot/upload directory on the TFTP server. After the upload, the client branches off into a loop in which the following steps are taken:

   - The DHCP lease file is restarted (dhcpcd -n).
   - A new attempt is made to load the config.<*MAC Address*> file from the TFTP server.
   - If the file does not exist, there is a 60 second time-out before a new run begins.

   If the configuration file does load, it contains data on image, configuration, synchronization, or partition parameters. For more information on the configuration file format, see the section called “The Netboot Client Configuration File config.<*MAC Address*>”.

g. All registered kernel modules are loaded. The kernel provides a system that checks for a module alias registered automatically by the kernel during boot time. If such an alias matches the modinfo information from a kernel module, it is loaded.

h. If LOCAL_BOOT is not set, the PART: line in the configuration is analyzed. If it is found, a check is performed to see if any local systems need to be updated. If not, the local boot process continues immediately and no image download occurs. If an update is required, or no operating system is found, the client's hard disk is partitioned.

i. If NFSROOT, NBDROOT, and LOCAL_BOOT are not set, the images are downloaded with TFTP. If LOCAL_BOOT is set, this part of the initrd only reads the IMAGE information for later usage. If NFSROOT is set, the image root device is set to a remote NFS path. If NBDROOT is set, the nbd kernel module is loaded, and the nbd-client program sets up a new network block device called /dev/nd0. The image root device is then set to the network block device.

j. If LOCAL_BOOT is not set, the checksums are checked. If the check fails, another download attempt is started.

k. If LOCAL_BOOT is not set, a check for is made for RELOAD_CONFIG.

l. The operating system image is mounted.

m. If LOCAL_BOOT is not set, the CONF: line is evaluated. All the specified files are loaded from the TFTP server and stored in a /config/ path. The KIWI_INITRD: line is also evaluated. The specified initrd file is downloaded from the TFTP server and stored in the system image as /boot/initrd.

n. If LOCAL_BOOT is not set, all the user-land processes based on the boot image (dhcpcd -k) are terminated.

o. The contents of the COMBINED_IMAGE variable is evaluated to determine if the image is a splitted image. Both image parts are combined into one system by creating the appropriate filesystem links.

p. If LOCAL_BOOT is not set, the filesystem type of the system image and the available kernels are determined.

q. If LOCAL_BOOT is not set, the /etc/fstab, /boot/grub/menu.lst, /etc/grub.conf, and /etc/sysconfig/kernel configuration files are created.

r. If `LOCAL_BOOT` is not set, the configuration files stored in the `/config/`directory are copied into the mounted operating system image.

s. The system switches to the mounted operating system image. The root file system is converted to the operating system image via the `pivot_root` or `mount --move` command. All required configuration files are now present, either because they had been stored in the operating system image, or because they have been downloaded via TFTP.

t. The boot image is unmounted using the `exec umount` command.

u. At the termination of the `linuxrc` or `exec` command, the kernel initiates the `init` process, which starts processing the boot scripts as specified in `/etc/inittab` (for example, to configure the network interface).

The TFTP Server Structure

The TFTP server directory structure is divided into the following areas:

Image configurations
: The `/srv/tftpboot/KIWI/` directory contains the various `config.<MAC Address>` image configuration files.

Configuration files
: The`/srv/tftpboot/KIWI/<MAC Address>/` directory contains the various system configuration files, such as `xorg.conf`.

Boot files
: The `/srv/tftpboot/boot/` directory contains the `initrd.gz` and the kernel to boot.

PXE second stage boot loaders
: The `/srv/tftpboot/` directory contains the boot loaders for PXE (`pxelinux.0` and `mboot.c32`).

PXE configuration file
: The `/srv/tftpboot/pxelinux.cfg` file specifies the location of the PXE configuration file.

Images files and checksums
: The `/srv/tftpboot/image/` directory contains the image files and their checksums.

Upload area
: The `/srv/tftpboot/upload/` directory is where the `hwtype.<MAC Address>` files for registering new netboot clients are uploaded.

The Netboot Client Configuration File `config.<MAC Address>`

The `config.<MAC Address>` configuration file contains data about image, configuration, synchronization, and partition parameters. The configuration file is loaded from the TFTP server directory `/srv/tftpboot/KIWI` via TFTP for previously installed netboot clients. New netboot clients are immediately registered, and a new configuration file with the corresponding MAC address is created.

The following is an example of a cash register configuration file:

```
IMAGE=/dev/hda2;image/browser;1.1.1;192.168.1.1;4096
CONF=/KIWI/00:30:05:1D:75:D2/ntp.conf;/etc/ntp.conf;192.168.1.1;1024,     \
     /KIWI/00:30:05:1D:75:D2/xorg.xonf;/etc/X11/xorg.xonf;192.168.1.1;1024
PART=200;S;x,300;L;/,500;L;/opt,x;L;/home
DISK=/dev/hda
```

The following format is used:

```
IMAGE=device;name;version;srvip;bsize;compressed,...
SYNC=syncfilename;srvip;bsize
CONF=src;dest;srvip;bsize,...,src;dest;srvip;bsize
PART=size;id;Mount,...,size;id;Mount
JOURNAL=ext3
DISK=device
```

IMAGE
: Specifies which image (name) should be loaded with which version (version), and to which storage device (device) it should be linked to (for example, /dev/ram1 or /dev/hda2). The netboot client partition (device) hda2 defines the root file system /, and hda1 is used for the swap partition. The numbering of the hard disk device should not be confused with the RAM disk device, where /dev/ram0 is used for the initial RAM disk and cannot be used as a storage device for the second stage system image. We recommend using the device /dev/ram1 for the RAM disk. If the hard drive is used, a corresponding partitioning must be performed.

  srvrip specifies the server IP address for the TFTP download. It must always be specified, except in PART.

  bsize specifies the block size for the TFTP download. It must always be specified, except in PART. If the block size is too small according to the maximum number of data packages (32768), linuxrc automatically calculates a new blocksize for the download.

  compressed specifies if the image file on the TFTP server is compressed. If compressed is not specified, the standard download workflow is used.

