AutoYaST

SUSE Linux Enterprise Server 12 SP5

AutoYaST

SUSE Linux Enterprise Server 12 SP5

AutoYaST is a system for unattended mass deployment of SUSE Linux Enterprise Server systems using an AutoYaST profile containing installation and configuration data. The manual guides you through the basic steps of auto-installation: preparation, installation, and configuration.

Publication Date: December 02, 2019

SUSE LLC
10 Canal Park Drive
Suite 200
Cambridge MA 02141
USA
https://www.suse.com/documentation

Copyright © 2006– 2019 SUSE LLC and contributors. All rights reserved.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or (at your option) version 1.3; with the Invariant Section being this copyright notice and license. A copy of the license version 1.2 is included in the section entitled “GNU Free Documentation License”.

For SUSE trademarks, see http://www.suse.com/company/legal/. All other third-party trademarks are the property of their respective owners. Trademark symbols (®, ™ etc.) denote trademarks of SUSE and its affiliates. Asterisks (*) denote third-party trademarks.

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

Contents

1 Introduction 1

1.1 Motivation 1

1.2 Overview and Concept 2

2 The Control File 4

2.1 Introduction 4

2.2 Format 4

2.3 Structure 5

Resources and Properties 6 • Nested Resources 6 • Attributes 7

3 Creating a Control File 9

3.1 Collecting Information 9

3.2 Using the Configuration Management System (CMS) 9

Creating a New Control File 10

3.3 Creating/Editing a Control File Manually 11

3.4 Creating a Control File via Script with XSLT 11

4 Configuration and Installation Options 14

4.1 General Options 14

The Mode Section 15 • Configuring the Installation Settings Screen 19 • The Self-Update Section 19 • The Semi-Automatic Section 20 • The Signature Handling Section 21 • The Storage Section 23 • The Wait Section 23 • Blacklisting Unused Devices on IBM Z 25 • Examples for the `general` Section 25

4.2 Reporting 27

4.3 System Registration and Extension Selection 28

Extensions 32

4.4 The Boot Loader 33
Loader Type 34 • Globals 34 • Device map 38

4.5 Partitioning 39
Drive Configuration 39 • Partition Configuration 43 • Btrfs subvolumes 49 • RAID Options 50 • Automated Partitioning 51 • Advanced Partitioning Features 54 • Using an Existing Mount Table (fstab) 59 • Logical Volume Manager (LVM) 60 • Software RAID 61 • Multipath Support 63 • IBM Z Specific Configuration 64

4.6 iSCSI Initiator Overview 66

4.7 Fibre Channel over Ethernet Configuration (FCoE) 67

4.8 Country Settings 69

4.9 Software 70
Package Selection with Patterns 70 • Installing Additional/Customized Packages or Products 70 • Kernel Packages 77 • Removing Automatically Selected Packages 77 • Installing Recommended Packages/ Patterns 78 • Installing Packages in Stage 2 79 • Installing Patterns in Stage 2 79 • Online Update in Stage 2 79

4.10 Upgrade 80

4.11 Services and Targets 81

4.12 Network Configuration 82
Persistent Names of Network Interfaces 85 • s390 Options 85 • Proxy 86 • (X)Inetd 87

4.13 NIS Client 88

4.14 NIS Server 88

4.15 LDAP Server (Authentication Server) 91

4.16 Windows Domain Membership 95

4.17 Samba Server 96

4.18 Authentication Client 97

4.19 NFS Client and Server 98

4.20 NTP Client 99

4.21 Mail Configuration 100

4.22 HTTP Server Configuration 102

4.23 Squid Server 111

4.24 FTP Server 118

4.25 TFTP Server 122

4.26 Firstboot Workflow 123

4.27 Security Settings 123
Password Settings Options 124 • Boot Settings 124 • Login Settings 125 • New user settings (**useradd** settings) 125

4.28 Linux Audit Framework (LAF) 125

4.29 Users and Groups 127
Users 128 • User Defaults 133 • Groups 134 • Login Settings 135

4.30 Custom User Scripts 136
Pre-Install Scripts 136 • Post-partitioning Scripts 137 • Chroot Environment Scripts 138 • Post-Install Scripts 138 • Init Scripts 138 • Script XML Representation 140 • Script Example 143

4.31 System Variables (Sysconfig) 145

4.32 Adding Complete Configurations 146

4.33 Ask the User for Values during Installation 147
Default Value Scripts 156 • Scripts 157

4.34 Kernel Dumps 162
Memory Reservation 163 • Dump Saving 165 • E-Mail Notification 167 • Kdump Kernel Settings 169 • Expert Settings 170

4.35 DNS Server 171

4.36 DHCP Server 173

4.37 SUSE Firewall 177
General Firewall Configuration 177 • Firewall Zones Configuration 179 • A Full Example 181

4.38 Miscellaneous Hardware and System Components 182
Printer 182 • Sound devices 184

4.39 Importing SSH Keys and Configuration 184

4.40 Configuration Management 185

5 Rules and Classes 187

5.1 Rules-based Automatic Installation 187
Rules File Explained 188 • Custom Rules 191 • Match Types for Rules 191 • Combine Attributes 192 • Rules File Structure 192 • Predefined System Attributes 193 • Rules with Dialogs 195

5.2 Classes 198

5.3 Mixing Rules and Classes 200

5.4 Merging of Rules and Classes 200

6 The Auto-Installation Process 203

6.1 Introduction 203
X11 Interface (graphical) 203 • Serial console 203 • Text-based YaST Installation 203

6.2 Choosing the Right Boot Medium 204
Booting from a Flash Disk 204 • Booting from DVD-ROM 204 • Booting via PXE over the Network 204

6.3 Invoking the Auto-Installation Process 205
Command Line Options 206 • Auto-installing a Single System 212 • Combining the linuxrc `info` file with the AutoYaST control file 212

6.4 System Configuration 213
Post-Install and System Configuration 213 • System Customization 214

7 Running AutoYaST in an Installed System 215

A Handling Rules 217

B AutoYaST FAQ - Frequently Asked Questions 218

C Advanced Linuxrc Options 222

C.1 Passing parameters to Linuxrc 222

C.2 `info` file format 222

C.3 Advanced Network Setup 225

D GNU Licenses 227

D.1 GNU Free Documentation License 227

1 Introduction

AutoYaST is a system for unattended mass deployment of SUSE Linux Enterprise Server systems. AutoYaST installations are performed using an AutoYaST control file (also called “profile”) with installation and configuration data. That control file can be created using the configuration interface of AutoYaST and can be provided to YaST during installation in different ways.

1.1 Motivation

In an article in issue 78, the Linux Journal (http://www.linuxjournal.com/) writes:

“A standard Linux installation asks many questions about what to install, what hardware to configure, how to configure the network interface, etc. Answering these questions once is informative and maybe even fun. But imagine a system engineer who needs to set up a new Linux network with many machines. Now, the same issues need to be addressed and the same questions answered repeatedly. This makes the task very inefficient, not to mention a source of irritation and boredom. Hence, a need arises to automate this parameter and option selection.”

“The thought of simply copying the hard disks naturally crosses one's mind. This can be done quickly, and all the necessary functions and software will be copied without option selection. However, the fact is that simple copying of hard disks causes the individual computers to become too similar. This, in turn, creates an altogether new mission of having to reconfigure the individual settings on each PC. For example, IP addresses for each machine will need to be reset. If this is not done properly, strange and inexplicable behavior results.”

A regular installation of SUSE Linux Enterprise Server is semi-automated by default. The user is prompted to select the necessary information at the beginning of the installation (usually language only). YaST then generates a proposal for the underlying system depending on different factors and system parameters. Usually—and especially for new systems—such a proposal can be used to install the system and provides a usable installation. The steps following the proposal are fully automated.

AutoYaST can be used where no user intervention is required or where customization is required. Using an AutoYaST control file, YaST prepares the system for a custom installation and does not interact with the user, unless specified in the file controlling the installation.

AutoYaST is not an automated GUI system. This means that usually many screens will be skipped —you will never see the language selection interface, for example. AutoYaST will simply pass the language parameter to the sub-system without displaying any language related interface.

1.2 Overview and Concept

Using AutoYaST, multiple systems can easily be installed in parallel and quickly. They need to share the same environment and similar, but not necessarily identical, hardware. The installation is defined by an XML configuration file (usually named `autoinst.xml`) called the "AutoYaST control file". It can initially be created using existing configuration resources easily be tailored for any specific environment.

AutoYaST is fully integrated and provides various options for installing and configuring a system. The main advantage over other auto-installation systems is the possibility to configure a computer by using existing modules and avoiding using custom scripts which are normally executed at the end of the installation.

This document will guide you through the three steps of auto-installation:

- Preparation: All relevant information about the target system is collected and turned into the appropriate directives of the control file. The control file is transferred onto the target system where its directives will be parsed and fed into YaST.
- Installation: YaST performs the installation of the basic system using the data from the AutoYaST control file.
- Configuration: After the installation of the basic system, the system configuration is performed in the second stage of the installation. User defined post-installation scripts from the AutoYaST control file will also be executed at this stage.

Note: Second Stage

A regular installation of SUSE Linux Enterprise Server 12 SP5 is performed in a single stage. The auto-installation process, however, is divided into two stages. After the installation of the basic system the system boots into the second stage where the system configuration is done.

The second stage can be turned off with the `second_stage` parameter:

```
<general>
  <mode>
    <confirm config:type="boolean">false</confirm>
    <second_stage config:type="boolean">false</second_stage>
  </mode>
</general>
```

The complete and detailed process is illustrated in the following figure:

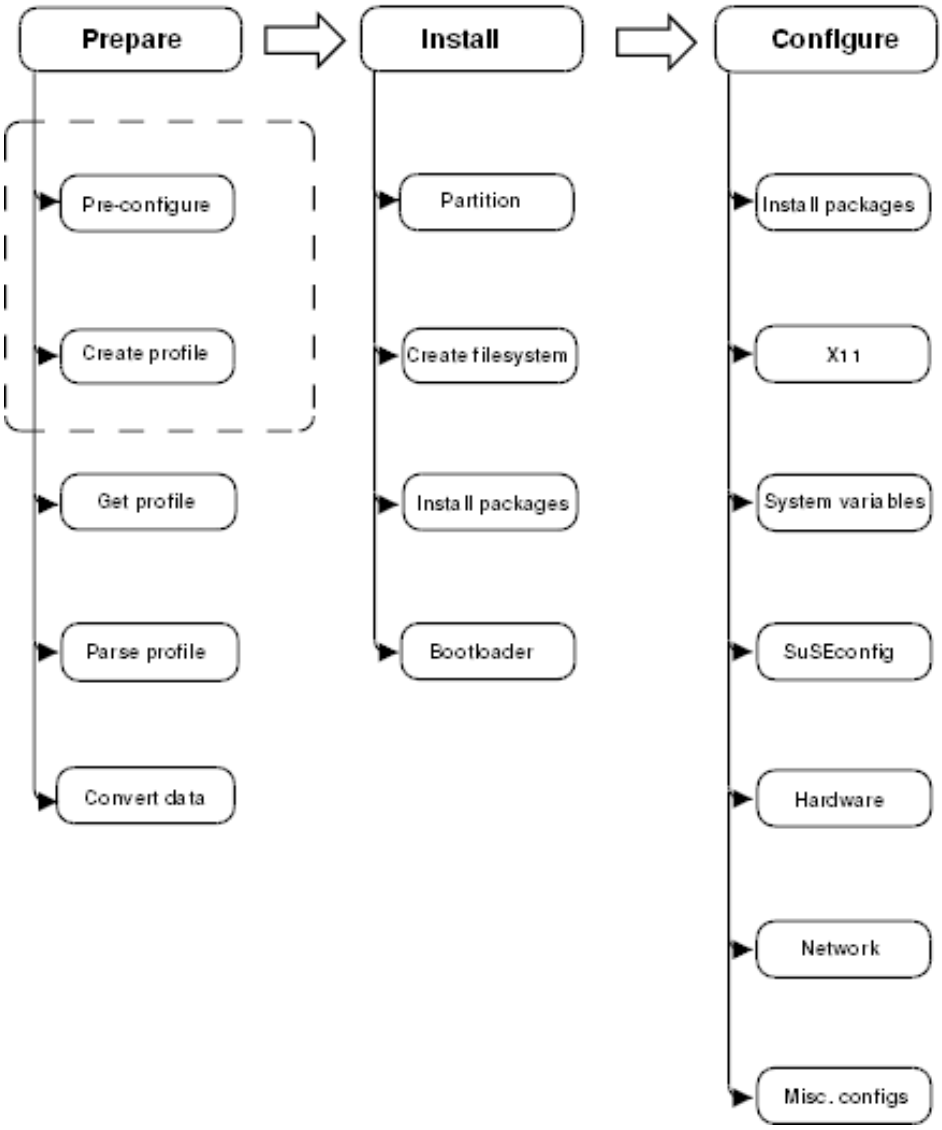

FIGURE 1.1: AUTO-INSTALLATION PROCESS

2 The Control File

2.1 Introduction

The control file usually is a configuration description for a single system. It consists of sets of resources with properties including support for complex structures such as lists, records, trees and large embedded or referenced objects.

Important: Control Files from Previous Releases are Incompatible

A lot of major changes were introduced with SUSE Linux Enterprise Server 12 SP5 (the switch to systemd and GRUB 2 for example). These changes also required fundamental changes in AutoYaST, therefore you cannot use AutoYaST control files created on previous SUSE Linux Enterprise Server versions to install SUSE Linux Enterprise Server 12 SP5 and vice versa.

2.2 Format

The XML configuration format provides a consistent file structure, which is easy to learn and to remember when attempting to configure a new system.

The AutoYaST control file uses XML to describe the system installation and configuration. XML is a commonly used markup, and many users are familiar with the concepts of the language and the tools used to process XML files. If you edit an existing control file or create a control file using an editor from scratch, it is strongly recommended to validate the control file. This can be done using a validating XML parser such as `xmllint` or `jing`, for example (see *Section 3.3, "Creating/Editing a Control File Manually"*).

The following example shows a control file in XML format:

EXAMPLE 2.1: AUTOYAST CONTROL FILE (PROFILE)

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile
  xmlns="http://www.suse.com/1.0/yast2ns"
  xmlns:config="http://www.suse.com/1.0/configns">
  <partitioning  config:type="list">
```

```
    <drive>
      <device>/dev/sda</device>
      <partitions config:type="list">
        <partition>
          <filesystem config:type="symbol">btrfs</filesystem>
          <size>10G</size>
          <mount>/</mount>
        </partition>
        <partition>
          <filesystem config:type="symbol">xfs</filesystem>
          <size>120G</size>
          <mount>/data</mount>
        </partition>
      </partitions>
    </drive>
  </partitioning>
  <scripts>
    <pre-scripts>
      <script>
        <interpreter>shell</interpreter>
        <filename>start.sh</filename>
        <source>
        <![CDATA[
#!/bin/sh
echo "Starting installation"
exit 0

]]>

        </source>
      </script>
    </pre-scripts>
  </scripts>
</profile>
```

2.3 Structure

Below is an example of a basic control file container, the actual content of which is explained later on in this chapter.

EXAMPLE 2.2: CONTROL FILE CONTAINER

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile
```

```
  xmlns="http://www.suse.com/1.0/yast2ns"
  xmlns:config="http://www.suse.com/1.0/configns">
  <!-- RESOURCES -->
</profile>
```

The `<profile>` element (root node) contains one or more distinct resource elements. The permissible resource elements are specified in the schema files

2.3.1 Resources and Properties

A resource element either contains multiple and distinct property and resource elements, or multiple instances of the same resource element, or it is empty. The permissible content of a resource element is specified in the schema files.

A property element is either empty or contains a literal value. The permissible property elements and values in each resource element are specified in the schema files

An element can be either a container of other elements (a resource) or it has a literal value (a property); it can never be both. This restriction is specified in the schema files. A configuration component with more than one value must either be represented as an embedded list in a property value or as a nested resource.

2.3.2 Nested Resources

Nested resource elements allow a tree-like structure of configuration components to be built to any level.

EXAMPLE 2.3: NESTED RESOURCES

```
...
<drive>
  <device>/dev/sda</device>
  <partitions> <!-- this is wrong, explanation below -->
    <partition>
      <size>10G</size>
      <mount>/</mount>
    </partition>
    <partition>
      <size>1G</size>
      <mount>/tmp</mount>
    </partition>
  </partitions>
```

```
</drive>
....
```

In the example above the disk resource consists of a device property and a partitions resource. The partitions resource contains multiple instances of the partition resource. Each partition resource contains a size and mount property.

The XML schema defines the partitions element as a resource supporting one or multiple partition element children. If only one partition resource is specified, it is important to use the `config:type` attribute of the partitions element to indicate that the content is a resource, in this case a list. Using the partitions element without specifying the type in this case will result in undefined behavior, as YaST will incorrectly interpret the partitions resource as a property. The example below illustrates this use case.

EXAMPLE 2.4: NESTED RESOURCES WITH TYPE ATTRIBUTES

```
...
<drive>
  <device>/dev/sda</device>
  <partitions config:type="list">
     <partition>
        <size>10G</size>
        <mount>/</mount>
     </partition>
     <partition>
        <size>1G</size>
        <mount>/tmp</mount>
     </partition>
  </partitions>
</drive>
....
```

2.3.3 Attributes

Global attributes are used to define metadata on resources and properties. Attributes are used to define context switching. They are also used for naming and typing properties as shown in the previous sections. Attributes are in a separate namespace so they do not need to be treated as reserved words in the default namespace.

Global attributes are defined in the configuration namespace and must always be prefixed with `config:` . All attributes are optional. Most can be used with both resource and property elements but some can only be used with one type of element which is specified in the schema files.

The type of an element is defined using the `config:type` attribute. The type of a resource element is always RESOURCE, although this can also be made explicit with this attribute (to ensure correct identification of an empty element, for example, when there is no schema file to refer to). A resource element cannot be any other type and this restriction is specified in the schema file. The type of a property element determines the interpretation of its literal value. The type of a property element defaults to `STRING`, as specified in the schema file. The full set of permissible types is specified in the schema file.

3 Creating a Control File

3.1 Collecting Information

To create the control file, you need to collect information about the systems you are going to install. This includes hardware data and network information among other things. Make sure you have the following information about the machines you want to install:

- Hard disk types and sizes
- Graphical interface and attached monitor, if any
- Network interface and MAC address if known (for example, when using DHCP)

3.2 Using the Configuration Management System (CMS)

To create the control file for one or more computers, a configuration interface based on YaST is provided. This system depends on existing modules which are usually used to configure a computer in regular operation mode, for example, after SUSE Linux Enterprise Server is installed.

The configuration management system lets you easily create control files and manage a repository of configurations for the use in a networked environment with multiple clients.

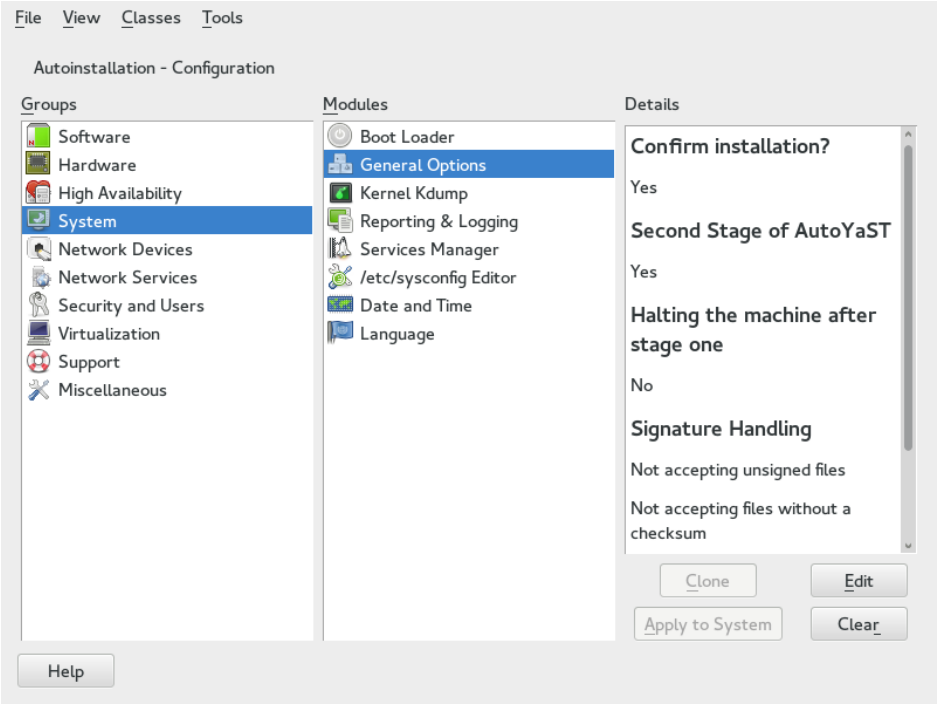

FIGURE 3.1: CONFIGURATION SYSTEM

3.2.1 Creating a New Control File

With some exceptions, almost all resources of the control file can be configured using the configuration management system. The system offers flexibility and the configuration of some resources is identical to the one available in the YaST control center. In addition to the existing and familiar modules new interfaces were created for special and complex configurations, for example for partitioning, general options and software.

Furthermore, using a CMS guarantees the validity of the resulting control file and its direct use for starting automated installation.

Make sure the configuration system is installed (package `autoyast2`) and call it using the YaST control center or as root with the following command (make sure the `DISPLAY` variable is set correctly to start the graphical user interface instead of the text-based one):

```
/sbin/yast2 autoyast
```

3.3 Creating/Editing a Control File Manually

If editing the control file manually, make sure it has a valid syntax. To check the syntax, use the tools already available on the distribution. For example, to verify that the file is well-formed (has a valid XML structure), use the utility **xmllint** available with the libxml2 package:

```
xmllint <control file>
```

If the control file is not well formed, for example, if a tag is not closed, **xmllint** will report the errors.

To validate the control file, use the tool **jing** from the package with the same name. During validation misplaced or missing tags and attributes and wrong attribute values are detected. the package jing is provided with the SUSE Software Development Kit.

```
jing /usr/share/YaST2/schema/autoyast/rng/profile.rng <control file>
```

/usr/share/YaST2/schema/autoyast/rng/profile.rng is provided by the package yast2-schema. This file describes the syntax and classes of an AutoYaST profile.

Before going on with the autoinstallation, fix any errors resulting from such checks. The autoinstallation process cannot be started with an invalid and not well-formed control file.

You can use any XML editor available on your system or any text editor with XML support (for example, Emacs, Vim). However, it is not optimal to create the control file manually for many machines and it should only be seen as an interface between the autoinstallation engine and the Configuration Management System (CMS).

Tip: Using Emacs as an XML Editor

The built-in nxml-mode turns Emacs into a fully-fledged XML editor with automatic tag completion and validation. Refer to the Emacs help for instructions on how to set up nxml-mode.

3.4 Creating a Control File via Script with XSLT

If you have a template and want to change a few things via script or command line, use an XSLT processor like xsltproc. For example, if you have an AutoYaST control file and want to fill out the host name via script for any reason (if doing this so often, you want to script it).

First, create an XSL file:

EXAMPLE 3.1: EXAMPLE FILE FOR REPLACING THE HOST NAME/DOMAIN BY SCRIPT

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:y2="http://www.suse.com/1.0/yast2ns"
  xmlns:config="http://www.suse.com/1.0/configns"
  xmlns="http://www.suse.com/1.0/yast2ns"
  version="1.0">
  <xsl:output method="xml" encoding="UTF-8" indent="yes" omit-xml-declaration="no" cdata-
section-elements="source"/>

  <!-- the parameter names -->
  <xsl:param name="hostname"/>
  <xsl:param name="domain"/>

  <xsl:template match="/">
    <xsl:apply-templates select="@*|node()"/>
  </xsl:template>

  <xsl:template match="y2:dns">
    <xsl:copy>
      <!-- where to copy the parameters -->
      <domain><xsl:value-of select="string($domain)"/></domain>
      <hostname><xsl:value-of select="string($hostname)"/></hostname>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>


  <xsl:template match="@*|node()" >
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```

This file expects the host name and the domain name as parameters from the user.

```
<xsl:param name="hostname"/>
<xsl:param name="domain"/>
```

There will be a copy of those parameters in the DNS section of the control file. That means, if there already is a domain element in the DNS section, you will get a second one which is not good.

For more information about XSLT, go to the official Web page www.w3.org/TR/xslt (http://www.w3.org/TR/xslt)

4 Configuration and Installation Options

This chapter introduces important parts of a control file for standard purposes. To learn about other available options, use the configuration management system.

Note that for some configuration options to work, additional packages need to be installed, depending on the software selection you have configured. If you choose to install a minimal system then some packages might be missing and need to be added to the individual package selection.

YaST will install packages required in the second phase of the installation and before the post-installation phase of AutoYaST has started. However, if necessary YaST modules are not available in the system, important configuration steps will be skipped. For example, no security settings will be configured if `yast2-security` is not installed.

