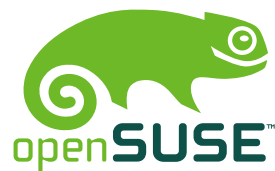# openSUSE - KIWI Image System

System Design

openSUSE - KIWI Image System: System Design

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

Published September 06, 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

Table of Contents

List of Figures

Chapter 1. Image Building with KIWI

Marcus Schäfer
Frank Sundermeyer

Basic structure only
$Revision: 212 $
$Date: 2007-02-22 17:21:45 +0100 (Thu, 22 Feb 2007) $

TODO: Abstract

1.1. Introduction

The openSUSE KIWI Image System provides a complete operating system image solution for Linux supported hardware platforms as well as for virtualisation systems like Xen, VMware and others. The KIWI architecture is designed as a two stage system. The first stage, based on a valid *software package source*, creates a so called *physical extend* as provided by an image description. The second stage creates an operating system image from the physical extend . The result of that second stage is called a *logical extend* or operating system image.
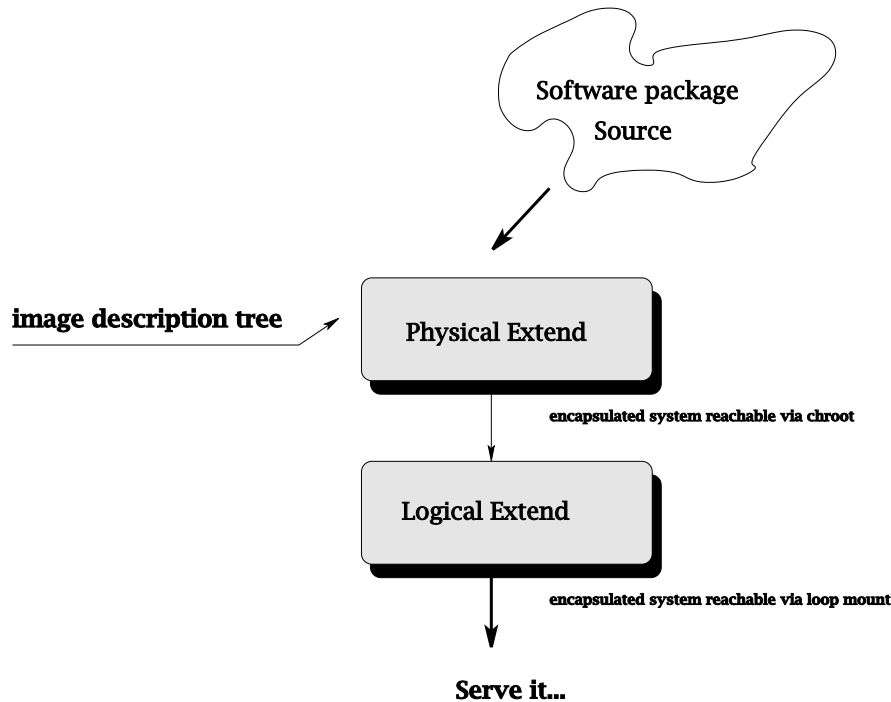
Concept

The first stage created by KIWI, the physical extend, requires at least one valid software package source, a so-called repository, in order to access the software to build a system. A repository consists of software packages, organized in a package tree that also includes some meta data. Software repositories can exist in different formats, therefore KIWI uses a Package-Manager to access them. With KIWI you have the choice for either *smart* or *zypper* to be used as package manager. Smart handles a wide range and also the most important repository formats. More information on smart is available at `http://labix.org/smart`.

The second stage—creating an operating system image—takes place without user interaction. Therefore all the necessary information needs to be created prior to the image building process. An image description tree stores all these information needed to create an image (see Section 1.1, "Introduction" for details).

Operating System Images

A regular installation process is starting from an installation image and installs single pieces of software until the system is complete. Normally such an installation process is interactive - the user is able to alter installation settings and configuration options. Contrary to that, an operating system image represents an already completed installation encapsulated as a file and optionally includes the configuration for a specific task. Such an operating system starts working as soon as the image has been installed to a system storage device (no matter if this is a volatile or non volatile storage device). An operating system image is deployed "as is"—no user interaction is possible.

Figure 1.1. Image Serving Architecture

Because this document contains conceptual information about an image system, it is important to understand what an operating system image is all about. A normal installation process is starting from a given installation source and installs single pieces of software until the system is complete. During this process there may be manual user intervention required. However an operating system image represents an already completed *installation* encapsulated as a file and optionally includes the configuration for a specific task. Such an operating system starts working as soon as the image has been brought to a system storage device no matter if this is a volatile or non volatile storage. The process of creating an image takes place without user interaction. This means all requirements of the encapsulated system has to be fulfilled before the image is created. According to this the so called *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 represents an operating system as part of a specific filesystem which could also be covered by the structures of real or virtual hardware layers. There are different types of images whereas KIWI supports the following:

The Live-System image [iso]
    The iso type is used to create live systems. A live system is an operating system on a CD or DVD. When the system boot 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 shut down the data will be lost. The generated iso file needs to be burned on the media

The virtual hard disk image [vmx]
    The vmx type is used to create a virtual disk. The disk provides the partition information the boot manager and all other data which are found on a real disk as well. Such an image can be used as disk for full-virtual systems like Qemu or VMware. KIWI also creates the VMware configuration file if requested. The generated image requires a virtualisation software to be installed. In a full virtualized all components are virtualized. This includes the storage devices as well as the processor and all other parts of the system. To activate a virtual disk system system the user only needs to call the correct "player" application which in case of QEMU is `qemu` and in case of VMware `vmplayer` is used.

The para-virtual Xen image [xen]
    The xen 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 one

can create new para virtual machines with the data provided by KIWI. The generated image requires a Xen hypervisor running on the computer.

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

The network image [pxe]
> the pxe type is used to create an operating system image based on a given filesystem including a special network boot image. The image itself will be stored on a server from where the boot image can download and activate it. The generated image requires a DHCP/TFTP network infrastructure running on a server

The network split image [split]
> the split type is used to create an operating system image based on two given filesystems. The image will be divided into two portions whereas the first portion represents the data which requires read/write access and the second portion represents the read-only data. According to this the second porition can be part of a compressed read-only filesystem. This image type doesn't automatically create its boot image and works only with netboot and xennetboot 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 influences the operating system environment like the used filesystem but the result is still an image made for the purpose described by the base type.