  ## Note

  The download will fail if you specify compressed when the image is not compressed. It will also fail if you do not specify compressed when the image is compressed. The name of the compressed image must contain the suffix .gz, and it must be compressed with the gzip tool. Using a compressed image automatically deactivates the TFTP multicast download option.

CONF
: Specifies a comma-separated list of source:target configuration files. The source (src) corresponds to the path on the TFTP server, and is loaded via TFTP. The download is made to the file on the netboot client indicated by the target (dest).

PART
: Specifies the partitioning data. The comma-separated list must contain the size (size), the type number (id), and the mount point (Mount). The size is measured by default in MB. All size specifications supported by the sfdisk program are also allowed. The type number specifies the ID of the partition. Valid ID's are listed via the sfdisk --list-types command. Mount specifies the partition's mount point.

  - The first element of the list must define the swap partition. The swap partition must not contain a mount point. A lowercase letter x must be set instead.
  - The second element of the list must define the root partition.
  - If a partition should take all the space left on a disk, use a lowercase x letter as the size specification.

DISK
: Specifies the hard disk. This is used only with PART, and it defines the device the hard disk can be addressed with (for example, /dev/hda).

RELOAD_IMAGE
: If set to a non-empty string, RELOAD_IMAGE forces the configured image to be loaded from the server even if the image on the disk is up-to-date. Used mainly for debugging purposes, this option makes sense only on diskful systems.

RELOAD_CONFIG
: If set to an non-empty string, RELOAD_CONFIG forces all configuration files to be loaded from the server. Used mainly for debugging purposes, this option makes sense only on diskful systems.

COMBINED_IMAGE
: If set to an non-empty string, COMBINED_IMAGE indicates that the two images specified must be combined into one bootable image. The first image defines the read-write part, and the second image defines the read-only part.

KIWI_INITRD
: Specifies the KIWI initrd to be used for a local boot of the system. The value must be set to the name of the initrd file which is used via PXE network boot. If the standard TFTP setup suggested with the kiwi-pxeboot package is used, all initrd files reside in the /srv/tftpboot/boot/ directory. Because the TFTP server does a chroot into the TFTP server path, you must specify the initrd file as follows:

  KIWI_INITRD=/boot/<*name-of-initrd-file*>

NFSROOT
: For netboot images, you can mount the system image root filesystem remotely via NFS (Network File System). This means there is a server which exports the root filesystem of the network client in such a way that the client can mount it read/write. In order to do that, the boot image must know the server IP address and the path name where the root directory exists on this server. The information must be provided as in the following example:

  NFSROOT=<*NFS.Server.IP.address*>;<*/path/to/root/tree*>

NBDROOT
: For netboot images, you can mount the system image image root filesystem remotely via NBD (Network Block Device). This means there is a server which exports the root directory of the system image via a specified port. The kernel provides the block layer, together with a remote port that uses the nbd-server program. For more information on how to set up the server, see the nbd-server man pages. The kernel on the remote client can set up a special network block device named /dev/nb0 using the nbd-client command. After this device exists, the mount program is used to mount the root filesystem. To allow the KIWI boot image to use that, the following information must be provided:

  NBDROOT=<*NBD.Server.IP.address*>;<*NBD-Port-Number*>;/dev/<*NBD-Device*>
  <*NBD-Swap-Port-Number*>;/dev/<*NBD-Swap-Device*>

  The NBD-Device, NBD-Swap-Port-Number, and NBD-Swap-Device variables are optional. If they are not set, the default values are used (/dev/nb0 for the NBD-Device, port number 9210 for the NBD-Swap-Port-Number, and /dev/nb1 for the NBD-Swap-Device). The swap space over the network using a network block device is only established if the client has less than 48 MB of RAM.

UNIONFS_CONFIG
: For netboot and usbboot images, you can use unionfs or aufs as a container filesystem in combination with a compressed system image. The recommended compressed filesystem type for the system image is squashfs. In case of a USB-stick system, the usbboot image automatically sets up the unionfs/aufs filesystem. For a PXE network image, the netboot image requires a config.<*MAC*> configuration as in the following example:

  UNIONFS_CONFIG=/dev/sda2,/dev/sda3,aufs

  In this example, the first device /dev/sda2 represents the read/write filesystem, and the second device /dev/sda3 represents the compressed system image filesystem. The container filesystem aufs is used to cover the read/write layer with the read-only device to one read/write filesystem. If a file on the read-only device is going to be written, the changes inodes are part of the read/write filesystem. The device specifications in UNIONFS_CONFIG must correspond with the IMAGE and PART information. The following example illustrates the interconnections:

  ```
  IMAGE=/dev/sda3;image/browser;1.1.1;192.168.1.1;4096
  PART=200;S;x,300;L;/,x;L;x
  ```

```
UNIONFS_CONFIG=/dev/sda2,/dev/sda3,aufs
DISK=/dev/sda
```

Because the second element of the `PART` list must define the `root` partition, it is important that the first device in `UNIONFS_CONFIG` references this device as a read/write device. The second device of `UNIONFS_CONFIG` must reference the given `IMAGE` device name.

`KIWI_KERNEL_OPTIONS`
Specifies additional command line options to be passed to the kernel when booting from disk. For example, to enable a splash screen, use `vga=0x317 splash=silent`.

`KIWI_BOOT_TIMEOUT`
Specifies the number of seconds to wait at the GRUB boot screen when doing a local boot. The default is 10 seconds.

The Netboot Client Control File `hwtype.<MAC Address>`

The control file is used to set up new netboot clients when there is no configuration file available corresponding to the client MAC address. The control file is created using the MAC address information, and uploaded to the `/srv/tftpboot/upload` directory on the TFTP server.

The Netboot Client Hardware Info File `hwinfo.<MAC Address>`

The hardware info file contains hardware details of the new netboot clients. There is no configuration file corresponding to the client MAC address. The hardware info file is created using the MAC address information, and uploaded to the `/srv/tftpboot/upload` directory on the TFTP server.