4.1 General Options

The general section includes all settings that influence the installation workflow. The overall structure of this section looks like the following:

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns"
 xmlns:config="http://www.suse.com/1.0/configns">
 <general>
  <ask-list>❶
   ...
  </ask-list>
  <cio_ignore>❷
   ...
  </cio_ignore>
  <mode>❸
   ...
  </mode>
  <proposals>❹
   ...
  </proposals>
  <self_update>❺
   ...
  </self_update>
  <self_update_url>
   ...
```

```
  </self_update_url>
  <semi-automatic config:type="list">❻
   ...
  </semi-automatic>
  <signature-handling>❼
   ...
  </signature-handling>
  <storage>❽
   ...
  </storage>
  <wait>❾
   ...
  </wait>
 </general>
<profile>
```

❶ *Section 4.33, “Ask the User for Values during Installation”*

❷ *Section 4.1.8, “Blacklisting Unused Devices on IBM Z”*

❸ *Section 4.1.1, “The Mode Section”*

❹ *Section 4.1.2, “Configuring the Installation Settings Screen”*

❺ *Section 4.1.3, “The Self-Update Section”*

❻ *Section 4.1.4, “The Semi-Automatic Section”*

❼ *Section 4.1.5, “The Signature Handling Section”*

❽ *Section 4.1.6, “The Storage Section”*

❾ *Section 4.1.7, “The Wait Section”*

4.1.1 The Mode Section

The mode section configures the behavior of AutoYaST with regard to user confirmations and rebooting. The following elements are allowed in the `mode` section:

`activate_systemd_default_target`

If you set this entry to `false`, the default `systemd` target will not be activated via the call **`systemctl isolate`**. Setting this value is optional. The default is `true`.

```
<general>
 <mode>
  <activate_systemd_default_target config:type="boolean">
   true
  </activate_systemd_default_target>
```

```
 </mode>
 ...
</general>
```

`confirm`

By default, the installation stops at the *Installation Settings* screen. Up to this point, no changes have been made to the system and settings may be changed on this screen. To proceed and finally start the installation, the user needs to confirm the settings. By setting this value to `false` the settings are automatically accepted and the installation starts. Only set to `false` if you want to carry out a fully unattended installation. Setting this value is optional. The default is `true`.

```
<general>
 <mode>
  <confirm config:type="boolean">true</confirm>
 </mode>
 ...
</general>
```

`confirm_base_product_license`

If you set this to `true`, the EULA of the base product will be shown. The user needs to accept this license. Otherwise the installation will be canceled. Setting this value is optional. The default is `false`. This setting applies to the base product license only. Use the flag `confirm_license` in the `add-on` section for additional licenses (see *Section 4.9.2, "Installing Additional/Customized Packages or Products"* for details).

```
<general>
 <mode>
  <confirm_base_product_license config:type="boolean">
   false
  </confirm_base_product_license>
 </mode>
 ...
</general>
```

`final_halt`

If you set this to `true`, the machine will shut down at the very end of the installation (when everything is installed and configured at the end of the second stage). Setting this value is optional. The default is `true`. It makes no sense to set both this and `final_reboot` to `true`.

```
<general>
```

```
 <mode>
  <final_halt config:type="boolean">true</final_halt>
 </mode>
 ...
</general>
```

`final_reboot`

If you set this to `true`, the machine will reboot at the end of the installation (when everything is installed and configured at the end of the second stage). Setting this value is optional. The default is `true`. It makes no sense to set both this and `final_halt` to `true`.

```
<general>
 <mode>
  <final_reboot config:type="boolean">true</final_reboot>
 </mode>
 ...
</general>
```

`final_restart_services`

If you set this entry to `false`, services will *not* be restarted at the end of the installation (when everything is installed and configured at the end of the second stage). Setting this value is optional. The default is `true`.

```
<general>
 <mode>
  <final_restart_services config:type="boolean">
   true
  </final_restart_services>
 </mode>
 ...
</general>
```

`halt`

Shuts down the machine after the first stage. All packages and the boot loader have been installed and all your chroot scripts have run. Instead of rebooting into stage two, the machine is turned off. If you turn it on again, the machine boots and the second stage of the autoinstallation starts. Setting this value is optional. The default is `false`.

```
<general>
 <mode>
  <halt config:type="boolean">false</halt>
 </mode>
 ...
```

```
</general>
```

`max_systemd_wait`

Specifies how long AutoYaST waits (in seconds) at most for `systemd` to set up the default target. Setting this value is optional and should not normally be required. The default is `30` (seconds).

```
<general>
 <mode>
  <max_systemd_wait config:type="integer">30</max_systemd_wait>
 </mode>
 ...
</general>
```

`ntp_sync_time_before_installation`

Specify the NTP server with which to synchronize time before starting the installation. Time synchronization will only occur if this option is set. Keep in mind that you need a network connection and access to a time server. Setting this value is optional. By default no time synchronization will occur.

```
<general>
 <mode>
  <ntp_sync_time_before_installation>
   &ntpname;
  </max_systemd_wait>
 </mode>
 ...
</general>
```

`second_stage`

A regular installation of SUSE Linux Enterprise Server is performed in a single stage. The auto-installation process, however, is divided into two stages. After the installation of the basic system the system boots into the second stage where the system configuration is done. Set this option to `false` to disable the second stage. Setting this value is optional. The default is `true`.

```
<general>
 <mode>
  <second_stage config:type="boolean">true</second_stage>
 </mode>
 ...
</general>
```

4.1.2 Configuring the Installation Settings Screen

AutoYaST allows you to configure the *Installation Settings* screen, which shows a summary of the installation settings. On this screen, the user can change the settings before confirming them to start the installation. Using the `proposal` tag, you can control which settings ("proposals") are shown in the installation screen. A list of valid proposals for your products is available from the `/control.xml` file on the installation medium. This setting is optional. By default all configuration options will be shown.

```
<proposals config:type="list">
 <proposal>partitions_proposal</proposal>
 <proposal>timezone_proposal</proposal>
 <proposal>software_proposal</proposal>
</proposals>
```

4.1.3 The Self-Update Section

During the installation, YaST can update itself to solve bugs in the installer that were discovered after the release. Refer to the *Deployment Guide* for further information about this feature. Use the following tags to configure the YaST self-update:

`self_update`

This option enables (set to `true`) or disables (set to `false`) the YaST self-update feature. Setting this value is optional. The default is `true`.

```
<general>
 <self_update config:type="boolean">true</self_update>
 ...
</general>
```

Alternatively, you can specify the boot parameter `self_update=1` on the kernel command line.

`self_update_url`

Location of the update repository to use during the YaST self-update. For more information, refer to the *Deployment Guide*.

Important: Installer Self-Update Repository Only

The `self_update_url` parameter expects only the installer self-update repository URL. Do not supply any other repository URL—for example the URL of the software update repository.

```
<general>
 <self_update_url>
  http://example.com/updates/$arch
 </self_update_url>
 ...
</general>
```

The URL may contain the variable `$arch`. It will be replaced by the system's architecture, such as `x86_64`, `s390x`, etc.
Alternatively, you can specify the boot parameter `self_update=1` together with `self_update=`*URL* on the kernel command line.

4.1.4 The Semi-Automatic Section

AutoYaST offers to start some YaST modules during the installation. This is useful if you want to give administrators installing the machine the possibility to manually configure some aspects of the installation while at the same time automating the rest of the installation. Within the semi-automatic section you can start the following YaST modules:

- The network settings module (`networking`)
- The partitioner (`partitioning`)
- The registration module (`scc`)

The following example starts all three supported YaST modules during the installation:

```
<general>
 <semi-automatic config:type="list">
  <semi-automatic_entry>networking</semi-automatic_entry>
  <semi-automatic_entry>scc</semi-automatic_entry>
  <semi-automatic_entry>partitioning</semi-automatic_entry>
 </semi-automatic>
</general>
```

4.1.5 The Signature Handling Section

By default AutoYaST will only install signed packages from sources with known GPG keys. Use this section to overwrite the default settings.

Warning: Overwriting the Signature Handling Defaults

Installing unsigned packages, packages with failing checksum checks, or when accepting packages from sources you do not trust is a major security risk. Packages may have been modified and may install malicious software on your machine. Only overwrite the defaults in this section if you are sure the repository and the packages can be trusted. SUSE is not responsible for any problems arising from software installed with integrity checks disabled.

Default values for all options are false. If an option is set to false and a package or repository fails the respective test, it is silently ignored and will not be installed.

`accept_unsigned_file`

If set to `true`, AutoYaST will accept unsigned files like the content file.

```
<general>
 <signature-handling>
  <accept_unsigned_file config:type="boolean">
   false
  </accept_unsigned_file>
 <signature-handling>
 ...
<general>
```

`accept_file_without_checksum`

If set to `true`, AutoYaST will accept files without a checksum in the content file.

```
<general>
 <signature-handling>
  <accept_file_without_checksum config:type="boolean">
   false
  </accept_file_without_checksum>
 <signature-handling>
 ...
<general>
```

`accept_verification_failed`

If set to `true`, AutoYaST will accept signed files even when the signature verification fails.

```
<general>
 <signature-handling>
  <accept_verification_failed config:type="boolean">
   false
  </accept_verification_failed>
 <signature-handling>
 ...
<general>
```

`accept_unknown_gpg_key`

If set to `true`, AutoYaST will accept new GPG keys of the installation sources, for example the key used to sign the content file.

```
<general>
 <signature-handling>
  <accept_unknown_gpg_key config:type="boolean">
   false
  </accept_unknown_gpg_key>
 <signature-handling>
 ...
<general>
```

`accept_non_trusted_gpg_key`

Set this option to `true` to accept known keys you have not yet trusted.

```
<general>
 <signature-handling>
  <accept_non_trusted_gpg_key config:type="boolean">
   false
  </accept_non_trusted_gpg_key>
 <signature-handling>
 ...
<general>
```

`import_gpg_key`

If set to `true`, AutoYaST will accept and import new GPG keys on the installation source in its database.

```
<general>
 <signature-handling>
  <import_gpg_key config:type="boolean">
```

```
   false
  </import_gpg_key>
 <signature-handling>
 ...
<general>
```

4.1.6 The Storage Section

This section lets you enable multipath support for the installation. You may also configure the partition alignment settings here.

`btrfs_set_default_subvolume_name`

See *Section 4.5.3, "Btrfs subvolumes"* for more information.

`start_multipath`

When installing on a network storage that is accessed via multiple paths, you need to enable multipath for the installation by setting this parameter to `true`. Setting this value is optional. The default is `false`.

```
<general>
 <storage>
  <start_multipath config:type="boolean">true</start_multipath>
 <storage>
 ...
</general>
```

Alternatively, you can use the following parameter on the Kernel command line: `LIBSTORAGE_MULTIPATH_AUTOSTART=ON`

4.1.7 The Wait Section

In the second stage of the installation the system is configured by running modules, for example the network configuration. Within the `wait` section you can define scripts that will get executed before and after a specific module has run. You can also configure a span of time in which the system is inactive ("sleeps") before and after each module.

`pre-modules`

Defines scripts and sleep time executed before a configuration module starts. The following code shows an example setting the sleep time to ten seconds and executing an echo command before running the network configuration module.

```
<general>
 <wait>
  <pre-modules config:type="list">
   <module>
    <name>networking</name>
    <sleep>
     <time config:type="integer">10</time>
     <feedback config:type="boolean">true</feedback>
    </sleep>
    <script>
     <source>echo foo</source>
     <debug config:type="boolean">false</debug>
    </script>
   </module>
  </pre-modules>
  ...
<general>
```

`post-modules`

Defines scripts and sleep time executed after a configuration module starts. The following code shows an example setting the sleep time to ten seconds and executing an echo command after running the network configuration module.

```
<general>
 <wait>
  <post-modules config:type="list">
   <module>
    <name>networking</name>
    <sleep>
     <time config:type="integer">10</time>
     <feedback config:type="boolean">true</feedback>
    </sleep>
    <script>
     <source>echo foo</source>
     <debug config:type="boolean">false</debug>
    </script>
   </module>
  </post-modules>
  ...
<general>
```

4.1.8 Blacklisting Unused Devices on IBM Z

On IBM Z you can prevent the kernel from looking at unused hardware devices, by running **`cio_ignore`** and blacklisting them. This is done by setting the AutoYaST parameter with the same name to `true`. Setting this value is optional and only applies to installations on IBM Z hardware. The default is `false`.

```
<general>
 <cio_ignore config:type="boolean">false</cio_ignore>
 ...
<general>
```

4.1.9 Examples for the `general` Section

Find examples covering several use cases in this section.

EXAMPLE 4.1: GENERAL OPTIONS

This example shows the most commonly used options in the general section. The scripts in the pre and post module sections are only dummy scripts illustrating the concept.

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns"
 xmlns:config="http://www.suse.com/1.0/configns">
 <general>
  <! -- Use cio_ignore on IBM &zseries; only -->
  <cio_ignore config:type="boolean">false</cio_ignore>
  <mode>
   <halt config:type="boolean">false</halt>
   <forceboot config:type="boolean">false</forceboot>
   <final_reboot config:type="boolean">false</final_reboot>
   <final_halt config:type="boolean">false</final_halt>
   <confirm_base_product_license config:type="boolean">
    false
   </confirm_base_product_license>
   <confirm config:type="boolean">true</confirm>
   <second_stage config:type="boolean">true</second_stage>
  </mode>
  <proposals config:type="list">
   <proposal>partitions_proposal</proposal>
  </proposals>
  <self_update config:type="boolean">true</self_update>
  <self_update_url>http://example.com/updates/$arch</self_update_url>
  <signature-handling>
```

```
   <accept_unsigned_file config:type="boolean">
    true
   </accept_unsigned_file>
   <accept_file_without_checksum config:type="boolean">
    true
   </accept_file_without_checksum>
   <accept_verification_failed config:type="boolean">
    true
   </accept_verification_failed>
   <accept_unknown_gpg_key config:type="boolean">
    true
   </accept_unknown_gpg_key>
   <import_gpg_key config:type="boolean">true</import_gpg_key>
   <accept_non_trusted_gpg_key config:type="boolean">
   true
   </accept_non_trusted_gpg_key>
  </signature-handling>
  <storage>
   <partition_alignment config:type="symbol">
    align_cylinder
   </partition_alignment>
  </storage>
  <wait>
   <pre-modules config:type="list">
    <module>
     <name>networking</name>
     <sleep>
      <time config:type="integer">10</time>
      <feedback config:type="boolean">true</feedback>
     </sleep>
     <script>
     <source>&gt;![CDATA[
echo "Sleeping 10 seconds"
      ]]&gt;</source>
     <debug config:type="boolean">false</debug>
     </script>
    </module>
   </pre-modules>
   <post-modules config:type="list">
    <module>
     <name>networking</name>
     <sleep>
      <time config:type="integer">10</time>
      <feedback config:type="boolean">true</feedback>
     </sleep>
     <script>
      <source>&gt;![CDATA[
```

```
echo "Sleeping 10 seconds"
      ]]&gt;</source>
      <debug config:type="boolean">false</debug>
     </script>
    </module>
   </post-modules>
  </wait>
 </general>
</profile>
```

4.2 Reporting

The `report` resource manages three types of pop-ups that may appear during installation:

- message pop-ups (usually non-critical, informative messages),
- warning pop-ups (if something might go wrong),
- error pop-ups (in case an error occurs).

EXAMPLE 4.2: REPORTING BEHAVIOR

```
<report>
  <errors>
    <show config:type="boolean">true</show>
    <timeout config:type="integer">0</timeout>
    <log config:type="boolean">true</log>
  </errors>
  <warnings>
    <show config:type="boolean">true</show>
    <timeout config:type="integer">10</timeout>
    <log config:type="boolean">true</log>
  </warnings>
  <messages>
    <show config:type="boolean">true</show>
    <timeout config:type="integer">10</timeout>
    <log config:type="boolean">true</log>
  </messages>
  <yesno_messages>
    <show config:type="boolean">true</show>
    <timeout config:type="integer">10</timeout>
    <log config:type="boolean">true</log>
  </yesno_messages>
</report>
```

Depending on your experience, you can skip, log and show (with timeout) those messages. It is recommended to show all `messages` with timeout. Warnings can be skipped in some places but should not be ignored.

The default setting in auto-installation mode is to show errors without timeout and to show all warnings/messages with a timeout of 10 seconds.

Warning: Critical System Messages

Note that not all messages during installation are controlled by the `report` resource. Some critical messages concerning package installation and partitioning will show up ignoring your settings in the `report` section. Usually those messages will need to be answered with *Yes* or *No*.

4.3 System Registration and Extension Selection

Registering the system with the can be configured within the `suse_register` resource. The following example registers the system with the SUSE Customer Center. In case your organization provides its own registration server, you need to specify the required data with the `reg_server*` properties. Refer to the table below for details.

```
<suse_register>
  <do_registration config:type="boolean">true</do_registration>
  <email>tux@example.com</email>
  <reg_code>MY_SECRET_REGCODE</reg_code>
  <install_updates config:type="boolean">true</install_updates>
  <slp_discovery config:type="boolean">false</slp_discovery>
</suse_register>
```

As an alternative to the fully automated registration, AutoYaST can also be configured to start the YaST registration module during the installation. this offers the possibility to enter the registration data manually. The following XML code is required:

```
<general>
 <semi-automatic config:type="list">
   <semi-automatic_entry>scc</semi-automatic_entry>
 </semi-automatic>
</general>
```

Tip: Using the Installation Network Settings

In case you need to use the same network settings that were used for the installation, AutoYaST needs to run the network setup in stage 1 right before the registration is started:

```
<networking>
  <setup_before_proposal config:type="boolean">true</setup_before_proposal>
</networking>
```

Element	Description	Comment
`do_registration`	Boolean `<do_registration config:type="boolean">true</do_registration>`	Specify whether the system should be registered or not. If set to `false` all other options are ignored and the system is not registered.
`e-mail`	E-mail address `<email>tux@example.com</email>`	Optional. The e-mail address matching the registration code.
`reg_code`	Text `<reg_code>SECRET_REGCODE</reg_code>`	Required. Registration code.
`install_updates`	Boolean `<install_updates config:type="boolean">true</install_updates>`	Optional. Determines if updates from the Updates channels should be installed. The default value is to not install them (`false`).
`slp_discovery`	Boolean `<slp_discovery config:type="boolean">true</slp_discovery>`	Optional. Search for a registration server via SLP. The default value is `false`.

Element	Description	Comment
		Expects to find a single server. If more than one server is found, the installation will fail. In case there is more than one registration server available, you need to specify one with `reg_server`. If neither `slp_discovery` nor `reg_server` are set, the system is registered with the SUSE Customer Center. This setting also affects the self-update feature: If it is disabled, no SLP search will be performed.

`reg_server`	URL
	`<reg_server>`
	`  https://smt.example.com`
	`</reg_server>`
		Optional. SMT server URL. If neither `slp_discovery` nor `reg_server` are set, the system is registered with the SUSE Customer Center.
		The SMT server is queried for a URL of the self-update repository. So if `self_update_url` is not set, the SMT server influences where the self-updates are downloaded from. Check out the *Deployment Guide* to find further information about this feature.

`reg_server_cert_fingerprint_type`	`SHA1` or `SHA256`
	`<reg_server_cert_fingerprint_type>`
	`  SHA1`
		Optional. Requires a checksum value provided with `reg_server_cert_finger-`

Element	Description	Comment
	`</reg_server_cert_fingerprint_type>`	`print`. Using the fingerprint is recommended, since it ensures the SSL certificate is verified. The matching certificate will be automatically imported when the SSL communication fails because of a verification error.

`reg_server_cert_fingerprint`	Server Certificate Fingerprint value in hexadecimal notion (case-insensitive).
	`<reg_server_cert_fingerprint>`
	`  01:AB...:EF`
	`</reg_server_cert_fingerprint>`
		Optional. Requires a fingerprint type value provided with `reg_server_cert_fingerprint_type`. Using the fingerprint is recommended, since it ensures the SSL certificate is verified. The matching certificate will be automatically imported when the SSL communication fails because of a verification error.

`reg_server_cert`	URL
	`<reg_server_cert>`
	`  http://smt.example.com/smt.crt`
	`</reg_server_cert>`
		Optional. URL of the SSL certificate on the server. Using this option is not recommended, since the certificate that is downloaded is not verified. Use `reg_server_cert_fingerprint` instead.

`addons`	Add-ons list	Specify an extension from the registration server that should be added to the instal-

Element	Description	Comment
		lation repositories. See *Section 4.3.1, "Extensions"* for details.

Tip: Creating Own Server Certificate Fingerprint

In order to generate a new Server Certificate Fingerprint for the entry `reg_server_cert_fingerprint` you can use the following command:

```
echo | openssl s_client -showcerts -connect scc.suse.com:443 2> /dev/null |
openssl x509 -noout -fingerprint -sha256
```

Replace `scc.suse.com` with your server, optionally use `-sha1` instead of `-sha256` (but the SHA256 default is better).

NOTE: This can be used in a trusted network only! In a non-trusted network (or over Internet) you should get the fingerprint directly on the server. Or verify that the downloaded certificate is exactly as on the server.

4.3.1 Extensions

The SUSE Customer Center provides several extensions, such as `sle-sdk` (SUSE Software Development Kit - SDK) that can be included as additional sources during the installation. Extensions can be added via the `addons` property within the `suse_register` block.

Note: Availability of Extensions

The availability of extensions is product and architecture dependent. Not all extensions are available on other architectures. The only extension available for SUSE Linux Enterprise Desktop is the `sle-sdk`.

Some extensions, such as the `sle-we`, `sle-ha` and `sle-ha-geo` require a registration code.

With **`SUSEConnect --list-extensions`** (available since SLES 12 SP1), you can list all available extensions in a registered system. The result looks like:

```
Install with: SUSEConnect -p sle-sdk/12.2/x86_64
```

The `-p` argument displays the *`NAME/VERSION/ARCH`* values that can be used in the AutoYaST profile as follows:

```
<addons config:type="list">
 <addon>
  <!-- SUSE Linux Enterprise Software Development Kit -->
  <name>sle-sdk</name>
  <version>12.2</version>
  <arch>x86_64</arch>
 </addon>
</addons>
```

For background information and add-on listings for SLES 12, 12 SP1, and 12 SP2, see https://github.com/yast/yast-registration/wiki/Available-SCC-Extensions-for-Use-in-Autoyast. The listings will get updated from time to time.

4.4 The Boot Loader

This documentation is for **`yast2-bootloader`** and applies to GRUB 2. For older product versions shipping with legacy GRUB, refer to the documentation that comes with your distribution in `/usr/share/doc/packages/autoyast2/`

The general structure of the AutoYaST boot loader part looks like the following:

```
<bootloader>
  <loader_type>
    <!-- boot loader type (grub2 or grub2-efi) -->
  </loader_type>
  <global>
    <!--
      entries defining the installation settings for GRUB 2 and
      the generic boot code
    -->
  </global>
  <device_map config:type="list">
    <!-- entries defining the order of devices -->
  </device_map>
 </bootloader>
```

4.4.1 Loader Type

Define which boot loader to use: `grub2` or `grub2-efi`.

```
<loader_type>grub2</loader_type>
```

4.4.2 Globals

This is an important if optional part. Define here where to install GRUB 2 and how the boot process will work. Again, **`yast2-bootloader`** proposes a configuration if you do not define one. Usually the AutoYaST control file includes only this part and all other parts are added automatically during installation by **`yast2-bootloader`**. Unless you have some special requirements, do not specify the boot loader configuration in the XML file.

```
<global>
  <activate config:type="boolean">true</activate>
  <timeout config:type="integer">10</timeout>
  <suse_btrfs config:type="boolean">true</suse_btrfs>
  <terminal>gfxterm</terminal>
  <gfxmode>1280x1024x24</gfxmode>
</global>
```

Attribute	Description
`activate`	Set the boot flag on the boot partition. The boot partition can be `/` if there is no separate /boot partition. If the boot partition is on a logical partition, the boot flag is set to the extended partition. `<activate config:type="boolean">true</activate>`
`append`	Kernel parameters added at the end of boot entries for normal and recovery mode. `<append>nomodeset vga=0x317</append>`

Attribute	Description
`boot_boot`	Write GRUB 2 to a separate /boot partition. If no separate /boot partition exists, GRUB 2 will be written to `/`. `<boot_boot>false</boot_boot>`
`boot_custom`	Write GRUB 2 to a custom device. `<boot_custom>/dev/sda3</boot_custom>`
`boot_extended`	Write GRUB 2 to the extended partition (important if you want to use a generic boot code and the `/boot` partition is logical). NOTE: if the boot partition is logical, you should use `boot_mbr` (write GRUB 2 to MBR) rather than `generic_mbr`. `<boot_extended>false</boot_extended>`
`boot_mbr`	Write GRUB 2 to MBR of the first disk in the order (device.map includes order of disks). `<boot_mbr>false</boot_mbr>`
`boot_root`	Write GRUB 2 to `/` partition. `<boot_root>false</boot_root>`
`generic_mbr`	Write generic boot code to MBR, will be ignored if boot_mbr is set to `true`. `<generic_mbr config:type="boolean">false</generic_mbr>`
`gfxmode`	Graphical resolution of the GRUB 2 screen (requires <terminal> to be set to `gfxterm`. Valid entries are `auto`, *`HORIZON-`*

Attribute	Description
	*TAL*x*VERTICAL*, or *HORIZONTAL*x*VERTICAL*x-*COLOR DEPTH*. You can see the screen resolutions supported by GRUB 2 on a particular system by using the **vbeinfo** command at the GRUB 2 command line in the running system. `<gfxmode>1280x1024x24</gfxmode>`
os_prober	If set to true, automatically searches for operating systems already installed and generates boot entries for them during the installation `<os_prober config:type="boolean">false</ os_prober>`
cpu_mitigations	Allows to choose a default setting of kernel boot command line parameters for CPU mitigation (and at the same time strike a balance between security and performance). Possible values are: auto. Enables all mitigations required for your CPU model, but does not protect against cross-CPU thread attacks. This setting may impact performance to some degree, depending on the workload. nosmt. Provides the full set of available security mitigations. Enables all mitigations required for your CPU model. In addition, it disables Simultaneous Multithreading (SMT) to avoid side-channel attacks across multiple CPU threads. This setting may further impact performance, depending on the workload.