1.2. Creating Operating System Images

When starting to create an image with KIWI it's required to create a so called *system image description*. Such a description is a directory containing at least one file named *config.xml*. The config.xml describes what packages/ patterns should make your image from which source repositories the data should be obtained which image type(s) can be generated from this description and many more (see Section 1.3, "The KIWI image description").

In order to be able to start with an example image one can use an existing system image description as template. An image description is normally provided as architecture independent package. For your reference download the *kiwi-desc-livesystem* from here: `http://download.opensuse.org/repositories/openSUSE:` `/Tools`. This package contains descriptions for openSUSE live systems which can serve as basis for any other image type as well.

The process of creating an image always start 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 first step is an operating system which has its root directory below the given destination directory, /tmp/myroot. After this step the new root directory serves as data source for creating the requested operating system image. This step is called the *create* step and requires at least the path to your previously created new root environment plus a destination directory for the image files as arguments:

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

The destination directory /tmp/myimages must exist and KIWI will store all file for the requested image type below it. KIWI allows to specify *defaultroot* and *defaultdestination* attributes as part of the config.xml. If these are present and no --root and/or --destdir options are specified the information from this default attributes are used.

KIWI does not only create your system image it also provides a subsystem for the image deployment. Each system image somehow needs to be activated. For example the system image for a network client needs to be downloaded from a server, it must be installed, maybe a disk needs to be partitioned, etc. All the steps 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 and furthermore KIWI creates them automatically while the *create* step is called. People who are used to Linux may know the name *initrd*, the KIWI boot images are special initrd images concerning 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 there use the naming scheme *type*boot. Concerning this the boot images for network images are stored in the directory netboot/ and the ones for the virtual disk images are stored in vmxboot/ and so on. The boot images shouldn't require any change they are a service to the customer, automatically created to provide a maximum of comfort to the customer. For detailed information of the boot images see the section called "`config`".

1.3. The KIWI image description

The creation of operating system images from a physical extend is based on image description trees. An image description tree contains a set of files in a certain directory structure that are required to generate an image using `KIWI`. An image description tree is organized as follows:

```
|
|- 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`

Optional Subdirectory that contains shell scripts that are executed after all packages have been installed, for example to 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`

An optional directory needed when creating a bootable CD. It contains files required by the isolinux boot loader. This includes the `isolinux.cfg,isolinux.msg` configuration files and the `isolinux.sh` build script. It creates an ISO image from a prebuild CD tree based upon the configuration from `isolinux.cfg`.

`config.sh`

Optional configuration script while creating the physical extend. This script is executed at the end of the installation when having switched to the operating sstem image with `chroot`, but *before* the package scripts paced in the config directory (see the section called "`config`") have run. It is used to configure the image system by, for example, activating or deactivating services.

`images.sh`

Optional configuration script executed at the beginning of the image creation process when creating the logical extend. It cleans the image system from programs and files only needed while the physical extend exists.

`config-yast.xml`

Optional AutoYaST configuration file. To generate such a file, check *Clone This System for AutoYaST* during the installation of openSUSE. This creates a ready-to-use profile as `/root/autoinst.xml` that can be used to create clones of this particular installation. To create an autoinstallation file from scratch or to edit an existing one, use the YaST module *Autoinstallation*.

In order to use `/root/autoinst.xml` move it to the images description tree and rename it to `config-yast .xml`. KIWI will process the file and setup your image as follows:

- When booting the image YaST is automatically started in AutoYaST mode

- The system is configured by YaST by applying the rules from `config-yast.xml`

- If the process finishes successfully the environment is cleaned and AutoYaST will not be started with next reboot.

`config-cdroot.tgz`

Optional compressed tar archive which is used for live systems only. The data in the archive will be uncompressed and stored in the CD/DVD root directory. The archive could be used to integrate a license or readme information on the CD or DVD

`root`

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

`config.xml`

The main configuration file, defining image type, base name, repositories, profiles, options, and the package/ pattern list.

Note

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

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

Such a script is called within the image environment, which means it is not possible to damage the host system with it script even if you are using absolute paths

`<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="...">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>
  <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" profiles="Name">❾
  <package name="Packagename" arch="Arch"/>
  <opensusePattern name="Patternname"/>
  <ignore name="Packagename"/>
</packages>
</image>
```

❶   The `image` element contains the attribute `name` which spcifies the base name of the image. It is automatically expanded with the version number and the current date. The version number is extracted from the directory in which the description files for this image are located. When using the optional attribute `inherit` with a path to another KIWI description, this description will be prepended to the current one. The mandatory attribute `schemeversion` must be set and allows version *1.4* at the moment.

❷   `description` element contains the attributes `author`, `contact` and `specification`. Author should be the name of the responsible person for this image. Contact should be a valid e-mail address in order to get in touch with the responsible person and the specification attribute contains a free form text describing what this image is good for.

❸   The `preferences` element contains information needed to create the logical extend. The following sub-elements are defined:

type
>   The image type of the logical extend. When specifying multiple entries, the additional attribute
>   `primary` setting the primary type needs to be filled in, otherwise the first entry of the `type` value
>   is used.
>
>   The following values for `type` are valid:
>
>   a. `ext2`, `ext3`, `reiserfs`, `squashfs`, `cpio`
>
>   b. `iso`, `split`, `usb vmx`, `xen`, `pxe`
>
>   The second group of types requires additional attributes:

primary
>   attribute to specify the primary type. The `KIWI` option `--type` allows to select between the types

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

flags
>   attribute to specify flags for the image type. Currently only the `compressed` flag for the `iso` type
>   exists. This flag causes the live media to be based on a squashfs compressed file system.

filesystem
>   attribute to specify the filesystem. Could be set to `ext2`, `ext3`, `reiserfs`, `squashfs` or `cpio`

The following list is an overview of different boot image (`initrd`) types. While creating the system
image KIWI automatically creates the specified ...*boot* boot 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 attribute `flags` is set to `compressed` squashfs compression will be used for the
>   read-only part of the iso. If set to `unified` the squashfs compressed read-only part will be used
>   in combination with the union filesystem named aufs which allows a complete read-write system
>   as long as the system runs.