The netboot boot image support the standard (default) profile, and a special Xen profile. If the `bootprofile` attribute is set to *xen*, the Xen kernel and modules are used within the boot image, which boots the system image directly with an enabled Xen kernel. This turns the system image into a hypervisor without requiring a reboot.

To boot a Xen kernel over the network via PXE, an additional boot loader called `mboot.c32` is required. `mboot.c32` is a COM32 module for H. Peter Anvin's SYSLINUX bootloader, which loads multiboot kernels and modules. It lets Xen be network-booted. For instructions on using mboot.c32, see com32/modules/mboot.doc [`http://www.kernel.org/git/?p=boot/syslinux/syslinux.git;a=blob;f=com32/modules/mboot.doc`].

The KIWI isoboot Image

Normally, an image is installed on a disk or into the main memory of a computer via a deployment architecture that uses a boot image to transfers the image into its final destination. With KIWI, this is handled by the isoboot image. You can use either of the following techniques to boot a live system:

- Use the *old* style to split the system image into Read-Only and Read/Write. The Read/Write image is pushed into main memory, while the Read-Only image is placed on the CD. (optional compression of the Read-Only image is possible). The isoboot image makes one root tree from both parts by using symbolic links. The CD-Boot structure of KIWI places the `/bin`, `/boot`, `/lib`,`/opt`, `/sbin`, and `/usr` directories on the CD, while the remainder of the directories are placed in the main memory of the system.

- Use the *new* style if you want to make use of the new overlay filesystems such as aufs or unionfs. In this case, the system image is not divided, and the isoboot boot image creates a writable RAM space which is overlayed into one root filesystem. The result is one Read/Write filesystem. You can also compress the system image, which lets you put more data on the CD. The new style is recommended over the old style.

Every isoboot description contains a `cdboot` directory with `isolinux.cfg` and `isolinux.msg` files. isolinux.cfg defines which boot parameters are used for booting the kernel, and `isolinux.msg` defines the message which is displayed if isolinux doesn't boot in graphics mode. An additional file, `isolinux.sh`, is a link to the standard suse-isolinux script, which creates an ISO image from a specified CD tree. The script also works

for non-SUSE distributions, but the CD header of these images contains information which should be replaced if the distribution is not SUSE-based.

The KIWI vmxboot Image

The KIWI vmxboot boot image is used for fully virtualized machines such as the ones provided by VMware or QEMU. The result of an image created with the VMX type is always a virtual disk containing the vmxboot image, the system image, a virtual disk geometry, and a boot loader (GRUB) to boot the vmxboot boot image.

The vmxboot boot image detects the virtual disk device, then activates the system on the virtual disk. The boot image also writes some default files such as `/etc/fstab` to allow the system default initrd to be created and used. This means that the vmxboot image is replaced during first boot by the default system initrd. The replacement only takes place if the default system initrd can boot the image, which is not the case, for example, if a compressed image solution with an overlay filesystem like aufs is used.

The KIWI oemboot Image

The KIWI oemboot boot image works only with the VMX boot type (and therefore only with virtual disks). The oemboot was designed for OEM customers who want to have a pre-installed Linux system delivered to their customers. The idea is to have a virtual disk image which is copied on the real disk of a computer.

As soon as the first boot takes place, the oemboot boot image repartitions the real disk, which it detects first, then it prepares the system to activate the YaST second stage process. Finally, the oemboot image boots into the system, and YaST starts with a special configuration sequence made for preinstalled systems.

The KIWI xenboot Image

The KIWI xenboot boot image works only with the Xen boot type. The Xen boot type creates all the information needed to run a Xen virtual machine with the previously created image, including the system image itself (which is a file containing the operating system data and the filesystem), the xenboot boot image as a compressed initrd file, the Xen kernel, and the Xen configuration file (which references the previously mentioned files). After KIWI finishes the image creation process, use the following command:

```
xm create -c xen-configuration-file
```

The primary task of the xenboot boot image is to load the required Xen modules, and to find out how the disk appears in the kernel. After this is done, the system can be activated.

The KIWI usbboot Image

The KIWI usbboot image works only with the USB boot type. The USB boot type creates all the files needed to install an operating system on a USB stick.

Use the following KIWI options for the installation:

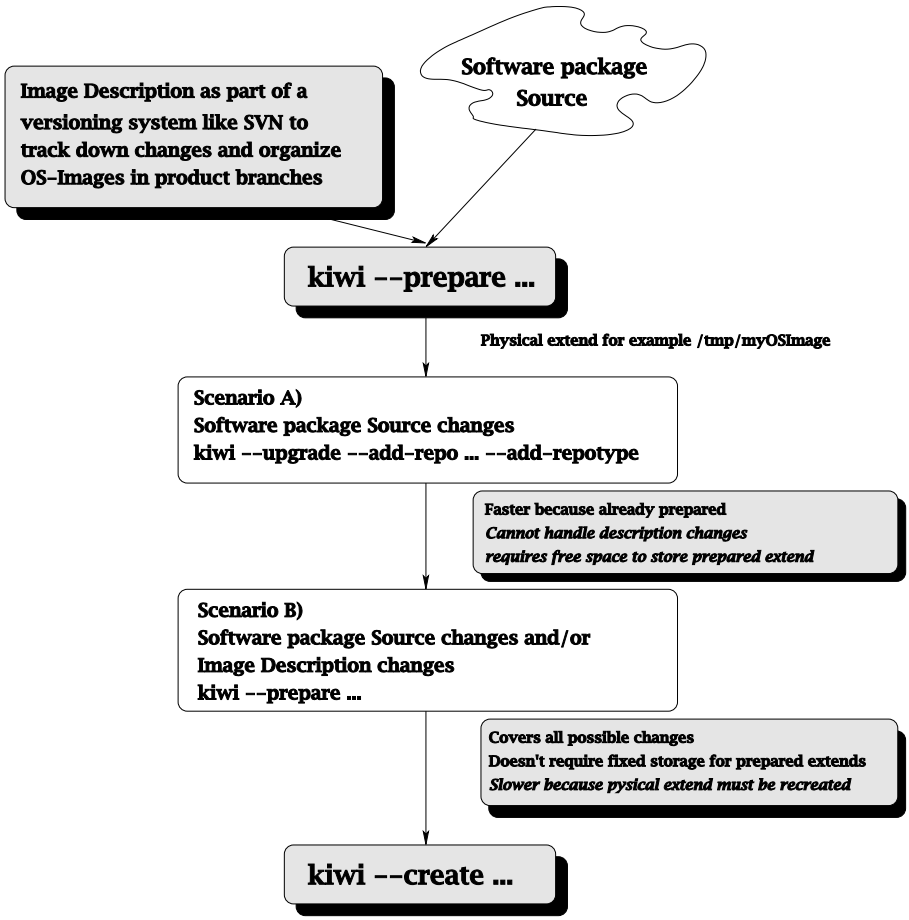
`--bootstick`, `--bootstick-system`

The files needed in this case include the system image itself and the usboot boot image (initrd). The primary task of the usbboot boot image is to find out how the BIOS mapped the USB stick. After the device is found, the system can be activated. KIWI also installs the boot manager to the USB stick so that each system can use the stick as the main operating system as soon as the BIOS can boot from a USB stick.

1.5. Maintaining Operating System Images

The creation of an operating system image usually results in a specific appliance solution for a particular customer or group of customers. But software develops, and you don't want your solution to become outdated. This section give you some idea on how to maintain images created by KIWI.

Figure 1.3. Image Maintenance Scenarios



The figure above illustrates two possible scenarios which would require you to update an operating system image.

- the section called “Software Changes”
- the section called “Configuration Changes or Enhancements”

Software Changes

The first reason for updating an image is changes to the software (for example, a new kernel should be used). If this change does not require additional software or changes to the system configuration, you can use the KIWI `upgrade` option to perform the update.

KIWI also lets you add an additional repository which might be needed if the updated software is not part of the original repository. It is important to note that this additional repository is *not* stored in the image description#s original `config.xml` file.

Configuration Changes or Enhancements

Another reason for updating a software image is configuration changes or enhancements (for example, an image replaced its browser with a better browser, or a new service such as Apache should be enabled). While it is possible to do these kind of changes manually within the physical extend, we don#t recommend it because it leaves the system in an unversioned condition, and no one would know what changes were made since the image was initially created.

Changes to the image configuration should be done within the image description. The image description itself should be part of a versioning system such as Subversion, which lets you keep track of changes and assign them

to product tags and branches. This means that images must be prepared from scratch and the old physical extend should be removed.

1.6. Real-Life Scenarios - A Tutorial

When you create an operating system image, you must decide how to activate the image on the target system. Because there are many possible target systems, the image deployment depends greatly on the system environment. For example, if a customer buys the SLEPOS product, the system environment is predetermined, so the he or she must follow the set rules to successfully deploy the image. This makes it easy for customers to control a complex system, but it also means that you lose the flexibility to use the same system in another environment.

KIWI doesn't require a specific system environment, and it does leave out some important tasks concerning the deployment architecture. So while KIWI is an image creator, it doesn't set up the deployment infrastructure. This chapter provides some examples of ways to deploy an image.

- the section called “Using the PXE Protocol to Deploy an Image via the Network ”
- the section called “Deploying a Split Image System via PXE”
- the section called “ Using an NFS Mounted `root` System to Deploy an Image via the Network ”
- the section called “Installing an Image from a CD, DVD, or USB Stick”
- the section called “USB Stick System”
- the section called “Using a Virtual Disk System (QEMU or VMware)”
- the section called “Using a Paravirtual Image with Xen”
- the section called “Using a Live CD System”

Using the PXE Protocol to Deploy an Image via the Network

PXE is a boot protocol generally implemented in the BIOS or boot ROM of network cards. When activated, it searches the network for a DHCP server from which it can obtain an IP address, and for information on where to find a TFTP server that can manage file transfers. If such a server exists, the second stage bootloader controls the subsequent boot process. Using PXE with KIWI requires the infrastructure explained in "The TFTP Server Structure," a TFTP server, and a DHCP server. KIWI provides the `kiwi-pxeboot` package, which sets up the boot structure and installs some prebuilt boot images.

If you plan on using a system image for which no prebuilt boot image exist, you must create a custom boot image before a system image can be deployed. Boot images consists of the image itself and the appropriate kernel for that image. Both file must be stored in the `/srv/tftpboot/boot` directory. Assuming there is no prebuilt boot image for the openSUSE 10.2 distribution, the steps to create it are as follows:

```
cd /usr/share/kiwi/image
kiwi --prepare netboot/suse-10.2 --root /tmp/myroot
kiwi --create /tmp/myroot -d /tmp
```

The result of this example is created in the `/tmp` directory:

```
ls -1 /tmp/initrd-netboot*
   /tmp/initrd-netboot-suse-10.2.i686-2.1.1.gz
   /tmp/initrd-netboot-suse-10.2.i686-2.1.1.kernel.2.6.18.2-31-default
```

If you don't want to prepare or create the boot image manually, you can let KIWI do the job by setting up the `pxe` type in your system image `config.xml` file. For example:

```
<preferences>
```

```
  <type filesystem="ext3" boot="netboot/suse-10.2">pxe</type>
  ...
</preferences>
```

In this case, the boot image is created automatically, using the same source as the system image, when the `\textbf{creation}` step is performed for the system image.