Attribute	Description
	`off`. Disables all mitigations. Side-channel attacks against your CPU are possible, depending on the CPU model. This setting has no impact on performance. `manual` . Does not set any mitigation level. Specify your CPU mitigations manually by using the kernel command line options. `<cpu_mitigations>auto</cpu_mitigations>` If not set in AutoYaST, the respective settings can be changed via kernel command line. By default, the (product-specific) settings in the `/control.xml` file on the installation medium are used (if nothing else is specified).
`suse_btrfs`	Obsolete and no longer used. Booting from Btrfs snapshots is automatically enabled since SLES 12 SP2.

`serial`	Command to execute if the GRUB 2 terminal mode is set to `serial`.
	
	`<serial>`
	`  serial --speed=115200 --unit=0 --word=8`
	`  --parity=no --stop=1`
	`</serials>`

`terminal`	Specify the GRUB 2 terminal mode to use, Valid entries are `console`, `gfxterm`, and `serial`. If set to `serial`, the serial command needs to be specified with <serial>, too. `<terminal>serial</terminal>`

Attribute	Description
`timeout`	The timeout in seconds until the default boot entry is booted automatically. `<timeout config:type="integer">10</timeout>`
`trusted_boot`	If set to `true`, then Trusted GRUB is used. Trusted GRUB supports Trusted Platform Module (TPM). Works only for `grub2` boot-loader. `<trusted_boot">true</trusted_boot>`
`vgamode`	Adds the kernel parameter `vga=VALUE` to the boot entries. `<vgamode>0x317</vgamode>`
`xen-append`	Kernel parameters added at the end of boot entries for Xen guests. `<append>nomodeset vga=0x317</append>`
`xen-kernel-append`	Kernel parameters added at the end of boot entries for Xen kernels on the VM Host Server. `<xen-append>dom0_mem=768M</xen-append>`

4.4.3 Device map

GRUB 2 avoids mapping problems between BIOS drives and Linux devices by using device ID strings (UUIDs) or file system labels when generating its configuration files. GRUB 2 utilities create a temporary device map on the fly, which is usually sufficient, particularly on single-disk systems. However, if you need to override the automatic device mapping mechanism, create your custom mapping in this section.

```
<device_map config:type="list">
  <device_map_entry>
    <firmware>hd0</firmware> <!-- order of devices in target map  -->
    <linux>/dev/disk/by-id/ata-ST3500418AS_6VM23FX0</linux> <!-- name of device (disk)
 -->
  </device_map_entry>
</device_map>
```

4.5 Partitioning

4.5.1 Drive Configuration

The elements listed below must be placed within the following XML structure:

```
<profile>
  <partitioning config:type="list">
    <drive>
     ...
    </drive>
  </partitioning>
</profile>
```

Attribute	Values	Description
`device`	The device you want to configure in this <drive> section. You can use persistent device names via id, like `/dev/disk/by-id/ata-WD-C_WD3200AAKS-75L9A0_WD-WMAV27368122` or *by-path*,like `/dev/disk/by-path/pci-0001:00:03.0-sc-si-0:0:0:0`. `<device>/dev/sda</device>`	Optional. If left out, AutoYaST tries to guess the device. See *Tip: Skipping Devices* on how to influence guessing. A RAID must always have `/dev/md` as device.

Attribute	Values	Description
`initialize`	If set to `true`, the partition table gets wiped out before AutoYaST starts the partition calculation. `<initialize config:type="boolean">true</initialize>`	Optional. The default is `false`.
`partitions`	A list of <partition> entries (see *Section 4.5.2, "Partition Configuration"*). `<partitions config:type="list"> <partition>...</partition> ... </partitions>`	Optional. If no partitions are specified, AutoYaST will create a reasonable partitioning (see *Section 4.5.5, "Automated Partitioning"*).
`pesize`	This value only makes sense with LVM. `<pesize>8M</pesize>`	Optional. Default is 4M for LVM volume groups.
`use`	Specifies the strategy AutoYaST will use to partition the hard disk. Choose between: • `all` (uses the whole device while calculating the new partitioning), • `linux` (only existing Linux partitions are used),	This parameter should be provided.

Attribute	Values	Description
	• `free` (only unused space on the device is used, no other partitions are touched), • 1,2,3 (a list of comma separated partition numbers to use).	
`type`	Specify the type of the `drive`, Choose between: • `CT_DISK` for physical hard disks (default), • `CT_LVM` for LVM volume groups, • `CT_DMMULTIPATH` for multipath devices. See *Section 4.5.10, "Multipath Support"* for further information. `<type config:type="symbol">CT_LVM</type>`	Optional. Default is `CT_DISK` for a normal physical hard disk.
`disklabel`	Describes the type of the partition table. Choose between: • `msdos` • `gpt` `<disklabel>gpt</disklabel>`	Optional. By default YaST decides what makes sense.

Attribute	Values	Description
`keep_unknown_lv`	This value only makes sense for type = CT_LVM drives. If you are reusing a logical volume group and you set this to `true`, all existing logical volumes in that group will not be touched unless they are specified in the <partitioning> section. So you can keep existing logical volumes without specifying them. `<keep_unknown_lv config:type="boolean">false</keep_unknown_lv>`	Optional. The default is `false`.
`enable_snapshots`	Enables snapshots on Btrfs file systems mounted at `/` (does not apply to other file systems, or Btrfs file systems not mounted at `/`). `<enable_snapshots config:type="boolean">false</enable_snapshots>`	Optional. The default is `true`.

Tip: Skipping Devices

You can influence AutoYaST's device-guessing for cases where you do not specify a <device> entry on your own. Usually AutoYaST would use the first device it can find that looks reasonable but you can configure it to skip some devices like this:

```
<partitioning config:type="list">
  <drive>
    <initialize config:type="boolean">true</initialize>
    <skip_list config:type="list">
      <listentry>
```

```
        <!-- skip devices that use the usb-storage driver -->
        <skip_key>driver</skip_key>
        <skip_value>usb-storage</skip_value>
      </listentry>
      <listentry>
        <!-- skip devices that are smaller than 1GB -->
        <skip_key>size_k</skip_key>
        <skip_value>1048576</skip_value>
        <skip_if_less_than config:type="boolean">true</skip_if_less_than>
      </listentry>
      <listentry>
        <!-- skip devices that are larger than 100GB -->
        <skip_key>size_k</skip_key>
        <skip_value>104857600</skip_value>
        <skip_if_more_than config:type="boolean">true</skip_if_more_than>
      </listentry>
    </skip_list>
  </drive>
</partitioning>
```

For a list of all possible <skip_key>, run **`yast2 ayast_probe`** on an already installed system.

4.5.2 Partition Configuration

The elements listed below must be placed within the following XML structure:

```
<drive>
  <partitions config:type="list">
    <partition>
      ...
    </partition>
  </partitions>
</drive>
```

Attribute	Values	Description
`create`	Specify if this partition must be created or if it already exists. `<create config:type="boolean" >false</create>`	If set to `false`, you also need to set `partition_nr` to tell AutoYaST the partition number.

Attribute	Values	Description
`crypt_fs`	Partition will be encrypted. `<crypt_fs config:type="boolean">false</crypt_fs>`	Default is `false`.
`crypt_key`	Encryption key `<crypt_key>xxxxxxxx</crypt_key>`	Only needed if `crypt_fs` has been set to `true`.

`mount`	The mount point of this partition.
	`<mount>/</mount>`
	`<mount>swap</mount>`
		You should have at least a root partition (/) and a swap partition.

`fstopt`	Mount options for this partition.
	`<fstopt>`
	`ro,noatime,user,data=ordered,acl,user_xattr`
	`</fstopt>`
		See **`man mount`** for available mount options.

`label`	The label of the partition (useful for the `mountby` parameter; see below). `<label>mydata</label>`	See **`man e2label`** for an example.
`uuid`	The UUID of the partition (only useful for the `mountby` parameter; see below). `<uuid >1b4e28ba-2fa1-11d2-883f-b9a761bde3fb</uuid>`	See **`man uuidgen`**.
`size`	The size of the partition, for example 4G, 4500M, etc. The /boot partition and the swap partition can have `auto` as size. Then AutoYaST calculates a reasonable size. One partition can have the value `max` to use all remaining space.	

Attribute	Values	Description
	You can also specify the size in percentage. So 10% will use 10% of the size of the hard disk or volume group. You can mix auto, max, size, and percentage as you like. `<size>10G</size>`	
`format`	Specify if AutoYaST should format the partition. `<format config:type="boolean">false</format>`	If you set `create` to `true`, then you likely want this option set to `true` as well.
`file system`	Specify the file system to use on this partition: • `btrfs` • `ext2` • `ext3` • `ext4` • `fat` • `xfs` • `swap` `<filesystem config:type="symbol">ext3</filesystem>`	Optional. The default is `btrfs` for the root partition (`/`) and `xfs` for data partitions.
`mkfs_options`	Specify an option string that is added to the mkfs command. `<mkfs_options>-I 128</mkfs_options>`	Optional. Only use this when you know what you are doing.

Attribute	Values	Description
`partition_nr`	The partition number of this partition. If you have set `create=false` or if you use LVM, then you can specify the partition via `partition_nr`. You can force AutoYaST to only create primary partitions by assigning numbers below 5. `<partition_nr config:type="integer" >2</partition_nr>`	Usually, numbers 1 to 4 are primary partitions while 5 and higher are logical partitions.
`partition_id`	The `partition_id` sets the id of the partition. If you want different identifiers than 131 for Linux partition or 130 for swap, configure them with partition_id. `<partition_id config:type="integer" >131</partition_id>` Possible values are: Swap: `130` Linux: `131` LVM: `142` MD RAID: `253` EFI partition: `259`	The default is `131` for Linux partition and `130` for swap.
`partition_type`	When using an `msdos` partition table, this element sets the type of the partition. The value can be `primary` or `logical`. This value is ignored when using a `gpt` partition table, because such a distinction does not exist in that case. `<partition_type>primary</partition_type>`	Optional. Allowed values are `primary` (default) and `logical`.

Attribute	Values	Description

mountby	Instead of a partition number, you can tell AutoYaST to mount a partition by device, label, uuid, path or id, which are the udev path and udev id (see /dev/disk/...).
	
	`<mountby config:type="symbol"`
	`>label</mountby>`
		See label and uuid documentation above. The default depends on YaST and usually is id.

subvolumes	List of subvolumes to create for a file system of type Btrfs. This key only makes sense for file systems of type Btrfs. See *Section 4.5.3, "Btrfs subvolumes"* for more information.
	
	`<subvolumes config:type="list">`
	`  <path>tmp</path>`
	`  <path>opt</path>`
	`  <path>srv</path>`
	`  <path>var/crash</path>`
	`  <path>var/lock</path>`
	`  <path>var/run</path>`
	`  <path>var/tmp</path>`
	`  <path>var/spool</path>`
	`  ...`
	`</subvolumes>`	

lv_name	If this partition is on a logical volume in a volume group, specify the logical volume name here (see the is_lvm_vg parameter in the drive configuration). `<lv_name>opt_lv</lv_name>`	
stripes	An integer that configures LVM striping. Specify across how many devices you want to stripe (spread data). `<stripes config:type="integer">2</stripes>`	

Attribute	Values	Description
stripesize	Specify the size of each block in KB. <stripesize config:type="integer" >4</stripesize>	
lvm_group	If this is a physical partition used by (part of) a volume group (LVM), you need to specify the name of the volume group here. <lvm_group>system</lvm_group>	
pool	pool must be set to true if the LVM logical volume should be an LVM thin pool. <pool config:type="boolean">false</pool>	
used_pool	The name of the LVM thin pool that is used as a data store for this thin logical volume. If this is set to something non-empty, it implies that the volume is a so-called thin logical volume. <used_pool>my_thin_pool</used_pool>	
raid_name	If this physical volume is part of a RAID, specify the name of the RAID. <raid_name>/dev/md0</raid_name>	
raid_type	Specify the type of the RAID. <raid_type>raid1</raid_type>	
raid_options	Specify RAID options, see below. <raid_options>...</raid_options>	

Attribute	Values	Description
`resize`	This boolean must be `true` if an existing partition should be resized. In this case, you want to set `create` to `false` and usually you do not want to `format` the partition. You need to tell AutoYaST the `partition_nr` and the `size`. The size can be in percentage of the original size or a number, like `800M`. `max` and `auto` do not work as size here. `<resize config:type="boolean" >false</resize>`	Resizing only works with physical disks, not with LVM volumes.

4.5.3 Btrfs subvolumes

As mentioned in *Section 4.5.2, "Partition Configuration"*, it is possible to define a set of subvolumes for each Btrfs file system. In its simplest form, is just a list of entries:

```
<subvolumes config:type="list">
  <path>tmp</path>
  <path>opt</path>
  <path>srv</path>
  <path>var/crash</path>
  <path>var/lock</path>
  <path>var/run</path>
  <path>var/tmp</path>
  <path>var/spool</path>
</subvolumes>
```

AutoYaST supports disabling copy-on-write for a given subvolume. In that case, a slightly more complex syntax should be used:

```
<subvolumes config:type="list">
<listentry>tmp</listentry>
<listentry>opt</listentry>
<listentry>srv</listentry>
<listentry>
  <path>var/lib/pgsql</path>
  <copy_on_write config:type="boolean">false</copy_on_write>
```

```
</listentry>
</subvolumes>
```

If there is a default subvolume used for the distribution (for example `@` in SUSE Linux Enterprise Server), the name of this default subvolume is automatically prefixed to the names in this list. This behavior can be disabled by setting the `btrfs_set_default_subvolume_name` in the `general/storage` section.

```
<general>
  <storage>
    <btrfs_set_default_subvolume_name config:type="boolean">false</
btrfs_set_default_subvolume_name>
  </storage>
</general>
```

4.5.4 RAID Options

The following elements must be placed within the following XML structure:

```
<partition>
  <raid_options>
    ...
  </raid_options>
</partition>
```

Attribute	Values	Description
`chunk_size`	`<chunk_size>4</chunk_size>`	
`parity_algorithm`	Possible values are: `left_asymmetric`, `left_symmetric`, `right_asymmetric`, `right_symmetric`. For RAID6 and RAID10 the following values can be used: `parity_first`, `parity_last`, `left_asymmetric_6`, `left_symmetric_6`,	

Attribute	Values	Description
	`right_asymmetric_6`, `right_symmetric_6`, `parity_first_6`, `n2`, `o2`, `f2`, `n3`, `o3`, `f3` for RAID6 and RAID10. `<parity_algorithm>left_asymmetric</parity_algorithm>`	
`raid_type`	Possible values are: `raid0`, `raid1`, `raid5`, `raid6` and `raid10`. `<raid_type>raid1</raid_type>`	The default is `raid1`.
`device_order`	This list contains the optional order of the physical devices: `<device_order config:type="list"> <device>/dev/sdb2</device> <device>/dev/sda1</device> ... </device_order>`	This is optional and the default is alphabetical order.

4.5.5 Automated Partitioning

For automated partitioning, you only need to provide the sizes and mount points of partitions. All other data needed for successful partitioning is calculated during installation—unless provided in the control file.

If no partitions are defined and the specified drive is also the drive where the root partition should be created, the following partitions are created automatically:

- `/boot`

The size of the `/boot` partition is determined by the architecture of the target system.

- `swap`
  The size of the swap partition is determined by the amount of memory available in the system.

- `/` (root partition)
  The size of the root partition is determined by the space left after creating `swap` and `/boot`.

Depending on the initial status of the drive and how it was previously partitioned, it is possible to create the default partitioning in the following ways:

Use Free Space
: If the drive is already partitioned, it is possible to create the new partitions using the free space on the hard disk. This requires the availability of sufficient space for all selected packages in addition to swap.

Reuse all available space
: Use this option to delete all existing partitions (Linux and non-Linux).

Reuse all available Linux partitions
: This option deletes all existing Linux partitions. Other partitions (for example Windows partitions) remain untouched. Note that this works only if the Linux partitions are at the end of the device.

Reuse only specified partitions
: This option allows you to select specific partitions to delete. Start the selection with the last available partition.

Repartitioning only works if the selected partitions are neighbors and located at the end of the device.

Important: Beware of Data Loss

The value provided in the `use` property determines how existing data and partitions are treated. The value `all` means that the entire disk will be erased. Make backups and use the `confirm` property if you need to keep some partitions with important data. Otherwise, no pop-ups will notify you about partitions being deleted.

If multiple drives are in the target system, identify all drives with their device names and specify how the partitioning should be performed.

Partition sizes can be given in gigabytes, megabytes or can be set to a flexible value using the keywords `auto` and `max`. `max` uses all available space on a drive, therefore should only be set for the last partition on the drive. With `auto` the size of a `swap` or `boot` partition is determined automatically, depending on the memory available and the type of the system.

A fixed size can be given as shown below:

`1GB`, `1G`, `1000MB`, or `1000M` will all create a partition of the size 1 Gigabyte.

EXAMPLE 4.3: AUTOMATED PARTITIONING

> The following is an example of a single drive system, which is not pre-partitioned and should be automatically partitioned according to the described pre-defined partition plan. If you do not specify the device, it will be automatically detected.
>
> ```
> <partitioning  config:type="list">
>   <drive>
>     <device>/dev/sda</device>
>     <use>all</use>
>   </drive>
> </partitioning>
> ```

A more detailed example shows how existing partitions and multiple drives are handled.

EXAMPLE 4.4: DETAILED AUTOMATED PARTITIONING

```
<partitioning  config:type="list">
  <drive>
    <device>/dev/sda</device>
    <partitions config:type="list">
      <partition>
        <mount>/</mount>
        <size>10G</size>
      </partition>
      <partition>
        <mount>swap</mount>
        <size>1G</size>
      </partition>
    </partitions>
  </drive>
  <drive>
    <device>/dev/sdb</device>
```

```
    <use>all</use>
    <partitions config:type="list">
      <partition>
        <filesystem  config:type="symbol">reiser</filesystem>
        <mount>/data1</mount>
        <size>15G</size>
      </partition>
      <partition>
        <filesystem  config:type="symbol">jfs</filesystem>
        <mount>/data2</mount>
        <size>auto</size>
      </partition>
    </partitions>
    <use>free</use>
  </drive>
</partitioning>
```

4.5.6 Advanced Partitioning Features

4.5.6.1 Wipe out Partition Table

Usually this is not needed because AutoYaST can delete partitions one by one automatically. But you need the option to let AutoYaST clear the partition table instead of deleting partitions individually.

Go to the `drive` section and add:

```
<initialize config:type="boolean">true</initialize>
```

With this setting AutoYaST will delete the partition table before it starts to analyze the actual partitioning and calculates its partition plan. Of course this means, that you cannot keep any of your existing partitions.

4.5.6.2 Mount Options

By default a file system to be mounted is identified in `/etc/fstab` by the device name. This identification can be changed so the file system is found by searching for a UUID or a volume label. Note that not all file systems can be mounted by UUID or a volume label. To specify how a partition is to be mounted, use the `mountby` property which has the `symbol` type. Possible options are:

- `device` (default)
- `label`
- `UUID`

If you choose to mount the partition using a label, the name entered for the `label` property is used as the volume label.

Add any valid mount option in the fourth field of `/etc/fstab`. Multiple options are separated by commas. Possible fstab options:

Mount read-only (`ro`)

No write access to the file system. Default is `false`.

No access time (`noatime`)

Access times are not updated when a file is read. Default is `false`.

Mountable by User (`user`)

The file system can be mounted by a normal user. Default is `false`.

Data Journaling Mode (`ordered`, `journal`, `writeback`)

`journal`

All data is committed to the journal prior to being written to the main file system.

`ordered`

All data is directly written to the main file system before its metadata is committed to the journal.

`writeback`

Data ordering is not preserved.

Access Control List (`acl`)

Enable access control lists on the file system.

Extended User Attributes (`user_xattr`)

Allow extended user attributes on the file system.

EXAMPLE 4.5: MOUNT OPTIONS

```
<partitions config:type="list">
  <partition>
    <filesystem config:type="symbol">reiser</filesystem>
    <format config:type="boolean">true</format>
    <fstopt>ro,noatime,user,data=ordered,acl,user_xattr</fstopt>
    <mount>/local</mount>
    <mountby config:type="symbol">uuid</mountby>
    <partition_id config:type="integer">131</partition_id>
    <size>10G</size>
  </partition>
</partitions>
```

4.5.6.3 Keeping Specific Partitions

In some cases you should leave partitions untouched and only format specific target partitions, rather than creating them from scratch. For example, if different Linux installations coexist, or you have another operating system installed, likely you do not want to wipe these out. You may also want to leave data partitions untouched.

Such scenarios require certain knowledge about the target systems and hard disks. Depending on the scenario, you might need to know the exact partition table of the target hard disk with partition ids, sizes and numbers. With this data you can tell AutoYaST to keep certain partitions, format others and create new partitions if needed.

The following example will keep partitions 1, 2 and 5 and delete partition 6 to create two new partitions. All remaining partitions will only be formatted.

EXAMPLE 4.6: KEEPING PARTITIONS

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sdc</device>
      <partitions config:type="list">
        <partition>
          <create config:type="boolean">false</create>
          <format config:type="boolean">true</format>
          <mount>/</mount>
          <partition_nr config:type="integer">1</partition_nr>
```

```
        </partition>
        <partition>
          <create config:type="boolean">false</create>
          <format config:type="boolean">false</format>
          <partition_nr config:type="integer">2</partition_nr>
          <mount>/space</mount>
        </partition>
        <partition>
          <create config:type="boolean">false</create>
          <format config:type="boolean">true</format>
          <filesystem config:type="symbol">swap</filesystem>
          <partition_nr config:type="integer">5</partition_nr>
          <mount>swap</mount>
        </partition>
        <partition>
          <format config:type="boolean">true</format>
          <mount>/space2</mount>
          <size>5G</size>
        </partition>
        <partition>
          <format config:type="boolean">true</format>
          <mount>/space3</mount>
          <size>max</size>
        </partition>
      </partitions>
    <use>6</use>
  </drive>
</partitioning>
```

The last example requires exact knowledge of the existing partition table and the partition numbers of those partitions that should be kept. In some cases however, such data may not be available, especially in a mixed hardware environment with different hard disk types and configurations. The following scenario is for a system with a non-Linux OS with a designated area for a Linux installation.

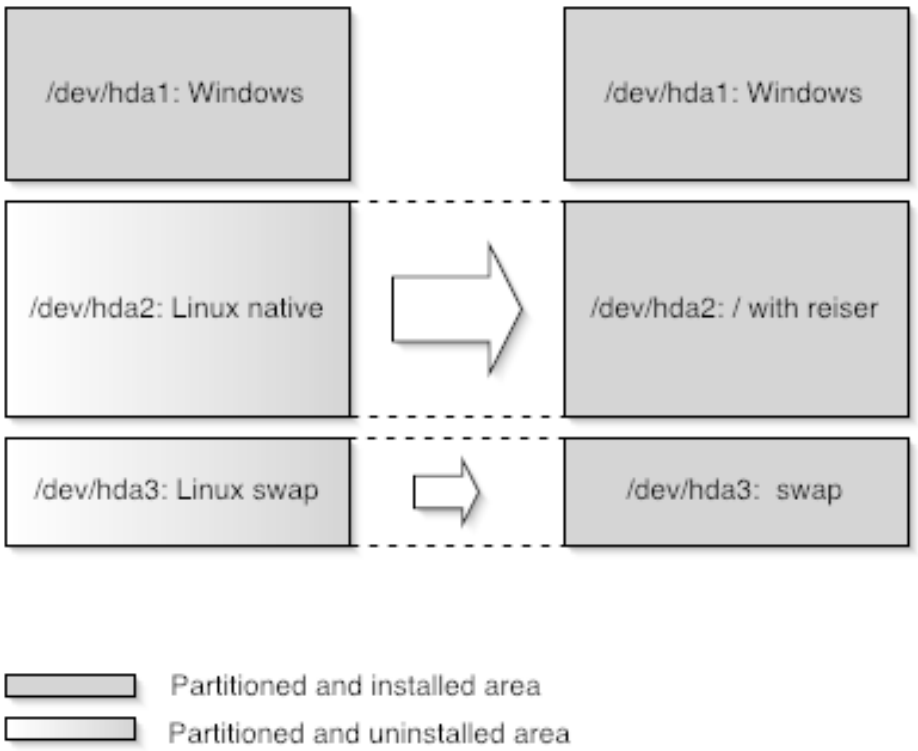

FIGURE 4.1: KEEPING PARTITIONS

In this scenario, shown in figure *Figure 4.1, "Keeping partitions"*, AutoYaST will not create new partitions. Instead it searches for certain partition types on the system and uses them according to the partitioning plan in the control file. No partition numbers are given in this case, only the mount points and the partition types (additional configuration data can be provided, for example file system options, encryption and file system type).

EXAMPLE 4.7: AUTO-DETECTION OF PARTITIONS TO BE KEPT.