`<type boot="usbboot/suse-..." filesystem="ext3">usb</type>`
>   The attribute `filesystem` specifies one of `ext2,ext3,squashfs` or `reiserfs` The created
>   system and boot images are suitable to run on an USB stick and can be deployed to it with the KIWI
>   `--bootstick` / `--bootstick-system` options

`<type boot="vmxboot/suse-..." filesystem="ext3" format="...">vmx</`
`type>`
>   The attribute `filesystem` specifies one of `ext2,ext3`, or `reiserfs`. The final result is one
>   file appearing as virtual disk which includes the system and boot images as well as the disk geometry
>   partition information and the boot manager. The attribute `format` specifies one of `vvfat`, `vpc`,
>   `bochs`, `dmg`, `cloop`, `vmdk`, `qcow`, `cow` or `iso`. Except the iso format these are all virtual disk
>   formats supported by the Qemu disk tool named qemu-convert. The iso format has a special meaning
>   and makes only sense in combination with an *oemboot* boot image. Using this format will generate
>   a so called install CD/DVD from the created virtual disk image. In order to be able to install the
>   virtual disk system from a CD onto a real hard disk of a computer the oemboot boot image must
>   be used. The process will create a bootable ISO image including the virtual disk image and the
>   oemboot boot image. The oemboot image's first task is to install the virtual disk image file onto the
>   real hard disk using the *dd* command. As the virtual hard disk contains a boot manager the newly
>   installed system is able to boot after the installation. The oemboot image's second task is to re-

partition the disk according to its real disk geometry, install the distribution default initrd and boot into the final system.

`<type boot="xenboot/suse-..." filesystem="ext3">xen</type>`
: The attribute `filesystem` specifies one of `ext2`,`ext3`, or `reiserfs`. The created system and boot images as well as the xen configuration file are suitable to run a Xen virtual machine using the `xm` program.

`<type boot="netboot/suse-..." filesystem="ext3">pxe</type>`
: The attribute `filesystem` specifies one of `ext2`,`ext3`, or `reiserfs` and the value of `boot` is the path to a netboot or xennetboot boot image description. The created system and boot images are suitable to drive a network client station. An appropriate network infrastructure including 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 attribute `filesystem` specifies a filesystem pair whereas *`type-rw`* specifies one of `ext2`, `ext3` or `reiserfs` and *`type-ro`* specifies one of `ext2`, `ext3`, `reiserfs`, `cramfs` or `squashfs`. Booting split images is only supported by the netboot and xennetboot boot images which need to 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 where no packages have been added or removed

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

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

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

> ## Note
>
> KIWI supports the feature of extending the image size automatically if the specified value is too small. If the actual size is more than 100MB larger than the spcified one, KIWI will abort with an error message. On the other hand, KIWI does *not* automatically reduce the images size if the specified value is too large, because the extra space may be needed to, for example, run custom scripts. If no size is specified at all KIWI will use the required size plus approximately 10% free space.

`packagemanager`
: Name of the packagemanager to be used for package installation. Currently `smart` is the default packagemanager but `zypper` is supported as well.

`compressed`
: The compressed flag indicates 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 living in the image. For network clients it makes sense to download the image as compressed image.

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

timezone
>   The time zone. Available time zones are located in the directory `/usr/share/zoneinfo`. Specify the attribute value relative to `/usr/share/zoneinfo`— e.g. `Europe/Berlin` for `/usr/share/zoneinfo/Europe/Berlin`. KIWI uses this value to set up the timezone in `/etc/localtime` for the image.

locale
>   Contains the name of the locale to use and therefore defines the contents of the `RC_LANG` system environment variable set in `/etc/sysconfig/language`

defaultdestination
>   Optional attribute `defaultdestination` is used if the option `destdir` is not specified while calling KIWI.

defaultroot
>   Optional attribute `defaultroot` is used if the option `root` is not specified while calling KIWI.

❹ The optional `profiles` element contains information which allows you to maintain one image description while still allowing for some variation in the set of packages and drivers that are included. A `profile` element has to be specified for each variation. The child element `profile` with the attributes `name` and `description` specifies an alias name which is used in sections to mark them as belonging to a profile and a more detailed description explaining what this profile does. To mark a set of packages and/or drivers as belonging to a profile, simply annotate them with the `profiles` attribute. If a `packages` tag has no `profiles` attribute, it is assumed to be present for all profiles. All of the above also goes for the `drivers` tag too.

❺ The optional `users` element contains the users to be added to the image. The attribute `group` specifies the group the user(s) belongs to. If this group doesn't exist it will be created. A `user` element has to be specified for each group. Specify the users belonging to a group with the child element `user` with the attributes `name`, `pwd` and `home` for username, password and the path to the home directory.

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

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

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

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

❼ The `repository` element defines the source path and type used by the package manager. The attribute `type` specifies the type of the repository, for example, `type="yast2"`. The child element `source` contains the attribute `path` to setup the the location of the repository, for example, `source="/image/CDs/full-i386"`. The path specification can be one of:

- local path starting with `/`

- `this://` relative path name which is relative to the image description which is referenced

- `http://` or `ftp://` Network-Location

- opensuse://Project-Name

- The path can include the %arch macro if needed

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

❽ The `deploy` element make sense for network clients only and describes where to get the system image and how should the client be prepared for that image. Preparation in the sense of an image means how should the disk be partitioned or what configuration files should be included. According to this the attributes `server` and `blocksize` specifies the TFTP server which takes over control for the download. The `partitions` tag specifies the partition(s) for one disk device (`device`). Each partition is specified by one subtag named `partition` which defines the type (see sfdisk --list-type), partition number, size , optional mountpoint and optional information whether this partition is the system image target partition or not. With the KIWI netboot image the first partition is always the swap partition and the second partition is by default used for the system image. With the optional target flag it is possible to indicate another than the second partition to be the one the system image is installed on. If *size* is set to "image" KIWI will calculate the required size for this partition in order to have enough space for the later image. The optional `union` is used if the system image is based on a read-only filesystem like squashfs. In this case KIWI can setup an aditional write partition and combine both with the given overlay filesystem. Right now there are two of such filesystems one is called *unionfs* and the other (preferred) one is called *aufs*. The partition which holds the read-only system image must be set as value to the attribute `ro` and the partition which should serve as write partition must be set as value to the attribute `rw`. The optional `configuration` tag can be used to integrate configuration files to a network client which are remotely stored on the server. The attribute `source` specifies the path on the server used by a tftp client program to download the file and the attribute `dest` specifies the target relatively to the root (/) of the network client.