The next step is to make the boot image known to the TFTP server. To do this, the files must be copied to the `/srv/tftpboot/boot` directory. They can also be renamed. For the following example, the boot image is renamed to `initrd`, and the boot kernel is renamed to `linux`.

```
cp initrd-netboot.i686-2.1.1.gz /srv/tftpboot/boot/initrd
cp initrd-netboot.i686-2.1.1.kernel.2.6.18.2-31-default \
  /srv/tftpboot/boot/linux
```

Switching the target machine on, which must run PXE by default, loads the `linux` kernel and the KIWI-created `initrd`. The `initrd` registers the machine if no configuration is found in `/srv/tftpboot/KIWI`. “Registration” means that a file including the MAC address of the machine is uploaded into the `/srv/tftpboot/upload` directory. For information on creating the configuration, see <xref linkend="sec.kiwi.activating.netboot.configfile" xrefstyle="HeadingOnPage"/>. The following sample configuration is for a machine that has a disk on `/dev/sda`:

```
IMAGE=/dev/sda2;my-example-suse-10.2.i686;1.1.2;192.168.100.2;4096
PART=1024;S;x,x;L;/
DISK=/dev/sda
```

According to this configuration, the KIWI boot image tries to download a system image named `my-example-suse-10.2.i686-1.1.2` from the TFTP server with an IP address of 192.168.100.2. On the TFTP server, the system images are stored in `/srv/tftpboot/image`. The boot image prepares the disk, and creates a 1 GB swap partition and another full-sized linux partition. The process of creating this `my-example-suse-10.2` image can be done using KIWI. After the boot image has successfully downloaded the system image, it is activated and operates as configured.

Deploying a Split Image System via PXE

KIWI supports system images split into two parts: a read-only part, and a read-write part. This lets you put data on different filesystems that usually contain read-only data available on a compressed filesystem such as cramfs or squashfs. Almost all available compressed filesystems have some kind of restrictions which needs attention before you use it in an image.

To turn a system image into a split image, only the type of the image must be adapted. This information is part of the `config.xml` file, and can be edited as in the following example:

```
<preferences>
  <type filesystem="ext3,cramfs">split</type>
  ...
</preferences>
```

Creating an image from this description results in two image files. One of the files contains the `-read-write` extension in its name. Booting such an image always requires a boot process which must be able to bring both images together again. Because of this, split images can only be deployed in combination with one of the KIWI boot images.

To deploy the image, use PXE or a boot CD/USB-stick. The most important part of using a split images is the configuration for the target machine. For more information on this config.MAC file, see <xref linkend="sec.kiwi.activating.netboot.configfile" xrefstyle="HeadingOnPage"/>. When using a split image, the following information must be provided:

- The `IMAGE` key must contain both the read-write and the read-only image. The read-only image must appear as the first entry in the list.

- The `PART` key must specify a partition table with at least two partitions: an optional swap partition, and two system partitions which provide enough space for the first and second image portion.
- The `COMBINED_IMAGE` option, which tells the boot image to combine both images into one complete system.

The following example shows the configuration of a split image named `minimal-10.1 / minimal-10.1-read-only`:

```
IMAGE=/dev/sda2;minimal-10.1-read-only.i686;1.1.2;192.168.100.2;4096,\
      /dev/sda3;minimal-10.1.i686;1.1.2;192.168.100.2;4096
PART=200;S;x,500;L;/,x;L;
DISK=/dev/sda
COMBINED_IMAGE=yes
```

Using an NFS Mounted `root` System to Deploy an Image via the Network

KIWI was designed to upload an image onto a client in several different ways, including onto diskless machines. These devices don't provide permanent storage, and rely on the network. The most often-used process to activate such terminals is to NFS-mount the system image via the network. KIWI supports this as well, but it requires a terminal server configuration to export the system image using a NFS server.

To activate a diskless station, use the following steps:

1. Prepare the system image using the following command:

   ```
   kiwi --root /tmp/kiwi.nfsroot --prepare ...
   ```

2. Set up an NFS server which exports the `/tmp/kiwi.nfsroot` path.

   We recommend using the following export options in `/etc/exports`:

   ```
   /tmp/kiwi.nfsroot  *(rw,no_root_squash,sync,no_subtree_check)
   ```

3. Use KIWI to create an appropriate netboot boot image (`initrd`).

   “Appropriate” means that the package repository for the system image and the netboot image are the same.

4. Copy the boot image/kernel to the PXE server in `/srv/tftpboot/boot`.

   The `kiwi-pxeboot` package helps you set up the PXE/TFTP server.

5. Create a `config.<MAC>` file in `/srv/tftpboot/KIWI` with the following contents:

   ```
   NFSROOT=129.168.100.7;/tmp/kiwi.nfsroot
   ```

6. Boot the client.

   If everything works properly, the client receives the boot image and kernel via PXE/TFTP. The boot image NFS mounts the system image according to the data in config.<*MAC*>. After that, the mounted root filesystem is activated.

Installing an Image from a CD, DVD, or USB Stick

You can create a CD, DVD, or USB stick which contains an image you want to install on the hard disk of a computer. The installation process in KIWI is rather simple, but it doesn't consider previously installed operating systems or disk partitions, so be careful with this deployment method.

The process of booting a CD with an image as the content and the later first boot of this image is handled by the KIWI oemboot boot image. The system image in this case must be a virtual disk type, which requires oemboot to

be used in combination with the vmx image type. To indicate that the resulting virtual disk system image should be part of an ISO and the later install CD, set the attribute format with the value of `iso`. The system image description requires the following type specification:

```
<preferences>
  <type filesystem="ext3" boot="oemboot/suse-... format="iso">vmx</type>
  ...
</preferences>
```

To indicate that the resulting virtual disk system image should be part of a installation virtual disk which can be dumped on an USB stick, the value `usb` must be set.The system image description requires the following type specification:

```
<preferences>
  <type filesystem="ext3" boot="oemboot/suse-... format="usb">vmx</type>
  ...
</preferences>
```

Use the following KIWI commands to create an install ISO image:

```
kiwi --prepare /path/to/the/system/image/description --root /tmp/myRoot
kiwi --create /tmp/myRoot
```

The command creates a bootable ISO image which should be burned on a CD. KIWI informs the user about the file name to be burned on CD or DVD.

USB Stick System

It has become popular to store complete operating systems on a USB stick. All required system and user data is stored completely on the stick.

This tutorial explains how to create a bootable USB system on a concrete device. Several assumptions are made, and the process is described step-by-step. This section also includes an overview of known problems to help you avoid false positives.