```
<partitioning config:type="list">
  <drive>
    <partitions config:type="list">
      <partition>
        <create config:type="boolean">false</create>
        <format config:type="boolean">true</format>
        <mount>/</mount>
        <partition_id config:type="integer">131</partition_id>
      </partition>
      <partition>
        <create config:type="boolean">false</create>
        <format config:type="boolean">true</format>
        <filesystem config:type="symbol">swap</filesystem>
        <partition_id config:type="integer">130</partition_id>
```

```
        <mount>swap</mount>
      </partition>
    </partitions>
  </drive>
</partitioning>
```

4.5.7 Using an Existing Mount Table (fstab)

Note: Cannot Be Combined with `partitioning` Section

This section will be ignored if you have defined your own `partitioning` section too.

This option will allow AutoYaST to use an existing `/etc/fstab` and use the partition data from a previous installation. All partitions are kept and no new partitions are created. The partitions will be formatted and mounted as specified in `/etc/fstab` on a Linux root partition.

Although the default behavior is to format all partitions, it is also possible to leave some partitions (for example data partitions) untouched and only mount them. If multiple installations are found on the system (multiple root partitions with different `fstab` files, the installation will abort, unless the root partition is configured in the control file. The following example illustrates how this option can be used:

EXAMPLE 4.8: READING AN EXISTING `/etc/fstab`

```
<partitioning_advanced>
  <fstab>
    <!-- Read data from existing fstab. If multiple root partitions are
            found, use the one specified below. Otherwise the first root
            partition is taken -->
    <!-- <root_partition>/dev/sda5</root_partition> -->
    <use_existing_fstab config:type="boolean">true</use_existing_fstab>
    <!-- all partitions found in fstab will be formatted and mounted
            by default unless a partition is listed below with different
            settings -->
    <partitions config:type="list">
      <partition>
        <format config:type="boolean">false</format>
        <mount>/bootmirror</mount>
      </partition>
    </partitions>
  </fstab>
</partitioning_advanced>
```

4.5.8 Logical Volume Manager (LVM)

To configure LVM, first create a physical volume using the normal partitioning method described above.

EXAMPLE 4.9: CREATE LVM PHYSICAL VOLUME

The following example shows how to prepare for LVM in the `partitioning` resource. A non-formatted partition is created on device `/dev/sda1` of the type `LVM` and with the volume group `system`. This partition will use all space available on the drive.

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <partitions config:type="list">
      <partition>
        <create config:type="boolean">true</create>
        <lvm_group>system</lvm_group>
        <partition_type>primary</partition_type>
        <partition_id config:type="integer">142</partition_id>
        <partition_nr config:type="integer">1</partition_nr>
        <size>max</size>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
</partitioning>
```

EXAMPLE 4.10: LVM LOGICAL VOLUMES

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <partitions config:type="list">
      <partition>
        <lvm_group>system</lvm_group>
        <partition_type>primary</partition_type>
        <size>max</size>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
  <drive>
    <device>/dev/system</device>
      <is_lvm_vg config:type="boolean">true</is_lvm_vg>
      <partitions config:type="list">
        <partition>
          <filesystem config:type="symbol">reiser</filesystem>
```

```
          <lv_name>user_lv</lv_name>
          <mount>/usr</mount>
          <size>15G</size>
        </partition>
        <partition>
          <filesystem config:type="symbol">reiser</filesystem>
          <lv_name>opt_lv</lv_name>
          <mount>/opt</mount>
          <size>10G</size>
        </partition>
        <partition>
          <filesystem config:type="symbol">reiser</filesystem>
          <lv_name>var_lv</lv_name>
          <mount>/var</mount>
          <size>1G</size>
        </partition>
      </partitions>
      <pesize>4M</pesize>
    <use>all</use>
  </drive>
</partitioning>
```

It is possible to set the `size` to `max` for the logical volumes. Of course, you can only use `max` for one(!) logical volume. You cannot set two logical volumes in one volume group to `max`.

4.5.9 Software RAID

Using AutoYaST, you can create and assemble software RAID devices. The supported RAID levels are the following:

RAID 0

This level increases your disk performance. There is *no* redundancy in this mode. If one of the drives crashes, data recovery will not be possible.

RAID 1

This mode offers the best redundancy. It can be used with two or more disks. An exact copy of all data is maintained on all disks. As long as at least one disk is still working, no data is lost. The partitions used for this type of RAID should have approximately the same size.

RAID 5

This mode combines management of a larger number of disks and still maintains some redundancy. This mode can be used on three disks or more. If one disk fails, all data is still intact. If two disks fail simultaneously, all data is lost.

Multipath

This mode allows access to the same physical device via multiple controllers for redundancy against a fault in a controller card. This mode can be used with at least two devices.

As with LVM, you need to create all RAID partitions first and assign them to the RAID device you want to create afterward. Additionally you need to specify whether a partition or a device should be part of the RAID or if it should be a Spare device.

The following example shows a simple RAID1 configuration:

EXAMPLE 4.11: RAID1 CONFIGURATION

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <partitions config:type="list">
      <partition>
        <partition_id config:type="integer">253</partition_id>
        <format config:type="boolean">false</format>
        <raid_name>/dev/md0</raid_name>
        <raid_type>raid</raid_type>
        <size>4G</size>
      </partition>

        <!-- Insert a configuration for the regular partitions located on
                /dev/sda here (for example / and swap) -->

    </partitions>
    <use>all</use>
  </drive>
  <drive>
    <device>/dev/sdb</device>
    <partitions config:type="list">
      <partition>
        <format config:type="boolean">false</format>
        <partition_id config:type="integer">253</partition_id>
        <raid_name>/dev/md0</raid_name>
        <raid_type>raid</raid_type>
        <size>4gb</size>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
  <drive>
    <device>/dev/md</device>
    <partitions config:type="list">
      <partition>
```

```
        <filesystem config:type="symbol">reiser</filesystem>
        <format config:type="boolean">true</format>
        <mount>/space</mount>
        <partition_id config:type="integer">131</partition_id>
        <partition_nr config:type="integer">0</partition_nr>
        <raid_options>
          <chunk_size>4</chunk_size>
          <parity_algorithm>left-asymmetric</parity_algorithm>
          <raid_type>raid1</raid_type>
        </raid_options>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
</partitioning>
```

Keep the following in mind when configuring a RAID:

- The device for raid is always `/dev/md`
- The property `partition_nr` is used to determine the MD device number. If `partition_nr` is equal to 0, then `/dev/md0` is configured.
- All RAID-specific options are contained in the `raid_options` resource.

4.5.10 Multipath Support

AutoYaST is able to handle multipath devices. In order to take advantage of them, you need to enable multipath support, as described in *Section 4.1.6, "The Storage Section"*, and to set the `type` element of each `drive` section to `CT_DMMULTIPATH`, instead of `CT_DISK`. Mixing `CT_DISK` and `CT_DMMULTIPATH` types will not work.

Example 4.12, "Using Multipath Devices" shows the relevant parts of a profile which instructs AutoYaST to partition a multipath device.

EXAMPLE 4.12: USING MULTIPATH DEVICES

```
<general>
  <storage>
    <start_multipath config:type="boolean">true</start_multipath>
  <storage>
</general>
<partitioning>
```

```
  <drive>
    <partitions config:type="list">
      <partition>
        <size>20G</size>
        <mount>/</mount>
        <filesystem config:type="symbol">ext4</filesystem>
      </partition>
      <partition>
        <size>auto</size>
        <mount>swap</mount>
      </partition>
    </partitions>
    <type config:type="symbol">CT_DMMULTIPATH</type>
    <use>all</use>
  </drive>
</partitioning>
```

4.5.11 IBM Z Specific Configuration

4.5.11.1 Configuring DASD Disks

The elements listed below must be placed within the following XML structure:

```
<dasd>
 <devices config:type="list">
  <listentry>
   ...
  </listentry>
 </devices>
</dasd>
```

tags in the <profile> section. Each disk needs to be configured in a separate <listentry> ... </listentry> section.

Attribute	Values	Description
`device`	`DASD` is the only value allowed `<device>DASD</dev_name>`	

Attribute	Values	Description
`dev_name`	The device (`dasdN`) you want to configure in this section. `<dev_name>/dev/dasda</dev_name>`	Optional but recommended. If left out, AutoYaST tries to guess the device.
`channel`	Channel by which the disk is accessed. `<channel>0.0.0150</channel>`	Mandatory.
`diag`	Enable or disable the use of `DIAG`. Possible values are `true` (enable) or `false` (disable). `<diag config:type="boolean">true</diag>`	Optional.

Important: Partitioning LDL-Formatted MDisks

For AutoYaST to successfully partition an LDL-formatted MDisk, set the parameters below to `false` as follows:

```
<initialize config:type="boolean">false</initialize>
<create config:type="boolean">false</create>
```

4.5.11.2 Configuring zFCP Disks

The following elements must be placed within the following XML structure:

```
<profile>
  <zfcp>
    <devices config:type="list">
      <listentry>
```

```
        ...
      </listentry>
    </devices>
  </zfcp>
<profile>
```

Each disk needs to be configured in a separate `listentry` section.

Attribute	Values
`controller_id`	Channel number `<controller_id >0.0.fc00</controller_id>`

4.6 iSCSI Initiator Overview

Using the `iscsi-client` resource, you can configure the target machine as an iSCSI client.

EXAMPLE 4.13: ISCSI CLIENT

```
  <iscsi-client>
    <initiatorname>iqn.2013-02.de.suse:01:e229358d2dea</initiatorname>
    <targets config:type="list">
      <listentry>
         <authmethod>None</authmethod>
         <portal>192.168.1.1:3260</portal>
         <startup>onboot</startup>
         <target>iqn.2001-05.com.doe:test</target>
         <iface>default</iface>
      </listentry>
    </targets>
    <version>1.0</version>
  </iscsi-client>
```

Attribute	Description
initiatorname	`InitiatorName` is a value from `/etc/isc-si/initiatorname.iscsi`. In case you have iBFT, this value will be added from there and you are only able to change it in the BIOS setup.

Attribute	Description
version	Version of the YaST module. Default: 1.0
targets	List of targets. Each entry contains: `authmethod` Authentication method: None/CHAP `portal` Portal address `startup` Value: manual/onboot `target` Target name `iface` Interface name

4.7 Fibre Channel over Ethernet Configuration (FCoE)

Using the `fcoe_cfg` resource, you can configure a Fibre Channel over Ethernet (FCoE).

EXAMPLE 4.14: FCOE CONFIGURATION

```
<fcoe-client>
  <fcoe_cfg>
    <DEBUG>no</DEBUG>
    <USE_SYSLOG>yes</USE_SYSLOG>
  </fcoe_cfg>
  <interfaces config:type="list">
    <listentry>
      <dev_name>eth3</dev_name>
      <mac_addr>01:000:000:000:42:42</mac_addr>
      <device>Gigabit 1313</device>
      <vlan_interface>200</vlan_interface>
      <fcoe_vlan>eth3.200</fcoe_vlan>
      <fcoe_enable>yes</fcoe_enable>
      <dcb_required>yes</dcb_required>
      <auto_vlan>no</auto_vlan>
      <dcb_capable>no</dcb_capable>
      <cfg_device>eth3.200</cfg_device>
    </listentry>
  </interfaces>
  <service_start>
    <fcoe config:type="boolean">true</fcoe>
    <lldpad config:type="boolean">true</lldpad>
  </service_start>
```

```
  </fcoe-client>
```

Attribute	Description	Values
fcoe_cfg	`DEBUG` is used to enable or disable debugging messages from the fcoe service script and fcoemon. `USE_SYSLOG` messages are sent to the system log if set to yes (data are logged to `/var/log/messages`).	yes/no
interfaces	List of network cards including the status of VLAN and FCoE configuration.	
service_start	Enable or disable the start of the services `fcoe` and `lldpad` at boot time. Starting the service `fcoe` means starting the Fibre Channel over Ethernet service daemon `fcoemon` which controls the FCoE interfaces and establishes a connection with the daemon `lldpad`. The `lldpad` service provides the Link Layer Discovery Protocol agent daemon `lldpad`, which informs `fcoemon` about DCB (Data Center Bridging) features and configuration of the interfaces.	yes/no

4.8 Country Settings

Language, timezone, and keyboard settings.

EXAMPLE 4.15: LANGUAGE

```
<language>
  <language>en_GB</language>
  <languages>de_DE,en_US</languages>
</language>
```

Attribute	Description	Values
`language`	Primary language	A list of available languages can be found under `/usr/share/YaST2/data/languages`
`languages`	Secondary languages separated by commas	A list of available languages can be found under `/usr/share/YaST2/data/languages`

If the configured value for the primary language is unknown, it will be reset to the default, `en_US`.

EXAMPLE 4.16: TIMEZONE

```
<timezone>
  <hwclock>UTC</hwclock>
  <timezone>Europe/Berlin</timezone>
</timezone>
```

Attribute	Description	Values
hwclock	Whether the hardware clock uses local time or UTC	localtime/UTC
timezone	Timezone	A list of available timezones can be found under `/usr/share/YaST2/data/timezone_raw.ycp`

EXAMPLE 4.17: KEYBOARD

```
      <keyboard>
        <keymap>german</keymap>
      </keyboard>
```

Attribute	Description	Values
keymap	Keyboard layout	A list of available keymaps can be found in `/usr/share/YaST2/data/keyboard_raw.ycp`

4.9 Software

4.9.1 Package Selection with Patterns

Patterns are configured like this:

EXAMPLE 4.18: PACKAGE SELECTION IN THE CONTROL FILE WITH PATTERNS

```
<software>
  <patterns config:type="list">
    <pattern>directory_server</pattern>
  </patterns>
  <packages  config:type="list">
    <package>apache</package>
    <package>postfix</package>
  </packages>
  <do_online_update config:type="boolean">true</do_online_update>
</software>
```

4.9.2 Installing Additional/Customized Packages or Products

In addition to the packages available for installation on the DVD-ROMs, you can add external packages including customized kernels. Customized kernel packages must be compatible to the SUSE packages and must install the kernel files to the same locations.

Unlike in earlier in versions, you do not need a special resource in the control file to install custom and external packages. Instead you need to re-create the package database and update it with any new packages or new package versions in the source repository.

A script is provided for this task which will query packages available in the repository and create the package database. Use the command **`/usr/bin/create_package_descr`**. It can be found in the `inst-source-utils` package in the openSUSE Build Service. When creating the database, all languages will be reset to English.

EXAMPLE 4.19: CREATING A PACKAGE DATABASE WITH THE ADDITIONAL PACKAGE INST-SOURCE-UTILS.RPM

The unpacked DVD is located in `/usr/local/DVDs/LATEST`.

```
cp /tmp/inst-source-utils-2016.7.26-1.2.noarch.rpm /usr/local/DVDs/LATEST/suse/
noarch
cd /usr/local/DVDs/LATEST/suse
create_package_descr -d /usr/local/CDs/LATEST/suse
```

In the above example, the directory `/usr/local/CDs/LATEST/suse` contains the architecture dependent (for example `x86_64`) and architecture independent packages (`noarch`). This might look different on other architectures.

The advantage of this method is that you can keep an up-to-date repository with fixed and updated package. Additionally this method makes the creation of custom CD-ROMs easier.

To add your own module such as the SDK (SUSE Software Development Kit), add a file `add_on_products.xml` to the installation source in the root directory.

The following example shows how the SDK module can be added to the base product repository. The complete SDK repository will be stored in the directory `/sdk`.

EXAMPLE 4.20: `add_on_products.xml`

This file describes an SDK module included in the base product.

```
<?xml version="1.0"?>
<add_on_products xmlns="http://www.suse.com/1.0/yast2ns"
   xmlns:config="http://www.suse.com/1.0/configns">
   <product_items config:type="list">
      <product_item>
         <name>SUSE Linux Enterprise Software Development Kit</name>
         <url>relurl:////sdk?alias=SLE_SDK</url>
         <path>/</path>
         <-- Users are asked whether to add such a product -->
```

```
        <ask_user config:type="boolean">false</ask_user>
        <-- Defines the default state of pre-selected state in case of ask_user
 used. -->
        <selected config:type="boolean">true</selected>
      </product_item>
    </product_items>
</add_on_products>
```

With a normal installation now the SDK module will be installed automatically. It will be not done via an AutoYaST installation. An additional entry would be needed for this in the AutoYaST control file `add-on` section.

Besides this special case all other modules, extensions and add-on products can be added from almost every other location during an AutoYaST installation.

EXAMPLE 4.21: ADDING SDK PRODUCT AUTO AUTOYAST CONFIGURATION FILE

```
<add-on>
  <add_on_products config:type="list">
    <listentry>
      <media_url>cd:///sdk</media_url>
      <product>sle-sdk</product>
      <alias>SLES SDK</alias>
      <product_dir>/</product_dir>
      <priority config:type="integer">20</priority>
      <ask_on_error config:type="boolean">false</ask_on_error>
      <confirm_license config:type="boolean">false</confirm_license>
      <name>SUSE Linux Enterprise Software Development Kit</name>
    </listentry>
  </add_on_products>
</add-on>
```

Attribute	Values
`media_url`	Product URL. Can have the prefix `cd:///`, `http://`, `ftp://`,...
`product`	Internal product name if the add-on is a product. The command **`zypper products`** shows the names of installed products.
`alias`	Repository alias name. Defined by the user.

Attribute	Values
`product_dir`	Additional subpath. Optional.
`priority`	Sets the repository libzypp priority. Priority of 1 is the highest. The higher the number the lower the priority. Default is 99.
`ask_on_error`	AutoYaST can ask the user to make add-on products, modules or extensions available instead of reporting a time-out error when no repository can be found at the given location. Set ask_on_error to `true` (the default is `false`).
`confirm_license`	The user has to confirm the license. Default is `false`.
`name`	Repository name. The command **`zypper lr`** shows the names of added repositories.

To use unsigned installation sources with AutoYaST, turn off the checks with the following configuration in your AutoYaST control file.

Note: Unsigned Installation Sources—Limitations

You can only disable signature checking during the first stage of the auto-installation process. In stage two, the installed system's configuration takes precedence over AutoYaST configuration.

The elements listed below must be placed within the following XML structure:

```
<general>
  <signature-handling>
    ...
  </signature-handling>
</general>
```

Default values for all options are `false`. If an option is set to `false` and a package or repository fails the respective test, it is silently ignored and will not be installed. Note that setting any of these options to `true` is a potential security risk. Never do it when using packages or repositories from third party sources.

Attribute	Values
`accept_unsigned_file`	If set to `true`, AutoYaST will accept unsigned files like the content file. `<accept_unsigned_file config:type="boolean">true</accept_unsigned_file>`
`accept_file_without_checksum`	If set to `true`, AutoYaST will accept files without a checksum in the content file. `<accept_file_without_checksum config:type="boolean">true</accept_file_without_checksum>`
`accept_verification_failed`	If set to `true`, AutoYaST will accept signed files even when the verification of the signature failed. `<accept_verification_failed config:type="boolean">true</accept_verification_failed>`
`accept_unknown_gpg_key`	If set to `true`, AutoYaST will accept new gpg keys of the installation sources, for example the key used to sign the content file. `<accept_unknown_gpg_key config:type="boolean">true</accept_unknown_gpg_key>`
`accept_non_trusted_gpg_key`	Set this option to `true` to accept known keys you have not yet trusted.

Attribute	Values
	`<accept_non_trusted_gpg_key config:type="boolean" >true</accept_non_trusted_gpg_key>`
`import_gpg_key`	If set to `true`, AutoYaST will accept and import new gpg keys on the installation source in its database. `<import_gpg_key config:type="boolean" >true</import_gpg_key>`

It is possible to configure the signature handling for each add-on product, module, or extension individually. The following elements must be between the `signature-handling` section of the individual add-on product, module, or extension. All settings are optional. If not configured, the global signature-handling from the `general` section is used.

Attribute	Values
`accept_unsigned_file`	If set to `true`, AutoYaST will accept unsigned files like the content file for this add-on product. `<accept_unsigned_file config:type="boolean" >true</accept_unsigned_file>`
`accept_file_without_checksum`	If set to `true`, AutoYaST will accept files without a checksum in the content file for this add-on. `<accept_file_without_checksum config:type="boolean" >true</accept_file_without_checksum>`
`accept_verification_failed`	If set to `true`, AutoYaST will accept signed files even when the verification of the signature fails.

Attribute	Values
	<accept_verification_failed config:type="boolean" >true</accept_verification_failed>
accept_unknown_gpg_key	If all is set to true, AutoYaST will accept new gpg keys on the installation source. <accept_unknown_gpg_key> <all config:type="boolean">true</all> </accept_unknown_gpg_key> Otherwise you can define single keys too. <accept_unknown_gpg_key> <all config:type="boolean">false</all> <keys config:type="list"> <keyid>3B3011B76B9D6523</keyid> </keys> </accept_unknown_gpg_key>
accept_non_trusted_gpg_key	This means, the key is known, but it is not trusted by you. You can trust all keys by adding: <accept_non_trusted_gpg_key> <all config:type="boolean">true</all> </accept_non_trusted_gpg_key> Or you can trust specific keys: <accept_non_trusted_gpg_key> <all config:type="boolean">false</all> <keys config:type="list"> <keyid>3B3011B76B9D6523</keyid> </keys> </accept_non_trusted_gpg_key>
import_gpg_key	If all is set to true, AutoYaST will accept and import all new gpg keys on the installation source into its database.

Attribute	Values

	`<import_gpg_key>`
	`  <all config:type="boolean">true</all>`
	`</import_gpg_key>`
	
	This can be done for specific keys only:
	
	`<import_gpg_key>`
	`  <all config:type="boolean">false</all>`
	`  <keys config:type="list">`
	`  <keyid>3B3011B76B9D6523</keyid>`
	`  </keys>`
	`</import_gpg_key>`

4.9.3 Kernel Packages

Kernel packages are not part of any selection. The required kernel is determined during installation. If the kernel package is added to any selection or to the individual package selection, installation will mostly fail because of conflicts.

To force the installation of a specific kernel, use the `kernel` property. The following is an example of forcing the installation of the default kernel. This kernel will be installed even if an SMP or other kernel is required.

EXAMPLE 4.22: KERNEL SELECTION IN THE CONTROL FILE

```
<software>
  <kernel>kernel-default</kernel>
  ...
</software>
```

4.9.4 Removing Automatically Selected Packages

Some packages are selected automatically either because of a dependency or because it is available in a selection.

Removing these packages might break the system consistency, and it is not recommended to remove basic packages unless a replacement which provides the same services is provided. The best example for this case are mail transfer agent (MTA) packages. By default, `postfix` will be

selected and installed. To use another MTA like `sendmail`, then postfix can be removed from the list of selected package using a list in the software resource. However, note that sendmail is not shipped with SUSE Linux Enterprise Server. The following example shows how this can be done:

EXAMPLE 4.23: PACKAGE SELECTION IN CONTROL FILE

```
<software>
  <packages  config:type="list">
    <package>sendmail</package>
  </packages>
  <remove-packages  config:type="list">
    <package>postfix</package>
  </remove-packages>
</software>
```

Note: Package Removal Failure

Note that it is not possible to remove a package, that is part of a pattern (see *Section 4.9.1, "Package Selection with Patterns"*). When specifying such a package for removal, the installation will fail with the following error message:

```
The package resolver run failed. Check
      your software section in the AutoYaST profile.
```

4.9.5 Installing Recommended Packages/Patterns

By default all recommended packages/patterns will be installed. To have a minimal installation which includes required packages only, you can switch off this behavior with the flag `install_recommended`. Note that this flag only affects a fresh installation and will be ignored during an upgrade.

```
<software>
  <install_recommended config:type="boolean">false
  </install_recommended>
</software>
```

Default: If this flag has not been set in the configuration file *all* recommended `packages` and *no* recommended `pattern` will be installed.

4.9.6 Installing Packages in Stage 2

To install packages after the reboot during stage two, you can use the `post-packages` element for that:

```
<software>
  <post-packages config:type="list">
    <package>yast2-cim</package>
  </post-packages>
</software>
```

4.9.7 Installing Patterns in Stage 2

You can also install patterns in stage 2. Use the `post-patterns` element for that:

```
<software>
  <post-patterns config:type="list">
    <pattern>apparmor</pattern>
  </post-patterns>
</software>
```

4.9.8 Online Update in Stage 2

You can perform an online update at the end of the installation. Set the boolean `do_online_update` to `true`. Of course this only makes sense if you add an online update repository in the suse-register/customer-center section, for example, or in a post-script. If the online update repository was already available in stage one via the add-on section, then AutoYaST has already installed the latest packages available. If a kernel update is done via online-update, a reboot at the end of stage two is triggered.

```
<software>
  <do_online_update config:type="boolean">true</do_online_update>
</software>
```

4.10 Upgrade

AutoYaST can also be used for doing a system upgrade. Besides upgrade packages, the following sections are supported too:

- `scripts/pre-scripts` Running user scripts very early, before anything else really happens.
- `add-on` Defining an additional add-on product.
- `language` Setting language.
- `timezone` Setting timezone.
- `keyboard` Setting keyboard.
- `software` Installing additional software/patterns. Removing installed packages.
- `suse_register` Running registration process.