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

image
> Packages used to finish the image installation. All packages which make up the image are listed there.

boot
> Packages used to start creating a new operating system root tree. Basic components which are required to chroot into that system like `glibc` are listed here.

xen
> Packages used when the image needs support for Xen based virtualisation. The attributes `memory` and `disk` defines how much memory the virtual system requires and what kind of device the disk should appear in the virtual instance. This information is used to create the appropriate Xen configuration file.

vmware
> Packages used when the image needs support for VMware based virtualisation. The attributes `memory` and `disk` defines how much memory the virtual system requires and what kind of device the disk should appear in the virtual instance. This information is used to create the appropriate VMware configuration file.

delete
> Packages stored for later deletion. The package names are available in the variable `$delete` of the .profile created by KIWI. The function *baseGetPackagesForDeletion()* returns the contents of this environment variable and can be used to delete the packages without taking care for requirements or dependencies.

Using a pattern name will enhance the package list with a number of additional packages belonging to this pattern. Support for patterns is SuSE-specific and available from openSUSE 10.2 or higher. If a pattern contains unwanted packages it is possible to specify a an ignore list with the element `ignore` and the attribute `name` containing the package name. Restricting a package to a specific architecture can be done by using the `arch` attribute a comma seperated list of allowed architectures in the `package` element.

1.4. Activating an Image

After a logical extend (an image) has been created from a physical extend there are in principal four possibilities to activate the image:

- On a *local system* the image can be installed by dumping (with the command `dd`) the image file on a previously created partition on a local harddisk. To activate the system a boot manager like grub or lilo needs to be used.

An USB stick system is also treated as a local system but KIWI supports you with the installation of such a system by its own --bootstick option.

- In case of a *network enabled system* (netboot client) the image can be installed via a special boot image. The boot image which serves as initial ramdisk (`initrd`) and the appropriate kernel are downloaded from a network service. The linux kernel automatically calls a program named `linuxrc` which takes over all tasks needed to download and install the system image. The installation can be done persistently on disk or temporary into the RAM of the machine. The network boot protocol supported by KIWI is PXE

- In case of a *para virtualized target system* like Xen, the image can be be installed by copying the image file and the KIWI created xen configuration file onto the target system. To activate the virtual system the command

```
xm create -c <xen config file>
```

is used.

- In case of a *full virtualized target system* like VMware, or QEMU the image represents a virtual disk which can be "played" by the virtualisation system.

No matter which of the above mentioned scenarios applies, there is always a special image which takes over control of the activation or deployment process. These images are called boot images and the following sections will explain the KIWI provided boot images in detail.

The KIWI netboot image

The KIWI netboot boot image can be used to install an operating system image to 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 transfer from and to the client

The Boot Process of a Netboot System

The following graphic illustrates a simplified boot process of a netboot client.

Figure 1.2. The Boot Process of a Netboot Client
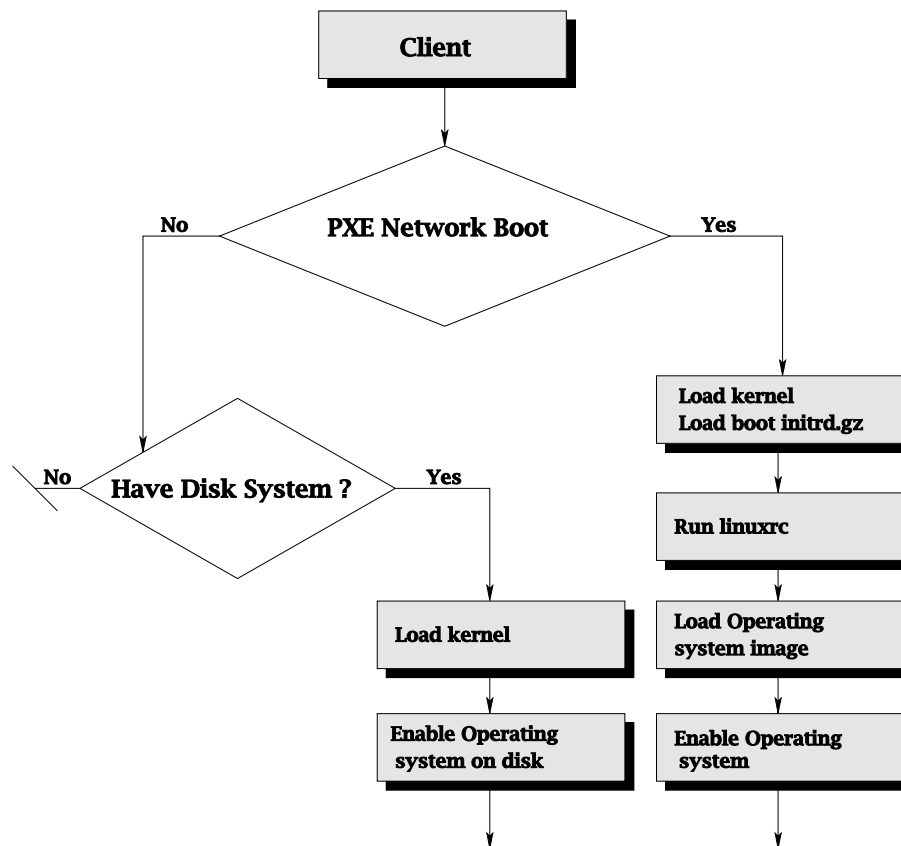
If the system is able to boot via a network, it will load the kernel and the compressed boot image from the network. The "brain" of the boot image is the `linuxrc` script, which does all the stuff controlled by an image configuration file also obtained from the network. The major task is to download and activate 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:

- Via PXE network boot or boot manager (GRUB), the client boots the initrd (`initrd.gz`) which is served by the TFTP server. If no PXE boot is possible, the client tries to boot from a local hard disk.