- the section called “USB Assumptions”
- the section called “USB Workflow”
- the section called “Preparing `config.xml` for USB”
- the section called “Preparing the Installation Source Directory”
- the section called “Modifying the Installation Source Directory”

USB Assumptions

- A bootable USB image is the result.
- Either openSUSE or a compatible Linux derivative is available on the build host.
- The latest version of KIWI is installed, with the following required packages:
  - `kiwi`
  - `kiwi-desc-isoboot`
  - `kiwi-desc-livesystem`

You can download and install these packages from the openSUSE BuildService, or you can obtain the latest svn snapshot from svn.berlios.de.

1. Check out the latest version from svn.berlios.de.

   ```
   svn co https://anonymous@svn.berlios.de/svnroot/repos/kiwi/kiwi-head
   ```

   Replace *anonymous* with a valid BerliOS account if one is available.

2. Make sure that the `kiwi-desc-isoboot`,`kiwi-desc-livesystem`, and `kiwi-desc-usbboot` packages are installed.

   ```
   rpm -qa kiwi*
   ```

3. Cd to the kiwi checkout directory (where the makefile resides), then execute the following:

   ```
   make && [sudo] make install
   ```

   `Root` privileges are required to perform the install step because files are copied to `/usr/share/kiwi` and `/usr/sbin`. It might be necessary to add the sudo command prefix if the previous commands were run as a normal user.

4. Create a working directory (recommended in `$HOME`), for example `configs`, and check out the image descriptions from forgesvn1.novell.com.

   ```
   svn co https://forgesvn1.novell.com/svn/opensuse/trunk/distribution/images configs
   ```

After this procedure, KIWI is installed as `/usr/sbin/kiwi`, and the image descriptions in `/usr/share/kiwi/images/`. These subdirectories contain `config.xml` files, which are used by the `kiwi` command internally, and must not be modified for any reason. If `kiwi` is installed from the subversion repository, it will not show up as an installed package. The rpm files contain dependencies that must be resolved manually. A list of dependencies is collected in the `README` file in the base directory of the KIWI repository.

USB Workflow

The main workflow consists of three consecutive commands shown in the following example:

Example 1.1. Basic kiwi Commands

```
kiwi -r <rootdir> --prepare <imagedir>
kiwi --create <rootdir> --type usb -d <outputdir>
kiwi --bootstick <outputdir>/initrd*.gz \
     --bootstick-system <outputdir>/<imagefile>;
```

In this example, the terms in braces (< >) have the following meaning and requirements:

Explanation of Options

rootdir
: Specifies the directory where the installed system is created. This directory is created by the `kiwi` command. If it existed previously, before, `kiwi` exits with a warning message.

imagedir
: Specifies the location of your edited `config.xml` file.

outputdir
: Specifies the location of the following generated output files:

Result Files of Second Stage

- `initrd-*.gz`
- `initrd-*.kernel`

- initrd-*.kernel.<*version*>;<*type*>
- initrd-*.md5 -- the MD5 checksum
- initrd-*.stickboot
- <*imagefile*> as named in the first line of config.xml

  For example: USB-Image-suse-10.3-Alpha5-Plus-<arch>-<version>
- imagefile's md5 sum

There are several areas where the user can modify the image creation. These areas are explained in the respective sections where the interaction makes sense.

Preparing config.xml for USB

Several changes should be made to the config.xml file delivered with the kiwi-desc-livesystem package. For example, the correct installation source must be used. In SuSE internal builds, the FACTORY tree is used. If FACTORY is currently rebuilt or in an unsynchronous state, you can use another installation source, such as /mounts/dist/install/SLP/openSUSE-10.3-LATEST-DVD/i386/DVD1.Unfortunately, there are no squashfs and aufs packages in this tree, so you will have to import these packages from another location.

If necessary, you can declare multiple installation sources. Any folder containing multiple RPM files that must be installed additionally (which are not already contained in a repository) can be added as a repository. A sample configuration file is shown here:

Example 1.2. Example USB Configuration File

```
<image name="USB-Image-suse-10.3-Alpha5plus">
 <description type="system">
  <author>Jan-Christoph Bornschlegel</author>
  <contact>jcbornschlegel@novell.com</contact>
  <specification>openSUSE 10.3 USB boot system</specification>
 </description>
 <preferences>
  <type primary="true" boot="isoboot/suse-10.3" flags="unified">iso</type>
  <type boot="vmxboot/suse-10.3" filesystem="ext3" format="vmdk">vmx</type>
  <type boot="xenboot/suse-10.3" filesystem="ext3">xen</type>
  <type boot="usbboot/suse-10.3" filesystem="squashfs">usb</type>
  <version>1.1.2</version>
  <size unit="M">2000</size>
  <packagemanager>smart</packagemanager>
  <rpm-check-signatures>False</rpm-check-signatures>
  <rpm-force>True</rpm-force>
 </preferences>
 <users group="users">
  <user name="linux" pwd="..." home="/home/linux"/>
 </users>
 <repository type="yast2">
  <source path="opensuse://SL-OSS-factory"/>
 </repository>
 <repository type="rpm-dir">
  <source path="[some plain RPM dir]"/>
 </repository>
 <packages type="image" patternType="plusSuggested">
  <package name="subversion"/>
  <package name="xkeyboard-config"/>
  <package name="vim"/>
  <package name="yast2-schema"/>
```

```
  <package name="yast2-theme-openSUSE"/>
  <package name="yast2-control-center"/>
  <package name="yast2-control-center-qt"/>
  <package name="yast2-live-installer"/>
  <opensusePattern name="default"/>
  <opensusePattern name="base"/>
  <opensusePattern name="enhanced_base"/>
  <opensusePattern name="x11"/>
  <opensusePattern name="yast2_basis"/>
  <opensusePattern name="yast2_install_wf"/>
  <opensusePattern name="apparmor"/>
  <opensusePattern name="imaging"/>
  <opensusePattern name="kde"/>
  <opensusePattern name="kde_basis"/>
  <opensusePattern name="kde_imaging"/>
  <opensusePattern name="office"/>
  <ignore  name="ash"/>
  ...

  <ignore  name="krb5-32bit"/>
 </packages>
 <packages type="xen" memory="512" disk="/dev/sda">
  <package name="kernel-xen"/>
  <package name="xen"/>
 </packages>
 <packages type="vmware" memory="512" disk="ide0">
 </packages>
    <packages type="boot">
        <package name="filesystem"/>
        <package name="glibc-locale"/>
  <package name="kernel-default"/>
        <package name="devs"/>
    </packages>
</image>
```

First, the image needs a name, which sets the filename mentioned in the section called “USB Workflow” as <imagename> (line 1). Then set the author and contact information. Lines 9-11 set the respective configurations for the desired image type. To use a specific type, the folder `/usr/share/kiwi/image/<type entry>` must already exist. You might have to check out additional modules or install additional `kiwi-desc-<type>` packages.