To control the upgrade process the following sections can be defined:

EXAMPLE 4.24: UPGRADE AND BACKUP

```
<upgrade>
    <stop_on_solver_conflict config:type="boolean">true</stop_on_solver_conflict>
  </upgrade>
  <backup>
    <sysconfig config:type="boolean">true</sysconfig>
    <modified config:type="boolean">true</modified>
    <remove_old config:type="boolean">true</remove_old>
  </backup>
```

Element	Description	Comment
stop_on_solver_conflict	Halt installation if there are package dependency issues.	
modified	Create backup of modified files.	
sysconfig	Create backup of `/etc/sysconfig` directory.	

Element	Description	Comment
remove_old	Remove backups from previous updates.	

To start the AutoYaST upgrade mode, you need:

PROCEDURE 4.1: STARTING AUTOYAST IN UPGRADE MODE

1. Copy the AutoYaST profile to `/root/autoupg.xml` on its file system.
2. Boot the system from the installation media.
3. Select the `Installation` menu item.
4. On the command line, set `autoupgrade=1`.
5. Press `Enter` to start the upgrade process.

4.11 Services and Targets

With the `services-manager` resource you can set the default systemd target and specify in detail which system services you want to start or deactivate.

The `default-target` property specifies the default systemd target into which the system boots. Valid options are `graphical` for a graphical login, or `multi-user` for a console login.

The <enable config:type="list"> and <disable config:type="list"> let you explicitly enable or disable services.

EXAMPLE 4.25: CONFIGURING SERVICES AND TARGETS

```
<services-manager>
  <default_target>multi-user</default_target>
  <services>
    <disable config:type="list">
      <service>cups</service>
    </disable>
    <enable config:type="list">
      <service>sshd</service>
    </enable>
  </services>
</services-manager>
```

4.12 Network Configuration

Network configuration is used to connect a single workstation to an Ethernet-based LAN or to configure a dial-up connection. More complex configurations (multiple network cards, routing, etc.) are also provided.

If the following setting is set to `true`, YaST will keep network settings created during the installation (via Linuxrc) and/or merge it with network settings from the AutoYaST control file (if defined). AutoYaST settings have higher priority than already present configuration files. YaST will write ifcfg-* files based on the entries in the control file without removing old ones. If there is an empty or no DNS and routing section, YaST will keep already existing values. Otherwise settings from the control file will be applied.

```
<keep_install_network
config:type="boolean">true</keep_install_network>
```

During the second stage, installation of additional packages will take place before the network, as described in the profile, is configured. `keep_install_network` is set by default to `true` to ensure that a network is available in case it is needed to install those packages. If all packages are installed during the first stage and the network is not needed early during the second one, setting `keep_install_network` to `false` will avoid copying the configuration.

To configure network settings and activate networking automatically, one global resource is used to store the whole network configuration.

EXAMPLE 4.26: NETWORK CONFIGURATION

```
<networking>
  <dns>
    <dhcp_hostname config:type="boolean">true</dhcp_hostname>
    <domain>site</domain>
    <hostname>linux-bqua</hostname>
    <nameservers config:type="list">
      <nameserver>192.168.1.116</nameserver>
      <nameserver>192.168.1.117</nameserver>
      <nameserver>192.168.1.118</nameserver>
    </nameservers>
    <resolv_conf_policy>auto</resolv_conf_policy>
    <searchlist config:type="list">
      <search>example.com</search>
      <search>example.net</search>
    </searchlist>
    <write_hostname config:type="boolean">false</write_hostname>
  </dns>
```

```
  <interfaces config:type="list">
    <interface>
      <bootproto>dhcp</bootproto>
      <device>eth0</device>
      <startmode>auto</startmode>
    </interface>
    <interface>
      <bootproto>static</bootproto>
      <broadcast>127.255.255.255</broadcast>
      <device>lo</device>
      <firewall>no</firewall>
      <ipaddr>127.0.0.1</ipaddr>
      <netmask>255.0.0.0</netmask>
      <network>127.0.0.0</network>
      <prefixlen>8</prefixlen>
      <startmode>nfsroot</startmode>
      <usercontrol>no</usercontrol>
    </interface>
  </interfaces>
  <ipv6 config:type="boolean">true</ipv6>
  <keep_install_network config:type="boolean">false</keep_install_network>
  <managed config:type="boolean">false</managed>       ###### NetworkManager?
  <net-udev config:type="list">
    <rule>
      <name>eth0</name>
      <rule>ATTR{address}</rule>
      <value>00:30:6E:08:EC:80</value>
    </rule>
  </net-udev>
  <s390-devices config:type="list">
    <listentry>
      <chanids>0.0.0800 0.0.0801 0.0.0802</chanids>
      <type>qeth</type>
    </listentry>
  </s390-devices>
  <routing>
    <ipv4_forward config:type="boolean">false</ipv4_forward>
    <ipv6_forward config:type="boolean">false</ipv6_forward>
    <routes config:type="list">
      <route>
        <destination>192.168.2.1</destination>
        <device>eth0</device>
        <extrapara>foo</extrapara>
        <gateway>-</gateway>
        <netmask>-</netmask>
      </route>
      <route>
```

```
        <destination>default</destination>
        <device>eth0</device>
        <gateway>192.168.1.1</gateway>
        <netmask>-</netmask>
      </route>
      <route>
        <destination>default</destination>
        <device>lo</device>
        <gateway>192.168.5.1</gateway>
        <netmask>-</netmask>
      </route>
    </routes>
  </routing>
</networking>
```

EXAMPLE 4.27: BRIDGE INTERFACE CONFIGURATION

```
<interfaces config:type="list">
  <interface>
    <device>br0</device>
    <bootproto>static</bootproto>
    <bridge>yes</bridge>
    <bridge_forwarddelay>0</bridge_forwarddelay>
    <bridge_ports>eth0 eth1</bridge_ports>
    <bridge_stp>off</bridge_stp>
    <ipaddr>192.168.122.100</ipaddr>
    <netmask>255.255.255.0</netmask>
    <network>192.168.122.0</network>
    <prefixlen>24</prefixlen>
    <startmode>auto</startmode>
  </interface>
  <interface>
    <device>eth0</device>
    <bootproto>none</bootproto>
    <startmode>hotplug</startmode>
  </interface>
  <interface>
    <device>eth1</device>
    <bootproto>none</bootproto>
    <startmode>hotplug</startmode>
  </interface>
</interfaces>
```

Tip: IPv6 Address Support

Using IPv6 addresses in AutoYaST is fully supported. To disable IPv6 Address Support, set <ipv6 config:type="boolean">false</ipv6>

4.12.1 Persistent Names of Network Interfaces

The following elements must be between the <net-udev>...</net-udev> tags.

Element	Description	Comment
name	Network interface name, for example `eth3`	required
rule	`ATTR{address}` for a MAC based rule, `KERNELS` for a bus ID based rule	required
value	for example `f0:de:f1:6b:da:69` for a MAC rule, `0000:00:1c.1 or 0.0.0700` for a bus ID rule	required

4.12.2 s390 Options

The following elements must be between the <s390-devices>...</s390-devices> tags.

Element	Description	Comment
type	qeth, ctc or iucv	
chanids	channel ids separated by spaces `<chanids>0.0.0700 0.0.0701 0.0.0702</chanids>`	

Element	Description	Comment
layer2	`<layer2 config:type="boolean">true</layer2>`	boolean; default: false
portname	QETH port name (deprecated since SLE 12 SP2)	
protocol	CTC / LCS protocol, a small number (as a string) `<protocol>1</protocol>`	optional
router	IUCV router/user	

In addition to the options mentioned above, AutoYaST also supports IBM Z-specific options in other sections of the configuration file. In particular, you can define the logical link address, or LLADDR (in the case of Ethernet, that is the MAC address). To do so, use the option `LLADDR` in the device definition.

Tip: LLADDR for VLANs

VLAN devices inherit their LLADDR from the underlying physical devices. To set a particular address for a VLAN device, set the LLADDR option for the underlying physical device.

4.12.3 Proxy

Configure your Internet proxy (caching) settings.

Configure proxies for HTTP, HTTPS, and FTP with `http_proxy`, `https_proxy` and `ftp_proxy`, respectively. Addresses or names that should be directly accessible need to be specified with `no_proxy` (space separated values). If you are using a proxy server with authorization, fill in `proxy_user` and `proxy_password`,

EXAMPLE 4.28: NETWORK CONFIGURATION: PROXY

```
<proxy>
  <enabled config:type="boolean">true</enabled>
  <ftp_proxy>http://192.168.1.240:3128</ftp_proxy>
  <http_proxy>http://192.168.1.240:3128</http_proxy>
```

```
  <no_proxy>www.example.com .example.org localhost</no_proxy>
  <proxy_password>testpw</proxy_password>
  <proxy_user>testuser</proxy_user>
</proxy>
```

4.12.4 (X)Inetd

The control file has elements to specify which superserver should be used (netd_service), whether it should be enabled (netd_status) and how the services should be configured (netd_conf).

A service description element needs two parts: key and non-key. When writing the configuration, services are matched using the key fields; to the matching service, non-key fields are applied. If no service matches, it is created. If more services match, a warning is reported. The key fields are *script, service, protocol and server*.

service and *protocol* are matched literally. *script* is the base name of the configuration file: usually a file in `/etc/xinetd.d`, for example "echo-udp", or "inetd.conf". For compatibility with 8.2, *server* is matched more loosely: if it is `/usr/sbin/tcpd`, the real server name is taken from *server_args*. After that, the basename of the first whitespace-separated word is taken and these values are compared.

EXAMPLE 4.29: INETD EXAMPLE

```
<profile>
  <inetd>
    <netd_service config:type="symbol">xinetd</netd_service>
    <netd_status config:type="integer">0</netd_status>
    <netd_conf config:type="list">
      <conf>
        <script>imap</script>
        <service>pop3</service>
        <enabled config:type="boolean">true</enabled>
      </conf>
      <conf>
        <server>in.ftpd</server>
        <server_args>-A</server_args>
        <enabled config:type="boolean">true</enabled>
      </conf>
      <conf>
        <service>daytime</service>
        <protocol>tcp</protocol>
      </conf>
      <conf>...</conf>
    </netd_conf>
```

```
  </inetd>
</profile>
```

4.13 NIS Client

Using the `nis` resource, you can configure the target machine as a NIS client. The following example shows a detailed configuration using multiple domains.

EXAMPLE 4.30: NETWORK CONFIGURATION: NIS

```
 <nis>
  <nis_broadcast config:type="boolean">true</nis_broadcast>
  <nis_broken_server config:type="boolean">true</nis_broken_server>
  <nis_by_dhcp config:type="boolean">false</nis_by_dhcp>
  <nis_domain>test.com</nis_domain>
  <nis_local_only config:type="boolean">true</nis_local_only>
  <nis_options></nis_options>
  <nis_other_domains config:type="list">
    <nis_other_domain>
      <nis_broadcast config:type="boolean">false</nis_broadcast>
      <nis_domain>domain.com</nis_domain>
      <nis_servers config:type="list">
        <nis_server>10.10.0.1</nis_server>
      </nis_servers>
    </nis_other_domain>
  </nis_other_domains>
  <nis_servers config:type="list">
    <nis_server>192.168.1.1</nis_server>
  </nis_servers>
  <start_autofs config:type="boolean">true</start_autofs>
  <start_nis config:type="boolean">true</start_nis>
</nis>
```

4.14 NIS Server

You can configure the target machine as a NIS server. NIS Master Server and NIS Slave Server and a combination of both are available.

EXAMPLE 4.31: NIS SERVER CONFIGURATION

```
  <nis_server>
    <domain>mydomain.de</domain>
    <maps_to_serve config:type="list">
```

```
      <nis_map>auto.master</nis_map>
      <nis_map>ethers</nis_map>
    </maps_to_serve>
    <merge_passwd config:type="boolean">false</merge_passwd>
    <mingid config:type="integer">0</mingid>
    <minuid config:type="integer">0</minuid>
    <nopush config:type="boolean">false</nopush>
    <pwd_chfn config:type="boolean">false</pwd_chfn>
    <pwd_chsh config:type="boolean">false</pwd_chsh>
    <pwd_srcdir>/etc</pwd_srcdir>
    <securenets config:type="list">
      <securenet>
        <netmask>255.0.0.0</netmask>
        <network>127.0.0.0</network>
      </securenet>
    </securenets>
    <server_type>master</server_type>
    <slaves config:type="list"/>
    <start_ypbind config:type="boolean">false</start_ypbind>
    <start_yppasswdd config:type="boolean">false</start_yppasswdd>
    <start_ypxfrd config:type="boolean">false</start_ypxfrd>
  </nis_server>
```

Attribute	Values	Description
`domain`	NIS domain name.	
`maps_to_serve`	List of maps which are available for the server.	Values: auto.master, ethers, group, hosts, netgrp, networks, passwd, protocols, rpc, services, shadow
`merge_passwd`	Select if your passwd file should be merged with the shadow file (only possible if the shadow file exists).	Value: true/false
`mingid`	Minimum GID to include in the user maps.	
`minuid`	Minimum UID to include in the user maps.	

Attribute	Values	Description
`nopush`	Do not push the changes to slave servers. (Useful if there are none).	Value: true/false
`pwd_chfn`	YPPWD_CHFN - allow changing the full name	Value: true/false
`pwd_chsh`	YPPWD_CHSH - allow changing the login shell	Value: true/false
`pwd_srcdir`	YPPWD_SRCDIR - source directory for passwd data	Default: `/etc`
`securenets`	List of allowed hosts to query the NIS server	A host address will be allowed if network is equal to the bitwise AND of the host's address and the netmask. The entry with netmask 255.0.0.0 and network 127.0.0.0 must exist to allow connections from the local host. Entering netmask 0.0.0.0 and network 0.0.0.0 gives access to all hosts.
`server_type`	Select whether to configure the NIS server as a master or a slave or not to configure a NIS server.	Values: master, slave, none
`slaves`	List of host names to configure as NIS server slaves.	

Attribute	Values	Description
start_ypbind	This host is also a NIS client (only when client is configured locally).	Value: true/false
start_yppasswdd	Also start the password daemon.	Value: true/false
start_ypxfrd	Also start the map transfer daemon. Fast Map distribution; it will speed up the transfer of maps to the slaves.	Value: true/false

4.15 LDAP Server (Authentication Server)

Using the auth-server resource, you can configure the target machine as an LDAP server. The following example shows a detailed configuration.

EXAMPLE 4.32: LDAP CONFIGURATION

```
<auth-server>
  <daemon>
    <listeners config:type="list">
      <listentry>ldap</listentry>
      <listentry>ldapi</listentry>
    </listeners>
    <serviceEnabled>1</serviceEnabled>
    <slp/>
  </daemon>
  <databases config:type="list">
    <listentry>
      <access config:type="list">
        <listentry>
          <access config:type="list">
            <listentry>
              <control/>
              <level>write</level>
              <type>self</type>
              <value/>
            </listentry>
```

```
      <listentry>
        <control/>
        <level>auth</level>
        <type>*</type>
        <value/>
      </listentry>
    </access>
    <target>
      <attrs>userPassword</attrs>
    </target>
  </listentry>
  <listentry>
    <access config:type="list">
      <listentry>
        <control/>
        <level>write</level>
        <type>self</type>
        <value/>
      </listentry>
      <listentry>
        <control/>
        <level>read</level>
        <type>*</type>
        <value/>
      </listentry>
    </access>
    <target>
      <attrs>shadowLastChange</attrs>
    </target>
  </listentry>
  <listentry>
    <access config:type="list">
      <listentry>
        <control/>
        <level>read</level>
        <type>self</type>
        <value/>
      </listentry>
      <listentry>
        <control/>
        <level>none</level>
        <type>*</type>
        <value/>
      </listentry>
    </access>
    <target>
      <attrs>userPKCS12</attrs>
```

```
            </target>
          </listentry>
          <listentry>
            <access config:type="list">
              <listentry>
                <control/>
                <level>read</level>
                <type>*</type>
                <value/>
              </listentry>
            </access>
            <target/>
          </listentry>
        </access>
        <checkpoint config:type="list">
          <listentry>1024</listentry>
          <listentry>5</listentry>
        </checkpoint>
        <directory>/var/lib/ldap</directory>
        <entrycache>10000</entrycache>
        <idlcache>30000</idlcache>
        <indexes>
          <cn>
            <eq>1</eq>
            <sub>1</sub>
          </cn>
          <displayName>
            <eq>1</eq>
            <sub>1</sub>
          </displayName>
          <gidNumber>
            <eq>1</eq>
          </gidNumber>
          <givenName>
            <eq>1</eq>
            <sub>1</sub>
          </givenName>
          <mail>
            <eq>1</eq>
          </mail>
          <member>
            <eq>1</eq>
          </member>
          <memberUid>
            <eq>1</eq>
          </memberUid>
          <objectclass>
```

```
          <eq>1</eq>
        </objectclass>
        <sn>
          <eq>1</eq>
          <sub>1</sub>
        </sn>
        <uid>
          <eq>1</eq>
          <sub>1</sub>
        </uid>
        <uidNumber>
          <eq>1</eq>
        </uidNumber>
      </indexes>
      <rootdn>cn=Administrator,DC=corp,DC=Fabrikam,DC=COM,CN=Karen Berge</rootdn>
      <rootpw>{SSHA}LCdgE3gNejqBogGI3ac1Xf4DOIVMSk9ZQg==</rootpw>
      <suffix>DC=corp,DC=Fabrikam,DC=COM,CN=Karen Berge</suffix>
      <type>hdb</type>
    </listentry>
  </databases>
  <globals>
    <allow config:type="list"/>
    <disallow config:type="list"/>
    <loglevel config:type="list">
      <listentry>none</listentry>
    </loglevel>
    <tlsconfig>
      <caCertDir/>
      <caCertFile/>
      <certFile/>
      <certKeyFile/>
      <crlCheck>0</crlCheck>
      <crlFile/>
      <verifyClient>0</verifyClient>
    </tlsconfig>
  </globals>
  <schema config:type="list">
    <listentry>
      <definition>dn: cn=schema,cn=config
          objectClass: olcSchemaConfig
          ......
          .....
          ....
          ...
          ..
          .
      </definition>
```

```
      <name>schema</name>
    </listentry>
    <listentry>
      <includeldif>/etc/openldap/schema/core.ldif</includeldif>
    </listentry>
    <listentry>
      <includeldif>/etc/openldap/schema/cosine.ldif</includeldif>
    </listentry>
    <listentry>
      <includeldif>/etc/openldap/schema/inetorgperson.ldif</includeldif>
    </listentry>
    <listentry>
      <includeschema>/etc/openldap/schema/rfc2307bis.schema</includeschema>
    </listentry>
    <listentry>
      <includeschema>/etc/openldap/schema/yast.schema</includeschema>
    </listentry>
  </schema>
</auth-server>
```

4.16 Windows Domain Membership

Using the `samba-client` resource, you can configure a membership of a workgroup, NT domain, or Active Directory domain.

EXAMPLE 4.33: SAMBA CLIENT CONFIGURATION

```
<samba-client>
  <disable_dhcp_hostname config:type="boolean">true</disable_dhcp_hostname>
  <global>
    <security>domain</security>
    <usershare_allow_guests>No</usershare_allow_guests>
    <usershare_max_shares>100</usershare_max_shares>
    <workgroup>WORKGROUP</workgroup>
  </global>
  <winbind config:type="boolean">false</winbind>
</samba-client>
```

Attribute	Values	Description
`disable_dhcp_hostname`	Do not allow DHCP to change the host name.	Value: true/false

Attribute	Values	Description
`global/security`	Kind of authentication regime (domain technology or Active Directory server (ADS)).	Value: ADS/domain
`global/usershare_allow_guests`	Sharing guest access is allowed.	Value: No/Yes
`global/usershare_max_shares`	Max. number of shares from `smb.conf`.	0 means that shares are not enabled.
`global/workgroup`	Workgroup or domain name.	
`winbind`	Using winbind.	Value: true/false

4.17 Samba Server

Configuration of a simple Samba server.

EXAMPLE 4.34: SAMBA SERVER CONFIGURATION

```
<samba-server>
  <accounts config:type="list"/>
  <backend/>
  <config config:type="list">
    <listentry>
      <name>global</name>
      <parameters>
        <security>domain</security>
        <usershare_allow_guests>No</usershare_allow_guests>
        <usershare_max_shares>100</usershare_max_shares>
        <workgroup>WORKGROUP</workgroup>
      </parameters>
    </listentry>
  </config>
  <service>Disabled</service>
  <trustdom/>
  <version>2.11</version>
</samba-server>
```

Attribute	Values	Description
`accounts`	List of Samba accounts.	
`backend`	List of available back-ends	Value: true/false
`config`	Setting additional user defined parameters in `/etc/samba/smb.conf`.	The example shows parameters in the `global` section of `/etc/samba/smb.conf`.
`service`	Samba service starts during boot.	Value: Enabled/Disabled
`trustdom/`	Trusted Domains.	A map of two maps (keys: `establish, revoke`). Each map contains entries in the format key: domainname value: password.
`version`	Samba version.	Default: 2.11

4.18 Authentication Client

The following is a simple example for an LDAP user authentication. NSS and PAM will automatically be configured accordingly. Required data are the name of the search base (base DN, for example, `dc=mydomain,dc=com`) and the IP address of the LDAP server.

EXAMPLE 4.35: NETWORK CONFIGURATION: AUTHENTICATION CLIENT

```
<auth-client>
  <sssd>yes</sssd>
  <nssldap>no</nssldap>
  <sssd_conf>
    <sssd>
      <config_file_version>2</config_file_version>
      <services>nss, pam, sudo</services>
      <domains>EXAMPLE</domains>
    </sssd>
    <auth_domains>
      <domain>
        <domain_name>EXAMPLE</domain_name>
```

```
        <id_provider>ldap</id_provider>
        <sudo_provider>ldap</sudo_provider>
        <ldap_uri>ldap://example.com</ldap_uri>
        <ldap_sudo_search_base>ou=sudoers,dc=example,dc=com</ldap_sudo_search_base>
      </domain>
    </auth_domains>
  </sssd_conf>
</auth-client>
```

Tip: Using ldaps://

To use LDAP with native SSL (rather than TLS), add the `ldaps` resource:

```
<auth-client>
  <sssd_conf>
    <auth_domains>
      <domain>
        <ldaps config:type="boolean">true</ldaps>
      </domain>
    </auth_domains>
  </sssd_conf>
</auth-client>
```

4.19 NFS Client and Server

Configuring a system as an NFS client or an NFS server is can be done using the configuration system. The following examples show how both NFS client and server can be configured.

From SUSE Linux Enterprise Server 12 SP5 on, the structure of NFS client configuration has changed. Some global configuration options were introduced: `enable_nfs4` to switch NFS4 support on/off and `idmapd_domain` to define domain name for rpc.idmapd (this only makes sense when NFS4 is enabled). Attention: the old structure is not compatible with the new one and the control files with an NFS section created on older releases will not work with newer products.

EXAMPLE 4.36: NETWORK CONFIGURATION: NFS CLIENT

```
<nfs>
  <enable_nfs4 config:type="boolean">true</enable_nfs4>
  <idmapd_domain>suse.cz</idmapd_domain>
  <nfs_entries config:type="list">
```

```
    <nfs_entry>
      <mount_point>/home</mount_point>
      <nfs_options>sec=krb5i,intr,rw</nfs_options>
      <server_path>saurus.suse.cz:/home</server_path>
      <vfstype>nfs4</vfstype>
    </nfs_entry>
    <nfs_entry>
      <mount_point>/work</mount_point>
      <nfs_options>defaults</nfs_options>
      <server_path>bivoj.suse.cz:/work</server_path>
      <vfstype>nfs</vfstype>
    </nfs_entry>
    <nfs_entry>
      <mount_point>/mnt</mount_point>
      <nfs_options>defaults</nfs_options>
      <server_path>fallback.suse.cz:/srv/dist</server_path>
      <vfstype>nfs</vfstype>
    </nfs_entry>
  </nfs_entries>
</nfs>
```

EXAMPLE 4.37: NETWORK CONFIGURATION: NFS SERVER

```
<nfs_server>
  <nfs_exports config:type="list">
    <nfs_export>
      <allowed config:type="list">
        <allowed_clients>*(ro,root_squash,sync)</allowed_clients>
      </allowed>
      <mountpoint>/home</mountpoint>
    </nfs_export>
    <nfs_export>
      <allowed config:type="list">
        <allowed_clients>*(ro,root_squash,sync)</allowed_clients>
      </allowed>
      <mountpoint>/work</mountpoint>
    </nfs_export>
  </nfs_exports>
  <start_nfsserver config:type="boolean">true</start_nfsserver>
</nfs_server>
```

4.20 NTP Client

Select whether to start the NTP daemon when booting the system. The NTP daemon resolves host names when initializing.

To run NTP daemon in chroot jail, set `start_in_chroot`. Starting any daemon in a chroot jail is more secure and strongly recommended. To adjust NTP servers, peers, local clocks, and NTP broadcasting, add the appropriate entry to the control file. An example of various configuration options is shown below.