- Running `linuxrc` starting the following process:

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

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

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

 4. 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 file `/var/lib/dhcpcd/dhcpcd-eth0.info` and the contents of DOMAIN and DNS are used to generate a `/etc/resolv.conf`.

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

 6. If `LOCAL_BOOT` is not set, the configuration file is loaded from the server directory `/var/lib/tftpboot/KIWI` via TFTP. At this point, the client expects the file `config.<MAC Address>`. If this

file is not available the next search is for config.*<HEX-IP>*. HEX-IP is the IP address of the client which each of the four parts of an IPv4 address converted into a hexadecimal value. If this file is not available the IP address search will check only three out of four IP tokens and so on. If still no file can be found the last search is for a file named `config.default`. If this file also is not available and cannot be loaded, it is assumed that the client hasn't existed before and can be immediately registered by uploading a control file to the TFTP server's upload directory `/var/lib/tftpboot/upload`. 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 file `config.`*<MAC address>* 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 infomation about the file format of the configuration file, refer to the section called "The Netboot Client Configuration File `config.`*<MAC Address>*".

7. All registered kernel modules are loaded. The kernel provides a system which allows to check 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 will be loaded.

8. If `LOCAL_BOOT` is not set, the `PART:` line in the configuration is analyzed. If it is found a check is performed to see whether any local system needs to be updated. If not, the local boot process continues immediately. No image download occurs. If an update is required (or no operating system can be found), the client's hard disk is partitioned.

9. If `NFSROOT` and `NBDROOT` and `LOCAL_BOOT` is not set, the images are downloaded with TFTP. If `LOCAL_BOOT` is set this part of the initrd will only read 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 setup a new network block device called /dev/nd0. The image root device is then set to the network block device

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

11. If `LOCAL_BOOT` is not set, a check for `RELOAD_CONFIG` is performed

12. The operating system image is mounted.

13. 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. Additionally the `KIWI_INITRD:` line is evaluated. The specified initrd file will be downloaded from the tftp server and stored in the system image as file `/boot/initrd`.

14. If `LOCAL_BOOT` is not set, all the user-land processes based on the boot image (`dhcpcd -k`) will be terminated.

15. Check if the image is a splitted image by evaluating the contents of the `COMBINED_IMAGE` variable. Both image partsare combined into one system by creating the appropriate filesystem links.

16. If `LOCAL_BOOT` is not set, the filesystem type of the system image as well as the available kernels are determined.

17. If `LOCAL_BOOT` is not set, important configuration files like `/etc/fstab`, `/boot/grub/menu.lst`, `/etc/grub.conf`, and `/etc/sysconfig/kernel` are created.

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

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

20. The boot image is unmounted using an `exec umount` call.

21. At termination of `linuxrc` or the exec call, the kernel initiates the `init` process that 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 main areas:

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

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

Boot files
:   The `/var/lib/tftpboot/boot/` directory is where the `initrd.gz`, and the kernel to boot are kept.

PXE second stage boot loader(s)
:   The `/var/lib/tftpboot/` directory is where the boot loaders for PXE are kept (`pxelinux.0`, `mboot.c32`)

PXE configuration file
:   `/var/lib/tftpboot/pxelinux.cfg` is the location of the PXE configuration file.

Image files and checksums
:   The `/var/lib/tftpboot/image/` directory is where all the image files and their checksums are kept.

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

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

The configuration file contains data about image, configuration, synchronization, and partition parameters. The configuration file is loaded from the TFTP server directory `/var/lib/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. This is an example for 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 (e.g., `/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 can not be used as storage device for the second stage system image. SUSE recommends to use 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. Must always be specified, except in PART. If the block size is too small according to the maximum number of data packages (32768), linuxrc will automatically calculate 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 and the image isn't compressed. It will also fail if you do not specify compressed but the image is compressed. The name of the compressed image has to contain the suffix .gz and needs to be compressed with the gzip tool. Using a compressed image will automatically deactivate the multicast download option of a TFTP.

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 in MB by default. Additionally all size specifications supported by the sfdisk program are allowed as well. 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 size specification.

DISK
: Specifies the hard disk. Used only with PART and defines the device the hard disk can be addressed with, e.g., /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 only makes sense on diskful systems.

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

COMBINED_IMAGE
: If set to an non-empty string, COMBINED_IMAGE indicates that the two images specified need to be combined into one bootable image, where 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 variable's 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 `/var/lib/tftpboot/boot/` directory. Because the tftpserver does a `chroot` into the tftp server path you need to specify the initrd file as follows:

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

NFSROOT

For netboot images there is the possibility to 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 a way that the client can mount it read/write. In order to do that the boot image needs to know the server IP address and the path name on where the root directory exists on this server. The information must be provided like the following example shows:

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

NBDROOT

For netboot images there is the possibility to 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 using the program `nbd-server`. For detailed information how to setup the server please consult the man page of nbd-server. The kernel on the remote client can setup a special network block device named `/dev/nb0` using the command `nbd-client`. Once this device exists the `mount` program is used to mount the root filesystem. In order to allow the KIWI boot image to that the following information must be provided:

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

The variables NBD-Device, NBD-Swap-Port-Number and NBD-Swap-Device are optional. If not set the default values aplies which are /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 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 there is the possibility to 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 will automatically setup the `unionfs/aufs` filesystem. In case of a PXE network image the netboot image requires a `config.<MAC>` setup like the following example shows:

```
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. Please note the device specifications in `UNIONFS_CONFIG` must correspond with the `IMAGE` and `PART` information. The following example explains 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
```

Since the second element of the `PART` list must define the `root` partition, it is absolutely important that the first device in `UNIONFS_CONFIG` references this device as read/write device. The second device of `UNIONFS_CONFIG` has to 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 instance, 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.