The `version` tag in line 12 is added to the image's filename. The `size` option mentioned in line 13 is the size to which the stick's filesystem is expanded at first boot. The additional installation source (in this case, a plain directory containing some RPM files) is added in lines 28-20.

The majority of changes are made in the <package name="..."/> section. The `<package name="something"/>` tag adds the <something-*.rpm> package to the installation, where "*" is a mix of version number, release number and architecture. The `<opensusePattern name="somepattern"/>` tag adds all packages required by the <somepattern> pattern to the installation. The `<ignore name="someignoredpackage"/>` tag omits the <someignoredpackage> package. This is an ugly method used to resolve pattern bugs. It happens when a pattern requires a package that cannot be resolved because it does not exist in the given installation sources (or the name changed). If this happens, the `prepare` step must be repeated (see the section called “Preparing the Installation Source Directory” for more information).

Preparing the Installation Source Directory

The prepare step collects all packages listed in the `config.xml` file and all packages required by the included patterns. Use the `-r` option to specify the target directory which is created by `kiwi`. An error message is reported if the directory already exists.

This step reveals all pattern problems and missing packages. Unfortunately, these issues must be fixed manually using the techniques described below. Unresolved dependencies can be a severe problem because it shows that some package should be in the installation source but are not. In that case, several solutions are possible:

- Find the package somewhere else and put it in using the `<package name="..."/>` tag.
- If a lot of packages are missing, and all of them can be found in the same repository, use the `<repository type="..."> ...</repository>` tags.
- The packages can be stored in a local directory, and that directory can be declared as an additional installation source of the type "rpm-dir" (plain RPM directory). In that case, the line looks like this:

```
<repository type="rpm-dir">
<source path="/some/directory/where/be/RPMS/"/>
</repository>
```

Modifying the Installation Source Directory

After the `--prepare` step is successfully completed, the installation source can be modified further. You can always perform a `chroot` command, then install packages using `smart` or `rpm`. The rpms must be accessible through the available channels, which means that you can add anything listed in the repositories declared in the `config.xml` file, or you can copy (or hardlink) them somewhere in the chroot environment before you chrooting there.

Be aware that the `smart` command cannot resolve certain dependencies. For example, if you have the packages `yast2-control-center` and `yast2-control-center-qt` copied or linked to `<rootdir>;/tmp`, you can install by calling the following commands:

```
cp /some/directory/where/be/RPMS/yast2-* <rootdir>/tmp
chroot <rootdir>
smart install /tmp/yast2-control-center /tmp/yast2-control-center-qt
```

If you install only `yast2-control-center-qt`, `smart` will not find `yast2-control-center` although it is required. This is why it is necessary to specify both packages if you use `smart`.

All of the repositories specified in the configuration file are added as `smart` channels, and any package within these channels can be manually installed after the prepare step if necessary. You can also uninstall packages if the image becomes too large.

Creating the USB Images

The images can be created as soon as everything is prepared in the chroot environment. Simply run the second command in Example 1.1, “Basic kiwi Commands”. The target directory <outputdir> must be manually created before, in contrast to the prepare step where `kiwi` creates the directory. Some of the "failed" messages can be safely ignored because they reflect known bugs and do not break the image creation. The files created are listed in Result Files of Second Stage.

You can create images for different architecures in the same target directory, because the filenames include the architectue.

Dump the USB Images on the USB Device

As the images are eventually created they can be dumped on the USB device. Some things must be considered before:

Considerations before Dump

- Any data on the stick before will be lost
- It is not possible to save a partition

`kiwi` will request the device file after it computes a list of possible devices. All connected USB "writable" devices should be listed; it is recommended to verify using for example `dmesg`. After the correct device file was entered the filesystem on the stick is created. This may take several minutes depending on the stick's write speed. There is a mathod for checking the progress. `Kiwi` uses `dd` to write the image on the device, so you can send the respective process a signal `USR1`. As result `dd` prints its current statistics to `stdout` and continues normally.

Example 1.3. How to Retrieve `dd` status

```
cd newimage
kiwi -bootstick initrd-usbboot-suse-10.3.i686-2.1.1.gz \
-bootstick-system USB-Image-suse-10.3-Factory.i686-1.1.2
(other shell)
killall -USR1 dd
```

Known Issues

There have been tests with mini-harddisk USB devices which showed some problems on some machines. Some problems arise trying to boot a laptop with special input devices (IBM ThinkPad with its joystick device for example), touchpad support is still a missing item.

Another known problem is the workaround time to fix missing package dependencies. It happened with some installation sources that several packages were missing. It is also strongly recommended not to modify the repositories themselves because then the metadata becomes incorrect and must be recreated. This is possible, but not very convenient.

Conflicts may arise depending on the used `smart` version, especially in one particular case: The host system's `smart` is used for the first stage of the image creation, the changeroot environment. For the second step the version installed in the changeroot environment is used. This leads to problems if the host's version is newer and can handle `gzip` compressed metadata and the installed version can not.

OEM Boot System

OEM Assumptions

This tutorial explains how a preload harddisk image is created with `kiwi`. A harddisk image allows hardware vendors to deploy a preconfigured system with the hardware to achieve convenient use and cost effective installation at the same time.

The image will be prepared and configured once and then deployed to many computers using a simple copy.