EXAMPLE 4.38: NETWORK CONFIGURATION: NTP CLIENT

```
<ntp-client>
  <configure_dhcp config:type="boolean">false</configure_dhcp>
  <peers config:type="list">
    <peer>
      <address>ntp.example.com</address>
      <options></options>
      <type>server</type>
    </peer>
  </peers>
  <start_at_boot config:type="boolean">true</start_at_boot>
  <start_in_chroot config:type="boolean">true</start_in_chroot>
</ntp-client>
```

4.21 Mail Configuration

For the mail configuration of the client, this module lets you create a detailed mail configuration. The module contains various options. We recommended you use it at least for the initial configuration.

EXAMPLE 4.39: MAIL CONFIGURATION

```
<mail>
  <aliases config:type="list">
    <alias>
      <alias>root</alias>
      <comment></comment>
      <destinations>foo</destinations>
    </alias>
    <alias>
      <alias>test</alias>
      <comment></comment>
      <destinations>foo</destinations>
    </alias>
  </aliases>
  <connection_type config:type="symbol">permanent</connection_type>
  <fetchmail config:type="list">
    <fetchmail_entry>
```

```
      <local_user>foo</local_user>
      <password>bar</password>
      <protocol>POP3</protocol>
      <remote_user>foo</remote_user>
      <server>pop.foo.com</server>
    </fetchmail_entry>
    <fetchmail_entry>
      <local_user>test</local_user>
      <password>bar</password>
      <protocol>IMAP</protocol>
      <remote_user>test</remote_user>
      <server>blah.com</server>
    </fetchmail_entry>
  </fetchmail>
  <from_header>test.com</from_header>
  <listen_remote config:type="boolean">true</listen_remote>
  <local_domains config:type="list">
    <domains>test1.com</domains>
  </local_domains>
  <masquerade_other_domains config:type="list">
      <domain>blah.com</domain>
  </masquerade_other_domains>
  <masquerade_users config:type="list">
    <masquerade_user>
      <address>joe@test.com</address>
      <comment></comment>
      <user>joeuser</user>
    </masquerade_user>
    <masquerade_user>
      <address>bar@test.com</address>
      <comment></comment>
      <user>foo</user>
    </masquerade_user>
  </masquerade_users>
  <mta config:type="symbol">postfix</mta>
  <outgoing_mail_server>test.com</outgoing_mail_server>
  <postfix_mda config:type="symbol">local</postfix_mda>
  <smtp_auth config:type="list">
    <listentry>
      <password>bar</password>
      <server>test.com</server>
      <user>foo</user>
    </listentry>
  </smtp_auth>
  <use_amavis config:type="boolean">true</use_amavis>
  <virtual_users config:type="list">
    <virtual_user>
```

```
      <alias>test.com</alias>
      <comment></comment>
      <destinations>foo.com</destinations>
    </virtual_user>
    <virtual_user>
      <alias>geek.com</alias>
      <comment></comment>
      <destinations>bar.com</destinations>
    </virtual_user>
  </virtual_users>
</mail>
```

4.22 HTTP Server Configuration

This section is used for configuration of an Apache HTTP server.

For less experienced users, we would suggest to configure the Apache server using the `HTTP server` YaST module. After that, call the `AutoYaST configuration` module, select the `HTTP server` YaST module and clone the Apache settings. These settings can be exported via the menu `File`.

EXAMPLE 4.40: HTTP SERVER CONFIGURATION

```
  <http-server>
    <Listen config:type="list">
      <listentry>
        <ADDRESS/>
        <PORT>80</PORT>
      </listentry>
    </Listen>
    <hosts config:type="list">
      <hosts_entry>
        <KEY>main</KEY>
        <VALUE config:type="list">
          <listentry>
            <KEY>DocumentRoot</KEY>
            <OVERHEAD>
            #
            # Global configuration that will be applicable for all
            # virtual hosts, unless deleted here or overriden elsewhere.
            #
            </OVERHEAD>
            <VALUE>"/srv/www/htdocs"</VALUE>
          </listentry>
          <listentry>
```

```
            <KEY>_SECTION</KEY>
            <OVERHEAD>
            #
            # Configure the DocumentRoot
            #
            </OVERHEAD>
            <SECTIONNAME>Directory</SECTIONNAME>
            <SECTIONPARAM>"/srv/www/htdocs"</SECTIONPARAM>
            <VALUE config:type="list">
              <listentry>
                <KEY>Options</KEY>
                <OVERHEAD>
                # Possible values for the Options directive are "None", "All",
                # or any combination of:
                #   Indexes Includes FollowSymLinks SymLinksifOwnerMatch
                #   ExecCGI MultiViews
                #
                # Note that "MultiViews" must be named *explicitly*
                # --- "Options All"
                # does not give it to you.
                #
                # The Options directive is both complicated and important.
                #  Please see
                #  http://httpd.apache.org/docs/2.4/mod/core.html#options
                # for more information.
                </OVERHEAD>
                <VALUE>None</VALUE>
              </listentry>
              <listentry>
                <KEY>AllowOverride</KEY>
                <OVERHEAD>
                # AllowOverride controls what directives may be placed in
                # .htaccess files. It can be "All", "None", or any combination
                # of the keywords:
                #   Options FileInfo AuthConfig Limit
                </OVERHEAD>
                <VALUE>None</VALUE>
              </listentry>
              <listentry>
                <KEY>_SECTION</KEY>
                <OVERHEAD>
                # Controls who can get stuff from this server.
                </OVERHEAD>
                <SECTIONNAME>IfModule</SECTIONNAME>
                <SECTIONPARAM>!mod_access_compat.c</SECTIONPARAM>
                <VALUE config:type="list">
                  <listentry>
```

```
                <KEY>Require</KEY>
                <VALUE>all granted</VALUE>
              </listentry>
            </VALUE>
          </listentry>
          <listentry>
            <KEY>_SECTION</KEY>
            <SECTIONNAME>IfModule</SECTIONNAME>
            <SECTIONPARAM>mod_access_compat.c</SECTIONPARAM>
            <VALUE config:type="list">
              <listentry>
                <KEY>Order</KEY>
                <VALUE>allow,deny</VALUE>
              </listentry>
              <listentry>
                <KEY>Allow</KEY>
                <VALUE>from all</VALUE>
              </listentry>
            </VALUE>
          </listentry>
        </VALUE>
      </listentry>
      <listentry>
        <KEY>Alias</KEY>
        <OVERHEAD>
        # Aliases: aliases can be added as needed (with no limit).
        # The format is Alias fakename realname
        #
        # Note that if you include a trailing / on fakename then the
        # server will require it to be present in the URL.  So "/icons"
        # is not aliased in this example, only "/icons/".  If the fakename
        # is slash-terminated, then the realname must also be slash
        # terminated, and if the fakename omits the trailing slash, the
        # realname must also omit it.
        # We include the /icons/ alias for FancyIndexed directory listings.
        # If you do not use FancyIndexing, you may comment this out.
        #
        </OVERHEAD>
        <VALUE>/icons/ "/usr/share/apache2/icons/"</VALUE>
      </listentry>
      <listentry>
        <KEY>_SECTION</KEY>
        <OVERHEAD>
        </OVERHEAD>
        <SECTIONNAME>Directory</SECTIONNAME>
        <SECTIONPARAM>"/usr/share/apache2/icons"</SECTIONPARAM>
        <VALUE config:type="list">
```

```
            <listentry>
              <KEY>Options</KEY>
              <VALUE>Indexes MultiViews</VALUE>
            </listentry>
            <listentry>
              <KEY>AllowOverride</KEY>
              <VALUE>None</VALUE>
            </listentry>
            <listentry>
              <KEY>_SECTION</KEY>
              <SECTIONNAME>IfModule</SECTIONNAME>
              <SECTIONPARAM>!mod_access_compat.c</SECTIONPARAM>
              <VALUE config:type="list">
                <listentry>
                  <KEY>Require</KEY>
                  <VALUE>all granted</VALUE>
                </listentry>
              </VALUE>
            </listentry>
            <listentry>
              <KEY>_SECTION</KEY>
              <SECTIONNAME>IfModule</SECTIONNAME>
              <SECTIONPARAM>mod_access_compat.c</SECTIONPARAM>
              <VALUE config:type="list">
                <listentry>
                  <KEY>Order</KEY>
                  <VALUE>allow,deny</VALUE>
                </listentry>
                <listentry>
                  <KEY>Allow</KEY>
                  <VALUE>from all</VALUE>
                </listentry>
              </VALUE>
            </listentry>
          </VALUE>
        </listentry>
        <listentry>
          <KEY>ScriptAlias</KEY>
          <OVERHEAD>
          # ScriptAlias: This controls which directories contain server
          # scripts. ScriptAliases are essentially the same as Aliases,
          # except that documents in the realname directory are treated
          # as applications and run by the server when requested rather
          # than as documents sent to the client.
          # The same rules about trailing "/" apply to ScriptAlias
          # directives as to Alias.
          #
```

```
      </OVERHEAD>
      <VALUE>/cgi-bin/ "/srv/www/cgi-bin/"</VALUE>
    </listentry>
    <listentry>
      <KEY>_SECTION</KEY>
      <OVERHEAD>
      # "/srv/www/cgi-bin" should be changed to wherever your
      # ScriptAliased CGI directory exists, if you have that configured.
      #
      </OVERHEAD>
      <SECTIONNAME>Directory</SECTIONNAME>
      <SECTIONPARAM>"/srv/www/cgi-bin"</SECTIONPARAM>
      <VALUE config:type="list">
        <listentry>
          <KEY>AllowOverride</KEY>
          <VALUE>None</VALUE>
        </listentry>
        <listentry>
          <KEY>Options</KEY>
          <VALUE>+ExecCGI -Includes</VALUE>
        </listentry>
        <listentry>
          <KEY>_SECTION</KEY>
          <SECTIONNAME>IfModule</SECTIONNAME>
          <SECTIONPARAM>!mod_access_compat.c</SECTIONPARAM>
          <VALUE config:type="list">
            <listentry>
              <KEY>Require</KEY>
              <VALUE>all granted</VALUE>
            </listentry>
          </VALUE>
        </listentry>
        <listentry>
          <KEY>_SECTION</KEY>
          <SECTIONNAME>IfModule</SECTIONNAME>
          <SECTIONPARAM>mod_access_compat.c</SECTIONPARAM>
          <VALUE config:type="list">
            <listentry>
              <KEY>Order</KEY>
              <VALUE>allow,deny</VALUE>
            </listentry>
            <listentry>
              <KEY>Allow</KEY>
              <VALUE>from all</VALUE>
            </listentry>
          </VALUE>
        </listentry>
```

```
      </VALUE>
    </listentry>
    <listentry>
      <KEY>_SECTION</KEY>
      <OVERHEAD>
      # UserDir: The name of the directory that is appended onto a
      # user's home directory if a ~user request is received.
      # To disable it, simply remove userdir from the list of modules
      # in APACHE_MODULES in /etc/sysconfig/apache2.
      #
      </OVERHEAD>
      <SECTIONNAME>IfModule</SECTIONNAME>
      <SECTIONPARAM>mod_userdir.c</SECTIONPARAM>
      <VALUE config:type="list">
        <listentry>
          <KEY>UserDir</KEY>
          <OVERHEAD>
          # Note that the name of the user directory ("public_html")
          # cannot simply be changed here, since it is a compile time
          # setting. The apache package would have to be rebuilt.
          # You could work around by deleting /usr/sbin/suexec, but
          # then all scripts from the directories would be executed
          # with the UID of the webserver.
          </OVERHEAD>
          <VALUE>public_html</VALUE>
        </listentry>
        <listentry>
          <KEY>Include</KEY>
          <OVERHEAD>
          # The actual configuration of the directory is in
          # /etc/apache2/mod_userdir.conf.
          </OVERHEAD>
          <VALUE>/etc/apache2/mod_userdir.conf</VALUE>
        </listentry>
      </VALUE>
    </listentry>
    <listentry>
      <KEY>IncludeOptional</KEY>
      <OVERHEAD>
      # Include all *.conf files from /etc/apache2/conf.d/.
      #
      # This is mostly meant as a place for other RPM packages to drop
      # in their configuration snippet.
      #
      #
      # You can comment this out here if you want those bits include
      # only in a certain virtual host, but not here.
```

```
            </OVERHEAD>
            <VALUE>/etc/apache2/conf.d/*.conf</VALUE>
          </listentry>
          <listentry>
            <KEY>IncludeOptional</KEY>
            <OVERHEAD>
            # The manual... if it is installed ('?' means it will not complain)
            </OVERHEAD>
            <VALUE>/etc/apache2/conf.d/apache2-manual?conf</VALUE>
          </listentry>
          <listentry>
            <KEY>ServerName</KEY>
            <VALUE>linux-wtyj</VALUE>
          </listentry>
          <listentry>
            <KEY>ServerAdmin</KEY>
            <OVERHEAD>
            </OVERHEAD>
            <VALUE>root@linux-wtyj</VALUE>
          </listentry>
          <listentry>
            <KEY>NameVirtualHost</KEY>
            <VALUE>192.168.43.2</VALUE>
          </listentry>
        </VALUE>
      </hosts_entry>
      <hosts_entry>
        <KEY>192.168.43.2/secondserver.suse.de</KEY>
        <VALUE config:type="list">
          <listentry>
            <KEY>DocumentRoot</KEY>
            <VALUE>/srv/www/htdocs</VALUE>
          </listentry>
          <listentry>
            <KEY>ServerName</KEY>
            <VALUE>secondserver.suse.de</VALUE>
          </listentry>
          <listentry>
            <KEY>ServerAdmin</KEY>
            <VALUE>second_server@suse.de</VALUE>
          </listentry>
          <listentry>
            <KEY>_SECTION</KEY>
            <SECTIONNAME>Directory</SECTIONNAME>
            <SECTIONPARAM>/srv/www/htdocs</SECTIONPARAM>
            <VALUE config:type="list">
              <listentry>
```

```
              <KEY>AllowOverride</KEY>
              <VALUE>None</VALUE>
            </listentry>
            <listentry>
              <KEY>Require</KEY>
              <VALUE>all granted</VALUE>
            </listentry>
          </VALUE>
        </listentry>
      </VALUE>
    </hosts_entry>
  </hosts>
  <modules config:type="list">
    <module_entry>
      <change>enable</change>
      <name>socache_shmcb</name>
      <userdefined config:type="boolean">true</userdefined>
    </module_entry>
    <module_entry>
      <change>enable</change>
      <name>reqtimeout</name>
      <userdefined config:type="boolean">true</userdefined>
    </module_entry>
    <module_entry>
      <change>enable</change>
      <name>authn_core</name>
      <userdefined config:type="boolean">true</userdefined>
    </module_entry>
    <module_entry>
      <change>enable</change>
      <name>authz_core</name>
      <userdefined config:type="boolean">true</userdefined>
    </module_entry>
  </modules>
  <service config:type="boolean">true</service>
  <version>2.9</version>
</http-server>
```

List Name	List Elements	Description
Listen		List of host `Listen` settings
	PORT	port address

List Name	List Elements	Description
	ADDRESS	Network address. All addresses will be taken if this entry is empty.
hosts		List of Hosts configuration
	KEY	Host name; `<KEY>main</KEY>` defines the main hosts, for example `<KEY>192.168.43.2/secondserver.suse.de</KEY>`
	VALUE	List of different values describing the host.
modules		Module list. Only user defined modules need to be described.
	name	Module name
	userdefined	For historical reasons, it is always set to `true`.
	change	For historical reasons, it is always set to `enable`.

Element	Description	Comment
version	Version of used Apache server	Only for information. Default 2.9
service	Enable Apache service	Optional. Default: false

Note: Firewall

To run an Apache server correctly, make sure the firewall is configured appropriately.

4.23 Squid Server

Squid is a caching and forwarding Web proxy.

EXAMPLE 4.41: SQUID SERVER CONFIGURATION

```
<squid>
  <acls config:type="list">
    <listentry>
      <name>QUERY</name>
      <options config:type="list">
        <option>cgi-bin \?</option>
      </options>
      <type>urlpath_regex</type>
    </listentry>
    <listentry>
      <name>apache</name>
      <options config:type="list">
        <option>Server</option>
        <option>^Apache</option>
      </options>
      <type>rep_header</type>
    </listentry>
    <listentry>
      <name>all</name>
      <options config:type="list">
        <option>0.0.0.0/0.0.0.0</option>
      </options>
      <type>src</type>
    </listentry>
    <listentry>
      <name>manager</name>
      <options config:type="list">
        <option>cache_object</option>
      </options>
      <type>proto</type>
    </listentry>
    <listentry>
      <name>localhost</name>
      <options config:type="list">
        <option>127.0.0.1/255.255.255.255</option>
      </options>
      <type>src</type>
    </listentry>
    <listentry>
      <name>to_localhost</name>
      <options config:type="list">
```

```
          <option>127.0.0.0/8</option>
        </options>
        <type>dst</type>
      </listentry>
      <listentry>
        <name>SSL_ports</name>
        <options config:type="list">
          <option>443</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>80</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>21</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>443</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>70</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>210</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
```

```
        <name>Safe_ports</name>
        <options config:type="list">
          <option>1025-65535</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>280</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>488</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>591</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>777</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>CONNECT</name>
        <options config:type="list">
          <option>CONNECT</option>
        </options>
        <type>method</type>
      </listentry>
    </acls>
    <http_accesses config:type="list">
      <listentry>
        <acl config:type="list">
          <listentry>manager</listentry>
          <listentry>localhost</listentry>
```

```
        </acl>
        <allow config:type="boolean">true</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>manager</listentry>
        </acl>
        <allow config:type="boolean">false</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>!Safe_ports</listentry>
        </acl>
        <allow config:type="boolean">false</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>CONNECT</listentry>
          <listentry>!SSL_ports</listentry>
        </acl>
        <allow config:type="boolean">false</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>localhost</listentry>
        </acl>
        <allow config:type="boolean">true</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>all</listentry>
        </acl>
        <allow config:type="boolean">false</allow>
      </listentry>
    </http_accesses>
    <http_ports config:type="list">
      <listentry>
        <host/>
        <port>3128</port>
        <transparent config:type="boolean">false</transparent>
      </listentry>
    </http_ports>
    <refresh_patterns config:type="list">
      <listentry>
        <case_sensitive config:type="boolean">true</case_sensitive>
        <max>10080</max>
        <min>1440</min>
```

```
      <percent>20</percent>
      <regexp>^ftp:</regexp>
    </listentry>
    <listentry>
      <case_sensitive config:type="boolean">true</case_sensitive>
      <max>1440</max>
      <min>1440</min>
      <percent>0</percent>
      <regexp>^gopher:</regexp>
    </listentry>
    <listentry>
      <case_sensitive config:type="boolean">true</case_sensitive>
      <max>4320</max>
      <min>0</min>
      <percent>20</percent>
      <regexp>.</regexp>
    </listentry>
  </refresh_patterns>
  <service_enabled_on_startup config:type="boolean">true</service_enabled_on_startup>
  <settings>
    <access_log config:type="list">
      <listentry>/var/log/squid/access.log</listentry>
    </access_log>
    <cache_dir config:type="list">
      <listentry>ufs</listentry>
      <listentry>/var/cache/squid</listentry>
      <listentry>100</listentry>
      <listentry>16</listentry>
      <listentry>256</listentry>
    </cache_dir>
    <cache_log config:type="list">
      <listentry>/var/log/squid/cache.log</listentry>
    </cache_log>
    <cache_mem config:type="list">
      <listentry>8</listentry>
      <listentry>MB</listentry>
    </cache_mem>
    <cache_mgr config:type="list">
      <listentry>webmaster</listentry>
    </cache_mgr>
    <cache_replacement_policy config:type="list">
      <listentry>lru</listentry>
    </cache_replacement_policy>
    <cache_store_log config:type="list">
      <listentry>/var/log/squid/store.log</listentry>
    </cache_store_log>
    <cache_swap_high config:type="list">
```

```
        <listentry>95</listentry>
      </cache_swap_high>
      <cache_swap_low config:type="list">
        <listentry>90</listentry>
      </cache_swap_low>
      <client_lifetime config:type="list">
        <listentry>1</listentry>
        <listentry>days</listentry>
      </client_lifetime>
      <connect_timeout config:type="list">
        <listentry>2</listentry>
        <listentry>minutes</listentry>
      </connect_timeout>
      <emulate_httpd_log config:type="list">
        <listentry>off</listentry>
      </emulate_httpd_log>
      <error_directory config:type="list">
        <listentry/>
      </error_directory>
      <ftp_passive config:type="list">
        <listentry>on</listentry>
      </ftp_passive>
      <maximum_object_size config:type="list">
        <listentry>4096</listentry>
        <listentry>KB</listentry>
      </maximum_object_size>
      <memory_replacement_policy config:type="list">
        <listentry>lru</listentry>
      </memory_replacement_policy>
      <minimum_object_size config:type="list">
        <listentry>0</listentry>
        <listentry>KB</listentry>
      </minimum_object_size>
    </settings>
  </squid>
```

Attribute	Values	Description
`acls`	List of Access Control Settings (ACLs).	Each list entry contains the name, type, and additional options. Use the YaST Squid configuration module to get an overview of possible entries.

Attribute	Values	Description
`http_accesses`	In the Access Control table, access can be denied or allowed to ACL Groups.	If there are more ACL Groups in one definition, access will be allowed or denied to members who belong to all ACL Groups at the same time. The Access Control table is checked in the order listed here. The first matching entry is used.
`http_ports`	Define all ports where Squid will listen for clients' HTTP requests.	`Host` can contain a host name or IP address or remain empty. `transparent` disables PMTU discovery when transparent.
`refresh_patterns`	Refresh patterns define how Squid treats the objects in the cache.	The refresh patterns are checked in the order listed here. The first matching entry is used. `Min` determines how long (in minutes) an object should be considered fresh if no explicit expiry time is given. `Max` is the upper limit of how long objects without an explicit expiry time will be considered fresh. `Percent` is the percentage of the object's age (time since last modification). An object without an explicit expiry time will be considered fresh.

Attribute	Values	Description
`settings`	Map of all available general parameters with default values.	Use the YaST Squid configuration module to get an overview about possible entries.
`service_enabled_on_startup`	Squid service start when booting.	Value: true/false

4.24 FTP Server

Configure your FTP Internet server settings.

EXAMPLE 4.42: FTP SERVER CONFIGURATION:

```
<ftp-server>
  <AnonAuthen>2</AnonAuthen>
  <AnonCreatDirs>NO</AnonCreatDirs>
  <AnonMaxRate>0</AnonMaxRate>
  <AnonReadOnly>NO</AnonReadOnly>
  <AntiWarez>YES</AntiWarez>
  <Banner>Welcome message</Banner>
  <CertFile/>
  <ChrootEnable>NO</ChrootEnable>
  <EnableUpload>YES</EnableUpload>
  <FTPUser>ftp</FTPUser>
  <FtpDirAnon>/srv/ftp</FtpDirAnon>
  <FtpDirLocal/>
  <GuestUser/>
  <LocalMaxRate>0</LocalMaxRate>
  <MaxClientsNumber>10</MaxClientsNumber>
  <MaxClientsPerIP>3</MaxClientsPerIP>
  <MaxIdleTime>15</MaxIdleTime>
  <PasMaxPort>40500</PasMaxPort>
  <PasMinPort>40000</PasMinPort>
  <PassiveMode>YES</PassiveMode>
  <SSL>0</SSL>
  <SSLEnable>NO</SSLEnable>
  <SSLv2>NO</SSLv2>
  <SSLv3>NO</SSLv3>
  <StartDaemon>2</StartDaemon>
```

```
    <StartXinetd>YES</StartXinetd>
    <TLS>YES</TLS>
    <Umask/>
    <UmaskAnon/>
    <UmaskLocal/>
    <VerboseLogging>NO</VerboseLogging>
    <VirtualUser>NO</VirtualUser>
  </ftp-server>
```

Element	Description	Comment
AnonAuthen	Enable/disable anonymous and local users.	Authenticated Users Only: 1; Anonymous Only: 0; Both: 2
AnonCreatDirs	Anonymous users can create directories.	Values: YES/NO
AnonReadOnly	Anonymous users can upload.	Values: YES/NO
AnonMaxRate	The maximum data transfer rate permitted for anonymous clients.	KB/s
AntiWarez	Disallow downloading of files that were uploaded but not validated by a local admin.	Values: YES/NO
Banner	Specify the name of a file containing the text to display when someone connects to the server.	
CertFile	DSA certificate to use for SSL-encrypted connections	This option specifies the location of the DSA certificate to use for SSL-encrypted connections.