The netboot client Control File `hwtype.<MAC Address>`

The control file is primarily used to set up new netboot clients. In this case, there is no configuration file corresponding to the client MAC address available. Using the MAC address information, the control file is created and uploaded to the TFTP servers upload directory `/var/lib/tftpboot/upload`.

The netboot client Hardware-Info File `hwinfo.<MAC Address>`

The hardware info file is primarily used to inform about hardware details of new netboot clients. In this case, there is no configuration file corresponding to the client MAC address available. Using the MAC address information, the hardware info file is created and uploaded to the TFTP servers upload directory `/var/lib/tftpboot/upload`.

The KIWI xennetboot image

The KIWI xennetboot boot image works exactly the same way than the netboot boot image. The only difference is the kernel which is used to boot the client machine. With netboot a standard kernel is used whereas the xennetboot image uses the Xen kernel within the boot image. This allows to boot the system image directly with an enabled Xen kernel and therefore turn the system image into a hypervisor without reboot. In order to be able to boot a Xen kernel via PXE over the network an additional boot loader called mboot.c32 is required. mboot.c32 is a COM32 module for H. Peter Anvin's fine SYSLINUX bootloader, to load multiboot kernels and modules. This allows Xen to be network-booted. Instructions for using mboot.c32 can be found here: 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 will be installed on a disk or into the main memory of a computer. This is done by a deployment architecture which transfers the image via a boot image into its final destination. such a boot image can also exist on a CD. In terms of KIWI this is handled by the isoboot image. Booting a live system can happen by using two techniques:

- Using the *old* style will split the system image into a Read-Only and a Read/Write. The Read/Write image is pushed into main memory whereas the Read-Only part will make it on the CD. (optional compression of the Read-Only part is possible) The isoboot image will take care to make one root tree from both parts by using symbolic links. The CD-Boot structure of KIWI will put the directories `/bin`, `/boot`, `/lib`, `/opt`, `/sbin` and `/usr` on the CD and the rest into the main memory of the system.

- Using the *new* style will make use of the new overlay filesystems like aufs or unionfs. In this case the system image is not divided and the isoboot boot image will create a writable RAM space which is overlayed into one root filesystem. The result is one filesystem which is completely Read/Write. Optionally one can compress the system image which allows to put more data on the CD. The new style is preferred over the old style

Each isoboot description contains a directory named *cdboot*. In there the files *isolinux.cfg* and *isolinux.msg* The first one defines which boot parameters are used for booting the kernel and the second defines the message which is displayed if isolinux doesn't boot in graphics mode. The additional file named *isolinux.sh* is a link to the standard suse-isolinux script which creates an iso image from a specified CD tree. The script will work for non SUSE distributions as well but in CD header of such an image contains information which should be replaced if the distribution is no SUSE based.

The KIWI vmxboot image

The KIWI vmxboot boot image is used for *full* virtualized machines like they are 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 takes over the task of detecting the device of the virtual disk and activates the system on the virtual disk. The boot image also writes some default files like /etc/fstab in order to allow the system default initrd to be created and used. This on the other 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 is able to boot the image which is not the case if for example a compressed image solution with 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 delivered to their customers. The idea is to have a virtual disk image which is simply copied on the real disk of a computer. As soon as the first boot takes place the oemboot boot image takes over the task of repartitioning the real disk which it detects first followed by preparing the system to activate the YaST second stage process. At the end the oemboot image boots into the system and YaST starts with a special configuration sequence made for pre installed systems.