In addition to the packages and preconditions mentioned in the section called “USB Assumptions” one more package must be installed which contains the OEM boot configuration and scripts: kiwi-desc-oemboot. Alternatively the files can also be obtained from the kiwi subversion repository. If you decide to use the repository version of `kiwi` the folder is already checked out.

OEM Workflow

The workflow is comparable to the USB workflow as described in the section called “USB Stick System” but involves only two `kiwi` invocations. Deployment is done by any copy mechanism, for example as simple as `dd`.

Example 1.4. Basic kiwi Commands for OEM

```
kiwi -r <rootdir> --prepare <imagedir>
kiwi --create <rootdir> -d <outputdir>
```

The files in brackets have exactly the same meaning as explained in the section called “USB Workflow”, the resulting files are different:

Result Files of Second Stage OEM

- `initrd-oemboot-*.gz`
- `initrd-oemboot-*.kernel`
- `initrd-oemboot-*.kernel.<version>;<type>`
- `initrd-oemboot-*.md5 -- the MD5 checksum`
-
- `<imagefile>` as named in the first line of `config.xml` plus architecture and version number

  For example: `preload-suse-SLED10.i686-1.1.2`
- imagefile's md5 sum
- The image as dumpable file (extension .raw)

  For example: `preload-suse-SLED10.i686-1.1.2.raw`

The ".raw" file can be dumped to the harddisk of the target system directly. For distribution, download or a recovery disk the image can be shrinked using `gzip`. In test environment we used the following method for deployment:

Procedure 1.1. Deploying the OEM Image on Test Machine

1. Boot the test machine from any device but harddisk.

   This is a very beautiful use case for the USB boot image explained in the section called “USB Stick System”.
2. Create a NFS export on the machine which hosts the image.

   Edit the file `/etc/exports` and add a line like this:

   ```
   /folder/on/host *.>domain>(ro,root_squash,sync)
   ```

   Restart (or start) the NFS server:

   ```
   [sudo] rcnfsserver restart
   ```

   Additionally it may be necessary to add the NFS port in your firewall configuration.
3. mount the exported diretory on the test machine:

   ```
   mount host:folder mountpoint
   ```
4. Dump the image to the harddisk using the device file.

   ```
   dd if=mountpoint/imagefile of=/dev/sda
   ```

   In case of a compressed image these commands must be:

   ```
   zcat mountpoint/imagefile.gz | dd of=/dev/sda
   ```

OEM Image Known Issues

The workaround time is rather long if the image has a lot of packages installed. The *imagename*`.raw file` becomes huge if you set a size in the `config.xml` file. If you comment this line, `kiwi` determines a size automatically which is rather small. Even with this it is still possible to compress the image and save another 30-50% of the size which allows the image to fit on a single layer DVD for instance for recovery disks.

Using a Virtual Disk System (QEMU or VMware)

To use a virtualized system, you must first create a virtual disk. You can do this by specifying the following `type` in the system's image `config.xml` file:

```
<preferences>
  <type filesystem="ext3" boot="vmxboot/suse-10.2">vmx</type>
  ...
</preferences>
```

After you do this, you can create the system image. The process creates the system image and the specified boot image (`vmxboot/suse-10.2` in our example). The result is then used to create the virtual disk. To create the system image, use the following command:

```
kiwi --prepare my-example-suse-10.2 --root /tmp/mysystem
kiwi --create /tmp/mysystem -d /tmp
```

The result of this command is a set of virtual disks, one with a `.qemu` suffix (QEMU), and one with a `.vmdk` suffix (VMware). To run the system on the virtual disk (with qemu as the example), use the following:

```
qemu /tmp/my-example-suse-10.2.i686-1.1.2.qemu
```

Using a Paravirtual Image with Xen

To use an image within Xen, create a system image and an `initrd` file. You can do this by specifying the following `type` in the system's image `config.xml` file:

```
<preferences>
  <type filesystem="ext3" boot="xenboot/suse-10.2">xen</type>
  ...
</preferences>
```

After you do this, you can create the system image. The process creates the system image and the specified boot image (`xenboot/suse-10.2` in our example). An appropriate Xen configuration file, with a `.xenconfig` suffix, is also created. To create the system image, use the following command:

```
kiwi --prepare my-example-suse-10.2 --root /tmp/mysystem
kiwi --create /tmp/mysystem -d /tmp
```

To run the system within Xen, use the following command:

```
xm create -c /tmp/my-example-suse-10.2.i686-1.1.2.xenconfig
```

Using a Live CD System

LiveDistro or Live CD is a generic term for an operating system distribution that is executed upon boot, without installation, on a hard drive. Typically, it is stored on a bootable medium, such as a CD-ROM (Live CD) or DVD (Live DVD). The term "live" comes from the fact that these distributions are a complete, runnable (that is, "live") instance of the operating system residing on the distribution medium, rather than the typical collection of packages that must first be installed on the target machine before using the operating system.

A LiveDistro does not alter the current operating system or files unless the user specifically requests it. The system returns to its previous state when the LiveDistro is ejected and the computer is rebooted. It does this by placing the files that are typically stored on the hard drive into temporary memory, such as a RAM disk. In fact, a hard drive is not needed at all. However, the lack of a hard drive cuts down on the RAM available to applications, which can reduce performance.

To create an `.iso` image that can be burned on CD, specify the boot image that should handle your live system. You can do this by setting the `type` of the image in the `config.xml` file as follows:

```
<preferences>
  <type boot="isoboot/suse-10.3">iso</type>
  ...
</preferences>
```

The `boot` attribute specifies the CD boot image which must exist in `/usr/share/kiwi/image/isoboot`. As with all boot images, the most important point is that the boot image must match the operating system image. This means that the kernel of the boot and operating system image must be the same. If you don't have a boot image that matches your operating system image, you should create your own boot image description. You can use an existing boot image and adapt it to your needs.

Appendix A. GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

A.1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

A.2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

A.3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

A.4. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a

computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

A.5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

A.6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

A.7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

A.8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

A.9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

A.10. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

A.11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.12. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.