Element	Description	Comment
ChrootEnable	When enabled, local users will be (by default) placed in a chroot jail in their home directory after login.	Warning: This option has security implications. Values: YES/NO
EnableUpload	If enabled, FTP users can upload.	To allow anonymous users to upload, enable `AnonReadOnly`. Values: YES/NO
FTPUser	Defining anonymous FTP user.	
FtpDirAnon	FTP directory for anonymous users.	Specify a directory which is used for FTP anonymous users.
FtpDirLocal	FTP directory for authenticated users.	Specify a directory which is used for FTP authenticated users.
LocalMaxRate	The maximum data transfer rate permitted for local authenticated users.	KB/s
MaxClientsNumber	The maximum number of clients allowed to connect.	
MaxClientsPerIP	Max clients for one IP.	The maximum number of clients allowed to connect from the same source Internet address.
MaxIdleTime	The maximum time (timeout) a remote client may wait between FTP commands.	Minutes

Element	Description	Comment
PasMaxPort	Maximum value for a port range for passive connection replies.	`PassiveMode` needs to be set to YES.
PasMinPort	Minimum value for a port range for passive connection replies.	`PassiveMode` needs to be set to YES.
PassiveMode	Enable Passive Mode	Value: YES/NO
SSL	Security Settings	Disable SSL/TLS: 0; Accept SSL and TLS: 1; Refuse Connections Without SSL/TLS: 2
SSLEnable	If enabled, SSL connections are allowed.	Value: YES/NO
SSLv2	If enabled, SSL version 2 connections are allowed.	Value: YES/NO
SSLv3	If enabled, SSL version 3 connections are allowed.	Value: YES/NO
StartDaemon	FTP daemon is started.	Manually: 0; when booting: 1; via xinetd: 2
StartXinetd	Has to be set to YES if StartDaemon is 2.	Value: YES/NO
TLS	If enabled, TLS connections are allowed.	Value: YES/NO
Umask	File creation mask. (umask for files):(umask for directories).	For example `177:077` if you feel paranoid.

Element	Description	Comment
UmaskAnon	The value to which the umask for file creation is set for anonymous users.	To specify octal values, remember the "0" prefix, otherwise the value will be treated as a base 10 integer.
UmaskLocal	Umask for authenticated users.	To specify octal values, remember the "0" prefix, otherwise the value will be treated as a base 10 integer.
VerboseLogging	When enabled, all FTP requests and responses are logged.	Value: YES/NO
VirtualUser	By using virtual users, FTP accounts can be administrated without affecting system accounts.	Value: YES/NO

Note: Firewall

Proper Firewall setting will be required for the FTP server to run correctly.

4.25 TFTP Server

Configure your TFTP Internet server settings.

Use this to enable a server for TFTP (trivial file transfer protocol). The server will be started using xinetd.

Note that TFTP and FTP are not the same.

EXAMPLE 4.43: TFTP SERVER CONFIGURATION:

```
<tftp-server>
  <start_tftpd config:type="boolean">true</start_tftpd>
  <tftp_directory>/tftpboot</tftp_directory>
```

```
  </tftp-server>
```

Element	Description	Comment
start_tftpd	Enabling TFTP server service.	Value: true/false
tftp_directory	Boot Image Directory: Specify the directory where served files are located.	The usual value is /tftpboot. The directory will be created if it does not exist. The server uses this as its root directory (using the -s option).

4.26 Firstboot Workflow

The YaST firstboot utility (YaST Initial System Configuration), which runs after the installation is completed, lets you configure the before creation of the install image. On the first boot after configuration, users are then guided through a series of steps that allow for easier configuration of their desktops. YaST firstboot does not run by default and needs to be configured to run.

EXAMPLE 4.44: ENABLING FIRSTBOOT WORKFLOW

```
    <firstboot>
      <firstboot_enabled config:type="boolean">true</firstboot_enabled>
    </firstboot>
```

4.27 Security Settings

Using the features of this module, you can to change the local security settings on the target system. The local security settings include the boot configuration, login settings, password settings, user addition settings, and file permissions.

Configuring the security settings automatically is similar to the `Custom Settings` in the security module available in the running system. This allows you create a customized configuration.

EXAMPLE 4.45: SECURITY CONFIGURATION

> See the reference for the meaning and the possible values of the settings in the following example.

```
<security>
  <console_shutdown>ignore</console_shutdown>
  <displaymanager_remote_access>no</displaymanager_remote_access>
  <fail_delay>3</fail_delay>
  <faillog_enab>yes</faillog_enab>
  <gid_max>60000</gid_max>
  <gid_min>101</gid_min>
  <gdm_shutdown>root</gdm_shutdown>
  <lastlog_enab>yes</lastlog_enab>
  <encryption>md5</encryption>
  <obscure_checks_enab>no</obscure_checks_enab>
  <pass_max_days>99999</pass_max_days>
  <pass_max_len>8</pass_max_len>
  <pass_min_days>1</pass_min_days>
  <pass_min_len>6</pass_min_len>
  <pass_warn_age>14</pass_warn_age>
  <passwd_use_cracklib>yes</passwd_use_cracklib>
  <permission_security>secure</permission_security>
  <run_updatedb_as>nobody</run_updatedb_as>
  <uid_max>60000</uid_max>
  <uid_min>500</uid_min>
</security>
```

4.27.1 Password Settings Options

Change various password settings. These settings are mainly stored in the `/etc/login.defs` file.

Use this resource to activate one of the encryption methods currently supported. If not set, `DES` is configured.

`DES`, the Linux default method, works in all network environments, but it restricts you to passwords no longer than eight characters. `MD5` allows longer passwords, thus provides more security, but some network protocols do not support this, and you may have problems with NIS. `Blowfish` is also supported.

Additionally, you can set up the system to check for password plausibility and length etc.

4.27.2 Boot Settings

Use the security resource, to change various boot settings.

How to interpret Ctrl–Alt–Del?

When someone at the console has pressed the Ctrl–Alt–Del key combination, the system usually reboots. Sometimes it is desirable to ignore this event, for example, when the system serves as both workstation and server.

Shutdown behavior of GDM

Configure a list of users allowed to shut down the machine from GDM.

4.27.3 Login Settings

Change various login settings. These settings are mainly stored in the `/etc/login.defs` file.

4.27.4 New user settings (**useradd** settings)

Set the minimum and maximum possible user and group ID

4.28 Linux Audit Framework (LAF)

This module allows the configuration of the audit daemon and to add rules for the audit subsystem.

EXAMPLE 4.46: LAF CONFIGURATION

```
<audit-laf>
  <auditd>
    <flush>INCREMENTAL</flush>
    <freq>20</freq>
    <log_file>/var/log/audit/audit.log</log_file>
    <log_format>RAW</log_format>
    <max_log_file>5</max_log_file>
    <max_log_file_action>ROTATE</max_log_file_action>
    <name_format>NONE</name_format>
    <num_logs>4</num_logs>
  </auditd>
  <rules/>
</audit-laf>
```

Attribute	Values	Description
auditd/flush	Describes how to write the data to disk.	If set to INCREMENTAL the Frequency parameter tells how many records to write before issuing an explicit flush to disk. NONE means: no special effort is made to flush data, DATA: keep data portion synchronized, SYNC: keep data and metadata fully synchronized.
auditd/freq	This parameter tells how many records to write before issuing an explicit flush to disk.	The parameter flush needs to be set to INCREMENTAL.
auditd/log_file	The full path name to the log file.	
auditd/log_fomat	How much information needs to be logged.	Set RAW to log all data (store in a format exactly as the kernel sends it) or NOLOG to discard all audit information instead of writing it to disk (does not affect data sent to the dispatcher).
auditd/max_log_file	How much information needs to be logged.	Unit: Megabytes
auditd/num_logs	Number of log files.	max_log_file_action needs to be set to ROTATE
auditd/max_log_file_action	What happens if the log capacity has been reached.	If the action is set to ROTATE the Number of Log Files specifies the number of

Attribute	Values	Description
		files to keep. Set to `SYSLOG`, the audit daemon will write a warning to /var/log/messages. With `SUSPEND` the daemon stops writing records to disk. `IGNORE` means do nothing, `KEEP_LOGS` is similar to `ROTATE`, but log files are not overwritten.
`auditd/name_format`	Computer Name Format describes how to write the computer name to the log file.	If `USER` is set, the User Defined Name is used. `NONE` means no computer name is inserted. `HOSTNAME` uses the name returned by the 'gethostname' syscall. `FQD` uses the fully qualified domain name.
`rules`	Rules for auditctl	You can edit the rules manually, which we only recommend for advanced users. For more information about all options, see **`man auditctl`**.

4.29 Users and Groups

AutoYaST supports defining local users, groups, special login settings and even default options for new users. Those settings are defined in the following sections:

users

List of users

user_defaults

Default options for new users

groups

List of groups

login_settings

Special login settings like password-less login or autologin

Note: Users and groups set up during the first stage

Users and groups are set up during the first stage, so you can set up a usable system without running the second stage.

4.29.1 Users

A list of users can be defined in the `<users>` section. Take into account that at least the `root` users should be set up so you can log in after the installation is finished.

EXAMPLE 4.47: MINIMAL USER CONFIGURATION

```
<users config:type="list">
  <user>
    <username>root</username>
    <user_password>password</user_password>
    <encrypted config:type="boolean">false</encrypted>
  </user>
    <user>
    <username>tux</username>
    <user_password>password</user_password>
    <encrypted config:type="boolean">false</encrypted>
  </user>
</users>
```

The following example shows a more complex scenario. System-wide default settings from `/etc/default/useradd`, such as the shell or the parent directory for the home directory, are applied.

EXAMPLE 4.48: COMPLEX USER CONFIGURATION

```
<users config:type="list">
  <user>
    <username>root</username>
```

```
    <user_password>password</user_password>
    <uid>1001</uid>
    <gid>100</gid>
    <encrypted config:type="boolean">false</encrypted>
    <fullname>Root User</fullname>
    <authorized_keys config:type="list">
      <listentry>command="/opt/login.sh" ssh-rsa
 AAAAB3NzaC1yc2EAAAADAQABAAABAQDKLt1vnW2vTJpBp3VK91rFsBvpY97NljsVLdgUrlPbZ/
L51FerQQ+djQ/ivDASQjO+567nMGqfYGFA/De1EGMMEoeShza67qjNi14L1HBGgVojaNajMR/
NI2d1kDyvsgRy7D7FT5UGGUNT0dlcSD3b85zwgHeYLidgcGIoKeRi7HpVDOOTyhwUv4sq3ubrPCWARgPeOLdVFa9clC8PTZdxSeKp4j
PvMDa96DpxH1VlzJlAIHQsMkMHbsCazPNC0++Kp5ZVERiH root@example.net</listentry>
    </authorized_keys>
  </user>
  <user>
    <username>tux</username>
    <user_password>password</user_password>
    <uid>1002</uid>
    <gid>100</gid>
    <encrypted config:type="boolean">false</encrypted>
    <fullname>Plain User</fullname>
    <home>/Users/plain</home>
    <password_settings>
      <max>120</max>
      <inact>5</inact>
    </password_settings>
  </user>
</users>
```

Note: `authorized_keys` File Will Be Overwritten

If the profile defines a set of SSH authorized keys for a user in the `authorized_keys` section, an existing `$HOME/.ssh/authorized_keys` file will be overwritten. If not existing, the file will be created with the content specified. Avoid overwriting an existing authorized_keys by not specifying the respective section in the AutoYaST control file.

Note: Specifying a user ID (`uid`)

Each user on a Linux system has a numeric user ID. You can either specify such a user ID within the AutoYaST control file manually by using `uid`, or let the system automatically choose a user ID by not using `uid`.

User IDs should be unique throughout the system. If not, some applications such as the login manager `gdm` may no longer work as expected.

When adding users with the AutoYaST control file, it is strongly recommended not to mix user defined IDs and automatically provided IDs. When doing so, unique IDs cannot be guaranteed. Either specify IDs for all users added with the AutoYaST control file or let the system choose the ID for all users.

Attribute	Values	Description
`username`	Text `<username>lukesw</username>`	Required. It should be a valid user name. Check `man 8 useradd` if you are not sure.
`fullname`	Text `<fullname>Tux Torvalds</fullname>`	Optional. User's full name.
`forename`	Text `<forname>Tux</forename>`	Optional. User's forename.
`surname`	Text `<surname>Skywalker</surname>`	Optional. User's surname.
`uid`	Number `<uid>1001</uid>`	Optional. User ID. It should be a unique and must be a non-negative number. If not specified, AutoYaST will automatically choose a user ID. Also refer to *Note: Specifying a user ID (`uid`)* for additional information.

Attribute	Values	Description
`gid`	Number `<gid>100</gid>`	Optional. Initial group ID. It must be a unique and non-negative number. Moreover it must refer to an existing group.
`home`	Path `<home>/home/luke</home>`	Optional. Absolute path to the user's home directory. By default, `/home/username` will be used (for example, `alice`'s home directory will be `/home/alice`).
`shell`	Path `<shell>/usr/bin/zsh</shell>`	Optional. `/bin/bash` is the default value. If you choose another one, make sure that it's installed (adding the corresponding package to the `software` section).
`user_password`	Text `<user_password>some-password</user_password>`	Optional. If you enter an exclamation mark (`!`), a random password will be generated. A user's password can be written in plain text (not recommended) or in encrypted form. To create an encrypted password, use **`mkpasswd`**. Enter the password as written in `/etc/shadow` (second column). To enable or disable the use of encrypted passwords in the profile, see the `encrypted` parameter.

Attribute	Values	Description
`encrypted`	Boolean <encrypted config:type="boolean">true</ encrypted>	Optional. Considered `false` if not present. Indicates if the user's password in the profile is encrypted or not. AutoYaST supports standard encryption algorithms (see **`man 3 crypt)`**.
`password_settings`	Password settings <password_settings> <expire/> <max>60</max> <warn>7</warn> </password_settings>	Optional. Some password settings can be customized: `expire` (account expiration date in format `YYYY-MM-DD`), `flag` (`/etc/shadow` flag), `inact` (number of days after password expiration that account is disabled), `max` (maximum number of days a password is valid), `min` (grace period in days until which a user can change password after it has expired) and `warn` (number of days before expiration when the password change reminder starts).
`authorized_keys`	List of authorized keys <authorized_keys config:type="list"> <listentry>ssh-rsa ...</ listentry> </authorized_keys>	A list of authorized keys to be written to `$HOME/.ssh/authorized_keys`. See example below.

4.29.2 User Defaults

The profile can specify a set of default values for new users like password expiration, initial group, home directory prefix, etc. Besides using them as default values for the users that are defined in the profile, AutoYaST will write those settings to `/etc/default/useradd` to be read for `useradd`.

Attribute	Values	Description
`group`	Text `<group>100</group>`	Optional. Default initial login group.
`groups`	Text `<groups>users</groups>`	Optional. List of additional groups.
`home`	Path `<home>/home</home>`	Optional. User's home directory prefix.
`expire`	Date `<expire>2017-12-31</expire>`	Optional. Default password expiration date in `YYYY-MM-DD` format.
`inactive`	Number `<inactive>3</inactive>`	Optional. Number of days after which an expired account is disabled.
`no_groups`	Boolean `<no_groups config:type="boolean">true</no_groups>`	Optional. Do not use secondary groups.
`shell`	Path `<shell>/usr/bin/fish</shell>`	Default login shell. `/bin/bash` is the default value. If you choose another one, make sure that it is installed

Attribute	Values	Description
		(adding the corresponding package to the `software` section).
`skel`	Path `<skel>/etc/skel</skel>`	Optional. Location of the files to be used as skeleton directory when adding a new user. You can find more information in `man 8 useradd`.
`umask`	File creation mode mask `<umask>022</umask>`	Set the file creation mode mask for the home directory. By default `useradd` will use `022`. Check `man 8 useradd` and `man 1 umask` for further information.

4.29.3 Groups

A list of groups can be defined in `<groups>` as shown in the example.

EXAMPLE 4.49: GROUP CONFIGURATION

```
<groups config:type="list">
  <group>
    <gid>100<gid>
    <groupname>users</groupname>
    <userlist>bob,alice</userlist>
  </group>
</groups>
```

Attribute	Values	Description
`groupname`	Text `<groupname>users</groupname>`	Required. It should be a valid group name. Check `man 8 groupadd` if you are not sure.
`gid`	Number `<gid>100</gid>`	Optional. Group ID. It must be a unique and non-negative number.
`group_password`	Text `<group_password>password</group_password>`	Optional. The group's password can be written in plain text (not recommended) or in encrypted form. Check the `encrypted` to select the desired behavior.
`encrypted`	Boolean `<encrypted config:type="boolean">true</encrypted>`	Optional. Indicates if the group's password in the profile is encrypted or not.
`userlist`	Users list `<userlist>bob,alice</userlist>`	Optional. A list of users who belong to the group. User names must be separated by commas.

4.29.4 Login Settings

Two special login settings can be enabled through an AutoYaST profile: autologin and password-less login. Both of them are disabled by default.

EXAMPLE 4.50: ENABLING AUTOLOGIN AND PASSWORD-LESS LOGIN

```
<login_settings>
  <autologin_user>vagrant</autologin_user>
  <password_less_login config:type="boolean">true</password_less_login>
```

```
</login_settings>
```

Attribute	Values	Description
password_less_login	Boolean <password_less_login config:type="boolean">true</password_less_login>	Optional. Enables password-less login. It only affects graphical login.
autologin_user	Text <autologin_user>alice</autologin_user>	Optional. Enables autologin for the given user.

4.30 Custom User Scripts

By adding scripts to the auto-installation process you can customize the installation according to your needs and take control in different stages of the installation.

In the auto-installation process, five types of scripts can be executed at different points in time during the installation:

All scripts need to be in the <scripts> section.

- pre-scripts (very early, before anything else really happens)
- postpartitioning-scripts (after partitioning and mounting to /mnt but before RPM installation)
- chroot-scripts (after the package installation, before the first boot)
- post-scripts (during the first boot of the installed system, no services running)
- init-scripts (during the first boot of the installed system, all services up and running)

4.30.1 Pre-Install Scripts

Executed before YaST does any real change to the system (before partitioning and package installation but after the hardware detection).

You can use a pre-script to modify your control file and let AutoYaST reread it. Find your control file in `/tmp/profile/autoinst.xml`. Adjust the file and store the modified version in `/tmp/profile/modified.xml`. AutoYaST will read the modified file after the pre-script finishes.

It is also possible to change the partitioning in your pre-script.

Note: Pre-Install Scripts with Confirmation

Pre-scripts are executed at an early stage of the installation. This means if you have requested to confirm the installation, the pre-scripts will be executed before the confirmation screen shows up (`profile/install/general/mode/confirm`).

Note: Pre-Install and Zypper

To call *zypper* in the pre-install script you will need to set the environment variable *ZYPP_LOCKFILE_ROOT="/var/run/autoyast"* to prevent conflicts with the running YaST process.

Pre-Install Script elements must be placed as follows:

```
<scripts>
  <pre-scripts config:type="list">
    <script>
      ...
    </script>
  </pre-scripts>
</scripts>
```

4.30.2 Post-partitioning Scripts

Executed after YaST has done the partitioning and written the fstab. The empty system is already mounted to `/mnt`.

Post-partitioning script elements must be placed as follows:

```
<scripts>
  <postpartitioning-scripts config:type="list">
    <script>
      ...
    </script>
  </postpartitioning-scripts>
```

```
</scripts>
```

4.30.3 Chroot Environment Scripts

Chroot scripts are executed before the machine reboots for the first time. You can execute chroot scripts before the installation chroots into the installed system and configures the boot loader or you can execute a script after the chroot into the installed system has happened (look at the `chrooted` parameter for that).

Chroot Environment script elements must be placed as follows:

```
<scripts>
  <chroot-scripts config:type="list">
    <script>
      ...
    </script>
  </chroot-scripts>
</scripts>
```

4.30.4 Post-Install Scripts

These scripts are executed after AutoYaST has completed the system configuration and after it has booted the system for the first time.

Post-install script elements must be placed as follows:

```
<scripts>
    <post-scripts config:type="list">
      <script>
        ...
      </script>
    </post-scripts>
  </scripts>
```

4.30.5 Init Scripts

These scripts are executed when YaST has finished, during the initial boot process after the network has been initialized. These final scripts are executed using `/usr/lib/YaST2/bin/autoyast-initscripts.sh` and are executed only once. Init scripts are configured using the tag *init-scripts*.

The following elements must be between the <scripts><init-scripts config:type="list"><script> ... </script></init-scripts>...</scripts> tags

TABLE 4.1: INIT SCRIPT XML REPRESENTATION

Element	Description	Comment
`location`	Define a location from where the script gets fetched. Locations can be the same as for the profile (HTTP, FTP, NFS, etc.). `<location >http://10.10.0.1/ myInitScript.sh</location>`	Either <location> or <source> must be defined.
`source`	The script itself (source code), encapsulated in a CDATA tag. If you do not want to put the whole shell script into the XML profile, use the location parameter. `<source> <![CDATA[ echo "Testing the init script" > /tmp/init_out.txt ]]> </source>`	Either <location> or <source> must be defined.
`filename`	The file name of the script. It will be stored in a temporary directory under `/tmp` `<filename>mynitScript5.sh</ filename>`	Optional in case you only have a single init script. The default name (`init-scripts`) is used in this case. If having specified more than one init script, you must set a unique name for each script.

Element	Description	Comment
`rerun`	A script is only run once. Even if you use ayast_setup to run an XML file multiple times, the script is only run once. Change this default behavior by setting this boolean to `true`. `<rerun config:type="boolean">true</rerun>`	Optional. Default is `false` (scripts only run once).

When added to the control file manually, scripts need to be included in a *CDATA* element to avoid confusion with the file syntax and other tags defined in the control file.

4.30.6 Script XML Representation

The XML elements described below can be used for all the script types described above, except for the `chrooted` element, which can only be used in chroot scripts.

TABLE 4.2: SCRIPT XML REPRESENTATION

Element	Description	Comment
`location`	Define a location from where the script gets fetched. Locations can be the same as for the control file (HTTP, FTP, NFS, etc.). `<location>http://10.10.0.1/myPreScript.sh</location>`	Either `location` or `source` must be defined.
`source`	The script itself (source code), encapsulated in a CDATA tag. If you do not want to put the whole shell script into the XML control file, refer to the location parameter. `<source>`	Either `location` or `source` must be defined.

Element	Description	Comment
	<![CDATA[echo "Testing the pre script" > /tmp/pre-script_out.txt]]> </source>	
`interpreter`	Specify the interpreter that must be used for the script. Supported options are shell and perl. <interpreter>perl</interpreter>	Optional (default is `shell`).
`file name`	The file name of the script. It will be stored in a temporary directory under `/tmp`. <filename>myPreScript5.sh</filename>	Optional. Default is the type of the script (pre-scripts in this case). If you have more than one script, you should define different names for each script.
`feedback`	If this boolean is `true`, output and error messages of the script (STDOUT and STDERR) will be shown in a pop-up. The user needs to confirm them via the OK button. <feedback config:type="boolean">true</feedback>	Optional, default is `false`.
`feedback_type`	This can be `message`, `warning` or `error`. Set the timeout for these pop-ups in the <report> section. <feedback_type>warning</feedback_type>	Optional, if missing, an always blocking pop-up is used.
`debug`	If this is `true`, every single line of a shell script is logged. Perl scripts are run with warnings turned on. <debug config:type="boolean">true</debug>	Optional, default is `true`.

Element	Description	Comment
`notification`	This text will be shown in a pop-up for the time the script is running in the background. `<notification>Please wait while script is running...</notification>`	Optional, if not configured, no notification pop-up will be shown.

`param-list`	It is possible to specify parameters given to the script being called. You may have more than one `param` entry. They are concatenated by a single space character on the script command line. If any shell quoting should be necessary (for example to protect embedded spaces) you need to include this.
	
	`<param-list config:type="list">`
	`  <param>par1</param>`
	`  <param>par2 par3</param>`
	`  <param>"par4.1 par4.2"</param>`
	`</param-list>`
		Optional, if not configured, no parameters get passed to script.

`rerun`	A script is only run once. Even if you use `ayast` setup to run an XML file multiple times, the script is only run once. Change this default behavior by setting this boolean to `true`. `<rerun config:type="boolean">true</rerun>`	Optional, default is `false` (scripts only run once).
`chrooted`	If set to `false`, the installed system remains mounted at `/mnt` and no chroot happens. The boot loader is not installed either at this stage. Setting it to `true` means, a chroot into `/mnt` is performed, where the installed system is mounted. The boot loader is installed, and if you want to change anything in the installed system, you do not need to use the `/mnt` prefix anymore.	Optional, default is `false`. This option is only available for chroot environment scripts.