The KIWI xenboot image

The KIWI xenboot boot image works only with the *xen* boot type. The xen boot type creates all information needed to run a Xen virtual machine with the previosly created image. All files in this case means the system image itself which is just a file containing the operating system data and the filesystem, the xenboot boot image as compressed initrd file, the xen kernel and the xen configuration file which references the previosly mentioned files. After KIWI has finished its image creation process a user only needs to call:

```
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. Once 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 files needed to install a USB stick with an operating system. The installation is done by using the KIWI options `--bootstick` , `--bootstick-system`. The files needed in this case are 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. Once the device was found the system can be activated. KIWI also installs the boot manager to the USB stick so that each systen could use the stick as main operating system as soon as the BIOS is able to boot from a USB stick.

1.5. Maintenance of Operating System Images

Creating an image often results in an appliance solution for a customer and gives you the freedom of a working solution at that time. But software develops and you don't want your solution to become outdated. Because of this together with an image people always should think of "image-maintenance". The following paragraph just reflects ideas how to maintain images created by KIWI.

Figure 1.3. Image Maintenance Scenarios
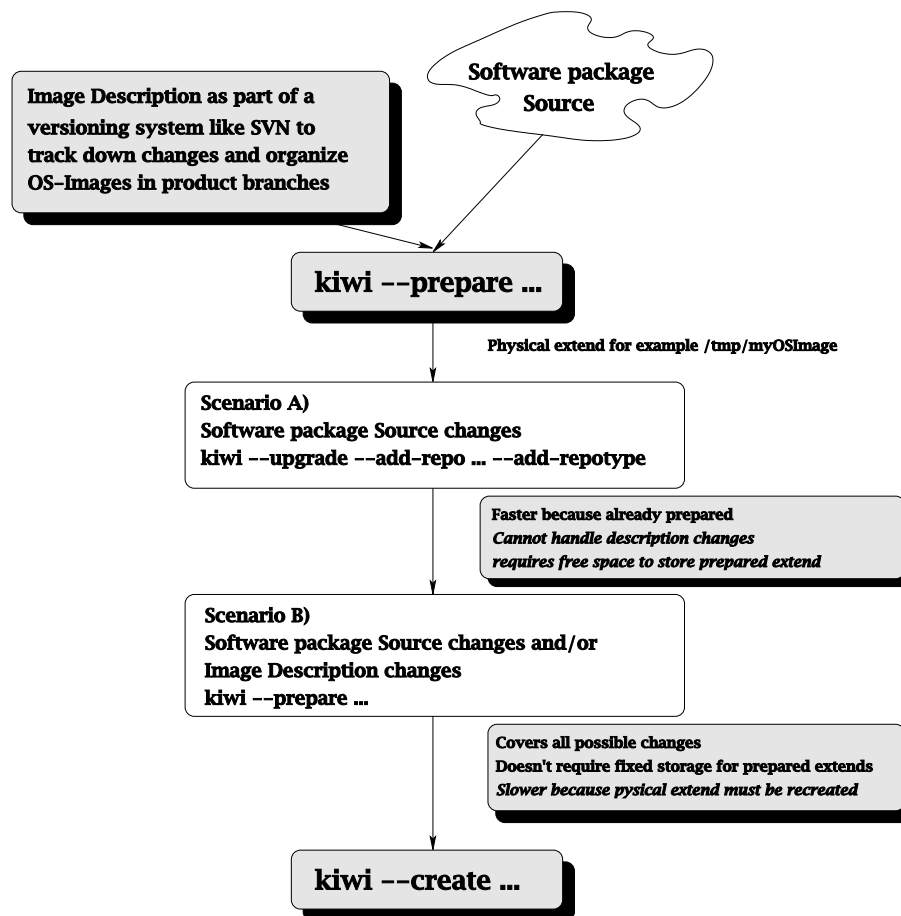
The picture above shows two possible scenarios which requires an image to become updated. The first reason for updating an image are changes to the software, for example a new kernel should be used. If this change doesn't require additional software or changes in the configuration the update can be done by KIWI itself using its `upgrade` option. In combination with `upgrade` KIWI allows to add an additional repository which may be needed if the updated software is not part of the original repository. An important thing to know is that this additional repository is *not* stored into the original `config.xml` file of the image description.

Another reason for updating an image beside software updates are configuration changes or enhancements, for example an image should have replaced its browser with another better browser or a new service like apache should be enabled. In principal it's possible to do all those changes manually within the physical extend but concerning maintenance this would be a nightmare. Why, because it will leave the system in an unversioned condition. Nobody knows what has changed since the very first preparation of this image. So in short *dont't modify physical extends manually*. Changes to the image configuration should be done within the image description. The image description itself should be part of a versioning system like subversion. All changes can be tracked down then and maybe more important can be assigned to product tags and branches. As a consequence an image must be prepared from scratch and the old physical extend could be removed.

1.6. Real-Life Scenarios - A Tutorial

Creating an operating system image always implies the question how to activate the image on the target system. As there are many possible targets the so called image deployment higly depends on the system environment. If a customer buys a product for example SLEPOS the system environment is given and it is required to take care for the rules to successfully deploy the image. On the one hand this makes it easy for customers to control a complex system but on the other hand one will loose the flexibility to use the same system in another environment.

KIWI doesn't stick to a specific system environment but therefore it leaves out important tasks concerning the deployment architecture. So to say KIWI is an image creator but it doesn't setup the deployment infrastructure.

This chapter provides information about the different possibilities to deploy an image and how KIWI fits into this picture.

Deploying via Network Using the PXE Protocol

PXE is a boot protocol mostly implemented in todays BIOS's or boot ROMs of some network cards. If activated it broadcasts the network for a DHCP to obtain an IP address and the information where to find a TFTP server to manage file transfers. If such a server exists the second stage bootloader controls the subsequently boot process. Using PXE with KIWI requires the infrastructure explained in section FIXME \ref{section:tftpstruct} and a TFTP server as well as a DHCP server running. KIWI provides the package `kiwi-pxeboot` which setup the boot structure and installs some prebuild boot images.

If the plan is to use a system image for which no prebuild boot image exist, it is needed to create a custom boot image before a system image could be deployed. Boot images consists of the image itself and the appropriate kernel to that image. Both file must be stored in the directory `/var/lib/tftpboot/boot`. Assuming there is no prebuild 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/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 will be created automatically and from the same source as the system image as soon as 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 `/var/lib/tftpboot/boot`} directory. Optionally they can 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 /var/lib/tftpboot/boot/initrd
cp initrd-netboot.i686-2.1.1.kernel.2.6.18.2-31-default \
  /var/lib/tftpboot/boot/linux
```

Switching on the target machine, which needs to run PXE by default, will load the linux kernel and the KIWI created initrd. The initrd will register the machine if there is no configuration found in `/var/lib/tftpboot/KIWI`. Registration means a file including the MAC address of the machine is uploaded into the directory `/var/lib/tftpboot/upload`. For information about creating the configuration refer to section FIXME \ref{section:confmac}. The following example configuration fits for a machine including a disk on `/dev/sda`:

```
IMAGE=/dev/sda2;full-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 will try to download a system image named `full-suse-10.2.i686-1.1.2` from the TFTP server with IP address 192.168.100.2. On the TFTP server the system images are stored in `/var/lib/tftpboot/image`. The boot image will prepare the disk and create a 1GB swap partition and another full size linux partition. The process of creating this full-suse-10.2 image can be done by using KIWI. Once the boot image has successfully downloaded the system image it is getting activated and operates as configured.

Deploying via TFTP with Initial Boot from CD or USB Stick

Sometimes it is not possible to use PXE as boot protocol. This could happen if the target machine doesn't support PXE or the installed network card doesn't provide a PXE boot rom. Preconditioned your network provides a DHCP and a TFTP server the problem can be solved by storing the boot image on another bootable medium like a CD or an USB stick. Referring to the information from the section above it is easy to create a bootable CD / USB stick with KIWI:

```
kiwi --bootcd /tmp/initrd-netboot-suse-10.2.i686-2.1.1.gz
```

The command will create a bootable ISO image which only needs to be burned on a CD. The following file will be created after the call: `/tmp/initrd-netboot-suse-10.2.i686-2.1.1.cdboot.iso`.

Plug in an USB stick then call:

```
kiwi --bootstick /tmp/initrd-netboot-suse-10.2.i686-2.1.1.gz
```

The command will create a bootable USB stick. KIWI is searching for the USB stick plugged in before and lists all devices found. The user needs to select one of the devices by typing one of the suggested device names. It is important to be careful at that stage because all data on the given device will be lost. Because of this reason the user has to type the device name which hopefully increases the chance not to do something thoughtlessly.

Split image system via PXE

KIWI supports system images to be splitted into two parts, a read-only part and a read-write part. This allows to put data on different filesystems which is mostly used to have read-only data available on a compressed filesystem like cramfs or squasfs. Please note that almost all compressed filesystems available have some kind of restrictions which needs attention before starting to 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 could be changed like the following example shows:

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

Creating an image from this description results in two image files whereas one of them will contain the `-read-only` extension in its name. Ay one may imagine 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 deployed in combination with one of the KIWI boot images.

To deploy the image only PXE or a boot CD / USB-stick can be used. The most important part while making use of split images is the configuration for the target machine. More information on this config.MAC file can be found in section FIXME \ref{section:confmac}. In case of a split image the following information must be provided:

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

- The PART key has to specifiy a partition table with at least three partitions. A swap partition and two system partitions which provides enough space for the first and the second image portion

- The Option COMBINED_IMAGE to tell the boot image to combine both images into one entire 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.i686;1.1.2;192.168.100.2;4096,\
      /dev/sda3;minimal-10.1-read-only.i686;1.1.2;192.168.100.2;4096
PART=200;S;x,500;L;/,x;L;
DISK=/dev/sda
COMBINED_IMAGE=yes
```

Deploying via Network Using an NFS Mounted `root` System

In principal KIWI was designed to upload an image onto a client in different ways but out there you will find diskless machines as well. Those devices doesn't provide permanent storage and rely on the network. The most oft used process to activate such terminals is to NFS mount the system image via the network. KIWI supports that as well but it additionally requires a terminal server configuration which needs to export the system image using a NFS server. The following steps needs to be performed in order to activate a diskless station:

1. Prepare the system image using

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

2. Setup an NFS server which exports the /tmp/kiwi.nfsroot path. The following export options in /etc/exports are recommended:

   ```
   /tmp/kiwi.nfsroot  *(rw,no\_root\_squash,sync,no\_subtree\_check)
   ```

3. Create an appropriate netboot boot image (initrd) with KIWI. Appropriate means the package repository for the system image and the netboot image must be the same

4. Copy the boot image/kernel to the PXE server in /var/lib/tftpboot/boot The package kiwi-pxeboot helps you in setting up the PXE/TFTP server.

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

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

6. Boot the client. If everything works the client will receive 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 will be activated.

Installation from CD or DVD

Sometimes it makes sense to create a CD or DVD which contains an image to become installed on a hard disk of the computer. The installation process in KIWI is rather simple. It doesn't take care for already installed operating systems or disk partitions so be careful with this deployment method. The process of booting a CD with an image as content and the later first boot of this image is handled by the KIWI oemboot boot image. The system image in that 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 the attribute format with the value *iso* must be set. Referring to this the system image description needs the following type specification:

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

Referring to the information above the following KIWI commands needs to be called to create an install ISO image:

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

The command will create a bootable ISO image which only needs to be burned on a CD. KIWI will inform the user about the file name to be burned on CD or DVD.

USB stick system

A very popular method is storing complete operating systems on an USB stick. This means not only the boot image and the kernel is stored on the stick but also the entire system image is part of the stick. To be able to do this KIWI provides special boot image(s) located in `/usr/share/kiwi/image/usbboot`. Currently it is tested and available for SuSE Linux 10.2 only but can be adapted to other versions as well. To use your system image on a boot stick setup the system's `config.xml` as follows:

```
<preferences>
  <type filesystem="ext3" boot="usbboot/suse-10.2">usb</type>
  ...
</preferences>
```

After this the boot and system image needs to be created. Concerning to the size of the stick the image is not allowed to grow beyond it. Assuming the system image is named full-suse-10.2 the command is as follows:

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

The result of the command above is represented in the image file `/tmp/full-suse-10.2.i686-1.1.2`. The usbboot image is created automatically and is stored as `/tmp/initrd-usbboot-suse-10.2.i686-2.1.1.gz`. To create a bootable stick with a SuSE Linux 10.2 operating system on it, plug in the stick and call:

```
kiwi --bootstick /tmp/initrd-usbboot-suse-10.2.i686-2.1.1.gz \
     --bootstick-system /tmp/full-suse-10.2.i686-1.1.2
```

Virtual disk system (QEMU or VMware)

To be able to use a virtualization system a virtual disk needs to be created. This can be done by specifying the following type in the system's image `config.xml`:

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

After this the system image can be created. The process will create the system image and the specified vmxboot/ suse-10.2 boot image. The result is then used to create the virtual disk. The following command needs to be called:

```
kiwi --prepare full-suse-10.2 --root /tmp/mysystem
```

```
kiwi --create /tmp/mysystem -d /tmp
```

The result of the command above is a set of virtual disks, one with suffix .qemu (QEMU) and the other with the suffix .vmdk (VMware). To run the system on the virtual disk with for example qemu call:

```
qemu /tmp/full-suse-10.2.i686-1.1.2.qemu
```

Para Virtual Image for Xen

To be able to use an image within Xen a system image and an initrd file needs to be created. This can be done by specifying the following type in the system's image config.xml:

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

After this the system image can be created. The process will create the system image and the specified xenboot/ suse-10.2 boot image. Additionally an appropriate Xen configuration file will be created. The config file will use the .xenconfig suffix. The following command needs to be called:

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

To run the system within Xen one need to call:

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

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), DVD (Live DVD). The term "live" derives from the fact that these distributions are a complete, runnable—i.e., "live"— instance of the operating system residing on the distribution medium, rather than the typical case of a collection of packages that must first be installed on the target machine before using the OS.

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 typically would be stored on a hard drive into temporary memory, such as a ram disk. In fact, a hard drive is not needed at all. However, this does cut down on the RAM available to applications, reducing performance somewhat.

How to Setup an Image as Live CD

To create an .iso image which can be burned on CD you only need to specify the boot image which should handle your live system. This is done 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 attribute boot refers to the CD boot image which must exist in /usr/share/kiwi/image/isoboot. Like for all boot images the most important point is that the boot image has to match the operating system image.

This means the kernel of the boot- and operating system image must be the same. If there is no boot image which matches you need to create your own boot image description. A good starting point for this is to use an existing boot image and adapt it to your needs.

1.7. Troubleshooting

TODO

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.