Element	Description	Comment
	`<chrooted config:type="boolean" >true</chrooted>`	

4.30.7 Script Example

EXAMPLE 4.51: SCRIPT CONFIGURATION

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://www.suse.com/1.0/
configns">
<scripts>
  <chroot-scripts config:type="list">
    <script>
      <chrooted config:type="boolean">true</chrooted>
      <filename>chroot.sh</filename>
      <interpreter>shell</interpreter>
      <source><![CDATA[
#!/bin/sh
echo "Testing chroot (chrooted) scripts"
ls
]]>
      </source>
    </script>
    <script>
      <filename>chroot.sh</filename>
        <interpreter>shell</interpreter>
        <source><![CDATA[
#!/bin/sh
echo "Testing chroot scripts"
df
cd /mnt
ls
]]>
        </source>
      </script>
    </chroot-scripts>
    <post-scripts config:type="list">
      <script>
        <filename>post.sh</filename>
        <interpreter>shell</interpreter>
        <source><![CDATA[
#!/bin/sh
```

```
echo "Running Post-install script"
systemctl start portmap
mount -a 192.168.1.1:/local /mnt
cp /mnt/test.sh /tmp
umount /mnt
]]>
        </source>
      </script>
      <script>
        <filename>post.pl</filename>
        <interpreter>perl</interpreter>
        <source><![CDATA[
#!/usr/bin/perl
print "Running Post-install script";

]]>
        </source>
      </script>
    </post-scripts>
    <pre-scripts config:type="list">
      <script>
        <interpreter>shell</interpreter>
        <location>http://192.168.1.1/profiles/scripts/prescripts.sh</location>
      </script>
      <script>
        <filename>pre.sh</filename>
        <interpreter>shell</interpreter>
        <source><![CDATA[
#!/bin/sh
echo "Running pre-install script"
]]>
        </source>
      </script>
    </pre-scripts>
    <postpartitioning-scripts config:type="list">
      <script>
        <filename>postpart.sh</filename>
        <interpreter>shell</interpreter>
        <debug config:type="boolean">false</debug>
        <feedback config:type="boolean">true</feedback>
        <source><![CDATA[
touch /mnt/testfile
echo Hi
]]>
        </source>
      </script>
```

```
    </postpartitioning-scripts>
  </scripts>
</profile>
```

After installation is finished, the scripts and the output logs can be found in the directory `/var/adm/autoinstall`. The scripts are located in the subdirectory `scripts` and the output logs in the `log` directory.

The log consists of the output produced when executing the shell scripts using the following command:

```
/bin/sh -x SCRIPT_NAME 2&>/var/adm/autoinstall/logs/SCRIPT_NAME.log
```

4.31 System Variables (Sysconfig)

Using the sysconfig resource, it is possible to define configuration variables in the sysconfig repository (`/etc/sysconfig`) directly. Sysconfig variables, offer the possibility to fine-tune many system components and environment variables exactly to your needs.

The following example shows how a variable can be set using the sysconfig resource.

EXAMPLE 4.52: SYSCONFIG CONFIGURATION

```
<sysconfig config:type="list" >
  <sysconfig_entry>
    <sysconfig_key>XNTPD_INITIAL_NTPDATE</sysconfig_key>
    <sysconfig_path>/etc/sysconfig/xntp</sysconfig_path>
    <sysconfig_value>ntp.host.com</sysconfig_value>
  </sysconfig_entry>
  <sysconfig_entry>
    <sysconfig_key>HTTP_PROXY</sysconfig_key>
    <sysconfig_path>/etc/sysconfig/proxy</sysconfig_path>
    <sysconfig_value>proxy.host.com:3128</sysconfig_value>
  </sysconfig_entry>
  <sysconfig_entry>
    <sysconfig_key>FTP_PROXY</sysconfig_key>
    <sysconfig_path>/etc/sysconfig/proxy</sysconfig_path>
    <sysconfig_value>proxy.host.com:3128</sysconfig_value>
  </sysconfig_entry>
</sysconfig>
```

Both relative and absolute paths can be provided. If no absolute path is given, it is treated as a sysconfig file under the `/etc/sysconfig` directory.

4.32 Adding Complete Configurations

For many applications and services you may have a configuration file which should be copied to the appropriate location on the installed system. For example, if you are installing a Web server, you may have a server configuration file (`httpd.conf`).

Using this resource, you can embed the file into the control file by specifying the final path on the installed system. YaST will copy this file to the specified location.

This feature requires the `autoyast2` package to be installed. If the package is missing, AutoYaST will automatically install the package if it is missing.

You can specify the `file_location` where the file should be retrieved from. This can also be a location on the network such as an HTTP server: `<file_location>http://my.server.site/issue</file_location>`.

You can create directories by specifying a `file_path` that ends with a slash.

EXAMPLE 4.53: DUMPING FILES INTO THE INSTALLED SYSTEM

```
<files config:type="list">
  <file>
    <file_path>/etc/apache2/httpd.conf</file_path>
    <file_contents>

<![CDATA[
some content
]]>

    </file_contents>
  </file>
  <file>
    <file_path>/mydir/a/b/c/</file_path> <!-- create directory -->
  </file>
</files>
```

A more advanced example is shown below. This configuration will create a file using the content supplied in `file_contents` and change the permissions and ownership of the file. After the file has been copied to the system, a script is executed. This can be used to modify the file and prepare it for the client's environment.

EXAMPLE 4.54: DUMPING FILES INTO THE INSTALLED SYSTEM

```
<files config:type="list">
  <file>
```

```
    <file_path>/etc/someconf.conf</file_path>
    <file_contents>

<![CDATA[
some content
]]>

    </file_contents>
    <file_owner>tux.users</file_owner>
    <file_permissions>444</file_permissions>
    <file_script>
      <interpreter>shell</interpreter>
      <source>

<![CDATA[
#!/bin/sh

echo "Testing file scripts" >> /etc/someconf.conf
df
cd /mnt
ls
]]>

      </source>
    </file_script>
  </file>
</files>
```

4.33 Ask the User for Values during Installation

You have the option to let the user decide the values of specific parts of the control file during the installation. If you use this feature, a pop-up will ask the user to enter a specific part of the control file during installation. If you want a full auto installation, but the user should set the password of the local account, you can do this via the `ask` directive in the control file.

The elements listed below must be placed within the following XML structure:

```
<general>
  <ask-list config:type="list">
    <ask>
      ...
    </ask>
  </ask-list>
</general>
```

TABLE 4.3: ASK THE USER FOR VALUES: XML REPRESENTATION

Element	Description	Comment
`question`	The question you want to ask the user. `<question>Enter the LDAP server</question>`	The default value is the path to the element (the path often looks strange, so we recommend entering a question).
`default`	Set a preselection for the user. A text entry will be filled out with this value. A check box will be true or false and a selection will have the given value preselected. `<default>dc=suse,dc=de</default>`	Optional.
`help`	An optional help text that is shown on the left side of the question. `<help>Enter the LDAP server address.</help>`	Optional.
`title`	An optional title that is shown above the questions. `<title>LDAP server</title>`	Optional.
`type`	The type of the element you want to change. Possible values are `symbol`, `boolean`, `string` and `integer`. The file system in the partition section is a symbol, while	Optional. The default is `string`. If type is `symbol`, you must provide the selection element too (see below).

Element	Description	Comment
	the `encrypted` element in the user configuration is a boolean. You can see the type of that element if you look in your control file at the `config:type="...."` attribute. You can also use `static_text` as type. A `static_text` is a text that does not require any user input and can be used to show information not included in the help text. `<type>symbol</type>`	
`password`	If this boolean is set to `true`, a password dialog pops up instead of a simple text entry. Setting this to `true` only makes sense if `type` is string. `<password config:type="boolean">true</password>`	Optional. The default is `false`.
`pathlist`	A list of `path` elements. A path is a comma separated list of elements that describes the path to the element you want to change. For example, the LDAP server element can be found in the control file in the `<ldap><ldap_server>` section. So if you want to	This information is optional but you should at least provide `path` or `file`.

Element	Description	Comment

	change that value, you need to set the path to `ldap,ldap_server`.
	
	`<pathlist config:type="list"> <path>networking,dns,hostname</path> <path>...</path> </pathlist>`
	
	To change the password of the first user in the control file, you need to set the path to `users,0,user_password`. The `0` indicates the first user in the <users config:type="list"> list of users in the control file. `1` would be the second one, and so on.
	
	`<users config:type="list"> <user> <username>root</username> <user_password>password to change</user_password> <encrypted config:type="boolean">false</encrypted> </user> <user> <username>tux</username> <user_password>password to change</user_password> <encrypted config:type="boolean">false</encrypted> </user>`	

Element	Description	Comment
	`</users>`	
`file`	You can store the answer to a question in a file, to use it in one of your scripts later. If you ask during `stage=initial` and you want to use the answer in stage 2, then you need to copy the answer-file in a chroot script that is running as `chrooted=false`. Use the command: **`cp /tmp/my_answer /mnt/tmp/`**. The reason is that `/tmp` in stage 1 is in the RAM disk and will be lost after the reboot, but the installed system is already mounted at `/mnt/`. `<file>/tmp/answer_hostname</file>`	This information is optional, but you should at least provide `path` or `file`.
stage	Stage configures the installation stage in which the question pops up. You can set this value to `cont` or `initial`. `initial` means the pop-up comes up very early in the installation, shortly after the pre-script has run. `cont` means, that the dialog with the question comes after the first reboot when the system boots for the very first	Optional. The default is `initial`.

Element	Description	Comment
	time. Questions you answer during the `initial` stage will write their answer into the control file on the hard disk. You should know that if you enter clear text passwords during `initial`. Of course it does not make sense to ask for the file system to use during the `cont` phase. The hard disk is already partitioned at that stage and the question will have no effect. `<stage>cont</stage>`	

`selection`	The selection element contains a list of `entry` elements. Each entry represents a possible option for the user to choose. The user cannot enter a value in a text box, but he can choose from a list of values.
	`<selection`
	` config:type="list">`
	`  <entry>`
	`  <value>`
	`  btrfs`
	`  </value>`
	`  <label>`
	`  Btrfs File System`
	`  </label>`
	`  </entry>`
	`  <entry>`
	`  <value>`
	` ext3
		Optional for `type=string`, not possible for `type=boolean` and mandatory for `type=symbol`.

Element	Description	Comment
	<pre> </value> <label> Extended3 File System </label> </entry> </selection></pre>	

`dialog`	You can ask more than one question per dialog. To do so, specify the dialog-id with an integer. All questions with the same dialog-id belong to the same dialog. The dialogs are sorted by the id too.
	<pre><dialog
	 config:type="integer">3</
	dialog></pre>
		Optional.

`element`	you can have more than one question per dialog. To make that possible you need to specify the element-id with an integer. The questions in a dialog are sorted by id.
	<pre><element
	 config:type="integer">1</
	element></pre>
		Optional (see dialog).

`width`	You can increase the default width of dialog. If there are multiple width specifications per dialog, the largest one is used. The number is roughly equivalent to the number of characters.	Optional.

Element	Description	Comment
	`<width config:type="integer">50</width>`	
`height`	You can increase default height of dialog. If there are multiple height specifications per dialog, largest one is used. The number is roughly equivalent to number of lines. `<height config:type="integer">15</height>`	Optional.
`frametitle`	You can have more than one question per dialog. Each question on a dialog has a frame that can have a frame title, a small caption for each question. You can put multiple elements into one frame. They need to have the same frame title. `<frametitle>User data</frametitle>`	Optional. Default is no frame title.
`script`	You can run scripts after a question has been answered (see the table below for detailed instructions about scripts). `<script>...</script>`	Optional (default is no script).

Element	Description	Comment
`ok_label`	You can change the label on the *Ok* button. The last element that specifies the label for a dialog wins. `<ok_label>Finish</ok_label>`	Optional.
`back_label`	You can change the label on the *Back* button. The last element that specifies the label for a dialog wins. `<back_label>change values</back_label>`	Optional.
`timeout`	You can specify an integer here that is used as timeout in seconds. If the user does not answer the question before the timeout, the default value is taken as answer. When the user touches or changes any widget in the dialog, the timeout is turned off and the dialog needs to be confirmed via *Ok*. `<timeout config:type="integer">30</timeout>`	Optional. A missing value is interpreted as `0`, which means that there is no timeout.
`default_value_script`	You can run scripts to set the default value for a question (see *Section 4.33.1, "Default Value Scripts"* for detailed instructions about default val-	Optional. Default is no script.

Element	Description	Comment
	ue scripts). This feature is useful if you can `calculate` a default value, especially in combination with the `timeout` option. `<default_value_script>...</default_value_script>`	

4.33.1 Default Value Scripts

You can run scripts to set the default value for a question. This feature is useful if you can `calculate` a default value, especially in combination with the `timeout` option.

The elements listed below must be placed within the following XML structure:

```
<general>
  <ask-list config:type="list">
    <ask>
      <default_value_script>
        ...
      </default_value_script>
    </ask>
  </ask-list>
</general>
```

TABLE 4.4: DEFAULT VALUE SCRIPTS: XML REPRESENTATION

Element	Description	Comment
`source`	The source code of the script. Whatever you **`echo`** to STDOUT will be used as default value for the ask-dialog. If your script has an exit code other than 0, the normal default element is used. Take care you use **`echo -n`** to	This value is required, otherwise nothing would be executed.

Element	Description	Comment
	suppress the `\n` and that you echo reasonable values and not “okay” for a boolean `<source>...</source>`	
`interpreter`	The interpreter to use. `<interpreter>perl</interpreter>`	The default value is `shell`. You can also set `/bin/myinterpreter` as value.

4.33.2 Scripts

You can run scripts after a question has been answered.

The elements listed below must be placed within the following XML structure:

```
<general>
  <ask-list config:type="list">
    <ask>
      <script>
        ...
      </script>
    </ask>
  </ask-list>
</general>
```

TABLE 4.5: SCRIPTS: XML REPRESENTATION

Element	Description	Comment
`file_name`	The file name of the script. `<filename>my_ask_script.sh</filename>`	The default is ask_script.sh
`source`	The source code of the script. Together with `rerun_on_error` activated, you check the value that	This value is required, otherwise nothing would be executed.

Element	Description	Comment
	was entered for sanity. Your script can create a file `/tmp/next_dialog` with a dialog id specifying the next dialog AutoYaST will raise. A value of -1 terminates the ask sequence. If that file is not created, AutoYaST will run the dialogs in the normal order (since 11.0 only). `<source>...</source>`	
`environment`	A boolean that passes the value of the answer to the question as an environment variable to the script. The variable is named `VAL`. `<environment config:type="boolean">true</environment>`	Optional. Default is `false`.
`feedback`	A boolean that turns on feedback for the script execution. STDOUT will be displayed in a pop-up window that must be confirmed after the script execution. `<feedback config:type="boolean">true</feedback>`	Optional, default is `false`.
`debug`	A boolean that turns on debugging for the script execution.	Optional, default is `true`. This value needs `feedback` to be turned on, too.

Element	Description	Comment
	`<debug config:type="boolean">true</debug>`	
`rerun_on_error`	A boolean that keeps the dialog open until the script has an exit code of 0 (zero). So you can parse and check the answers the user gave in the script and display an error with the `feedback` option. `<rerun_on_error config:type="boolean">true</rerun_on_error>`	Optional, default is `false`. This value should be used together with the feedback option.

Below you can see an example of the usage of the `ask` feature.

```
<general>
  <ask-list config:type="list">
    <ask>
      <pathlist config:type="list">
        <path>ldap,ldap_server</path>
      </pathlist>
      <stage>cont</stage>
      <help>Choose your server depending on your department</help>
      <selection config:type="list">
        <entry>
          <value>ldap1.mydom.de</value>
          <label>LDAP for development</label>
        </entry>
        <entry>
          <value>ldap2.mydom.de</value>
          <label>LDAP for sales</label>
        </entry>
      </selection>
      <default>ldap2.mydom.de</default>
      <default_value_script>
        <source> <![CDATA[
echo -n "ldap1.mydom.de"
]]>
        </source>
```

```
      </default_value_script>
    </ask>
    <ask>
      <pathlist config:type="list">
        <path>networking,dns,hostname</path>
      </pathlist>
      <question>Enter Hostname</question>
      <stage>initial</stage>
      <default>enter your hostname here</default>
    </ask>
    <ask>
      <pathlist config:type="list">
        <path>partitioning,0,partitions,0,filesystem</path>
      </pathlist>
      <question>File System</question>
      <type>symbol</type>
      <selection config:type="list">
        <entry>
          <value config:type="symbol">reiser</value>
          <label>default File System (recommended)</label>
        </entry>
        <entry>
          <value config:type="symbol">ext3</value>
          <label>Fallback File System</label>
        </entry>
      </selection>
    </ask>
  </ask-list>
</general>
```

The following example shows a to choose between AutoYaST control files. AutoYaST will read the `modified.xml` file again after the ask-dialogs are done. This way you can fetch a complete new control file.

```
<general>
  <ask-list config:type="list">
    <ask>
      <selection config:type="list">
        <entry>
          <value>part1.xml</value>
          <label>Simple partitioning</label>
        </entry>
        <entry>
          <value>part2.xml</value>
          <label>encrypted /tmp</label>
        </entry>
```

```
        <entry>
          <value>part3.xml</value>
          <label>LVM</label>
        </entry>
      </selection>
      <title>XML Profile</title>
      <question>Choose a profile</question>
      <stage>initial</stage>
      <default>part1.xml</default>
      <script>
        <filename>fetch.sh</filename>
        <environment config:type="boolean">true</environment>
        <source>
<![CDATA[
wget http://10.10.0.162/$VAL -O /tmp/profile/modified.xml 2>/dev/null
]]>
        </source>
        <debug config:type="boolean">false</debug>
        <feedback config:type="boolean">false</feedback>
      </script>
    </ask>tion>
  </ask-list>
</general>
```

You can verify the answer of a question with a script like this:

```
<general>
  <ask-list config:type="list">
    <ask>
      <script>
        <filename>my.sh</filename>
        <rerun_on_error config:type="boolean">true</rerun_on_error>
        <environment config:type="boolean">true</environment>
        <source><![CDATA[
if [ "$VAL" = "myhost" ]; then
    echo "Illegal Hostname!";
    exit 1;
fi
exit 0
]]>
        </source>
        <debug config:type="boolean">false</debug>
        <feedback config:type="boolean">true</feedback>
      </script>
      <dialog config:type="integer">0</dialog>
      <element config:type="integer">0</element>
      <pathlist config:type="list">
```

```
        <path>networking,dns,hostname</path>
      </pathlist>
      <question>Enter Hostname</question>
      <default>enter your hostname here</default>
    </ask>
  </ask-list>
</general>
```

4.34 Kernel Dumps

Note: Availability

This feature is not available on the IBM Z (s390x) architecture.

With Kdump the system can create crashdump files if the whole kernel crashes. Crash dump files contain the memory contents while the system crashed. Such core files can be analyzed later by support or a (kernel) developer to find the reason for the system crash. Kdump is mostly useful for servers where you cannot easily reproduce such crashes but it is important to get the problem fixed.

There is a downside to this. Enabling Kdump requires between 64 MB and 128 MB of additional system RAM reserved for Kdump in case the system crashes and the dump needs to be generated.

This section only describes how to set up Kdump with AutoYaST. It does not describe how Kdump works. For details, refer to the kdump(7) manual page.

The following example shows a general Kdump configuration.

EXAMPLE 4.55: KDUMP CONFIGURATION

```
<kdump>
  <!-- memory reservation -->
  <add_crash_kernel config:type="boolean">true</add_crash_kernel>
  <crash_kernel>256M-:64M</crash_kernel>
  <general>

    <!-- dump target settings -->
    <KDUMP_SAVEDIR>ftp://stravinsky.suse.de/incoming/dumps</KDUMP_SAVEDIR>
    <KDUMP_COPY_KERNEL>true</KDUMP_COPY_KERNEL>
    <KDUMP_FREE_DISK_SIZE>64</KDUMP_FREE_DISK_SIZE>
    <KDUMP_KEEP_OLD_DUMPS>5</KDUMP_KEEP_OLD_DUMPS>
```

```
    <!-- filtering and compression -->
    <KDUMP_DUMPFORMAT>compressed</KDUMP_DUMPFORMAT>
    <KDUMP_DUMPLEVEL>1</KDUMP_DUMPLEVEL>

    <!-- notification -->
    <KDUMP_NOTIFICATION_TO>tux@example.com</KDUMP_NOTIFICATION_TO>
    <KDUMP_NOTIFICATION_CC>spam@example.com devnull@example.com</KDUMP_NOTIFICATION_CC>
    <KDUMP_SMTP_SERVER>mail.example.com</KDUMP_SMTP_SERVER>
    <KDUMP_SMTP_USER></KDUMP_SMTP_USER>
    <KDUMP_SMTP_PASSWORD></KDUMP_SMTP_PASSWORD>

    <!-- kdump kernel -->
    <KDUMP_KERNELVER></KDUMP_KERNELVER>
    <KDUMP_COMMANDLINE></KDUMP_COMMANDLINE>
    <KDUMP_COMMANDLINE_APPEND></KDUMP_COMMANDLINE_APPEND>

    <!-- expert settings -->
    <KDUMP_IMMEDIATE_REBOOT>yes</KDUMP_IMMEDIATE_REBOOT>
    <KDUMP_VERBOSE>15</KDUMP_VERBOSE>
    <KEXEC_OPTIONS></KEXEC_OPTIONS>
  </general>
</kdump>
```

4.34.1 Memory Reservation

The first step is to reserve memory for Kdump at boot-up. Because the memory must be reserved very early during the boot process, the configuration is done via a kernel command line parameter called `crashkernel`. The reserved memory will be used to load a second kernel which will be executed without rebooting if the first kernel crashes. This second kernel has a special initrd, which contains all programs necessary to save the dump over the network or to disk, send a notification e-mail, and finally reboot.

To reserve memory for Kdump, specify the `amount` (such as `64M` to reserve 64 MB of memory from the RAM) and the `offset`. The syntax is `crashkernel=AMOUNT@OFFSET`. The kernel can auto-detect the right offset (except for the Xen hypervisor, where you need to specify `16M` as offset). The amount of memory that needs to be reserved depends on architecture and main memory. Refer to *Book "*System Analysis and Tuning Guide*", Chapter 17 "Kexec and Kdump", Section 17.7.1 "Manual Kdump Configuration"* for recommendations on the amount of memory to reserve for Kdump.

You can also use the extended command line syntax to specify the amount of reserved memory depending on the System RAM. That is useful if you share one AutoYaST control file for multiple installations or if you often remove or install memory on one machine. The syntax is:

```
BEGIN_RANGE_1-END_RANGE_1:AMOUNT_1,BEGIN_RANGE_2-END_RANGE_2:AMOUNT_2@OFFSET
```

`BEGIN_RANGE_1` is the start of the first memory range (for example: `0M`) and `END_RANGE_1` is the end of the first memory range (can be empty in case `infinity` should be assumed) and so on. For example, `256M-2G:64M,2G-:128M` reserves 64 MB of crashkernel memory if the system has between 256 MB and 2 GB RAM and reserves 128 MB of crashkernel memory if the system has more than 2 GB RAM.

On the other hand, it is possible to specify multiple values for the crashkernel parameter. For example, when you need to reserve different segments of low and high memory, use values like `72M,low` and `256M,high`:

EXAMPLE 4.56: KDUMP MEMORY RESERVATION WITH MULTIPLE VALUES

```
<kdump>
  <!-- memory reservation (high and low) -->
  <add_crash_kernel config:type="boolean">true</add_crash_kernel>
  <crash_kernel config:type="list">
    <listentry>72M,low</listentry>
    <listentry>256M,high</listentry>
  </crash_kernel>
</kdump>
```

The following table shows the settings necessary to reserve memory:

TABLE 4.6: KDUMP MEMORY RESERVATION SETTINGS:XML REPRESENTATION

Element	Description	Comment
`add_crash_kernel`	Set to `true` if memory should be reserved and Kdump enabled. `<add_crash_kernel config:type="boolean">true</add_crash_kernel>`	required
`crash_kernel`	Use the syntax of the crashkernel command line as discussed above.	required

Element	Description	Comment

	`<crash_kernel>256M:64M</crash_kernel>`
	A list of values is also supported.
	`<crash_kernel config:type="list"> <listentry>72M,low</listentry> <listentry>256M,high</listentry> </crash_kernel>`	

4.34.2 Dump Saving

4.34.2.1 Target

The element `KDUMP_SAVEDIR` specifies the URL to where the dump is saved. The following methods are possible:

- `file` to save to the local disk,
- `ftp` to save to an FTP server (without encryption),
- `sftp` to save to an SSH2 SFTP server,
- `nfs` to save to an NFS location and
- `cifs` to save the dump to a CIFS/SMP export from Samba or Microsoft Windows.

For details see the kdump(5) manual page. Two examples are: `file:///var/crash` (which is the default location according to FHS) and `ftp://user:password@host:port/incoming/dumps`. A subdirectory, with the time stamp contained in the name, will be created and the dumps saved there.

When the dump is saved to the local disk, `KDUMP_KEEP_OLD_DUMPS` can be used to delete old dumps automatically. Set it to the number of old dumps that should be kept. If the target partition would end up with less free disk space than specified in `KDUMP_FREE_DISK_SIZE`, the dump is not saved.

To save the whole kernel and the debug information (if installed) to the same directory, set `KDUMP_COPY_KERNEL` to `true`. You will have everything you need to analyze the dump in one directory (except kernel modules and their debugging information).

4.34.2.2 Filtering and Compression

The kernel dump is uncompressed and unfiltered. It can get as large as your system RAM. To get smaller files, compress the dump file afterward. The dump needs to be decompressed before opening.

To use page compression, which compresses every page and allows dynamic decompression with the crash(8) debugging tool, set `KDUMP_DUMPFORMAT` to `compressed` (default).

You may not want to save all memory pages, for example those filled with zeroes. To filter the dump, set the `KDUMP_DUMPLEVEL`. 0 produces a full dump and 31 is the smallest dump. The manual pages kdump(5) and makedumpfile(8) list for each value which pages will be saved.

4.34.2.3 Summary

TABLE 4.7: DUMP TARGET SETTINGS: XML REPRESENTATION

Element	Description	Comment
`KDUMP_SAVEDIR`	A URL that specifies the target to which the dump and related files will be saved. `<KDUMP_SAVEDIR>file:///var/crash/</KDUMP_SAVEDIR>`	required
`KDUMP_COPY_KERNEL`	Set to `true`, if not only the dump should be saved to `KDUMP_SAVEDIR` but also the kernel and its debugging information (if installed).	optional

Element	Description	Comment
	`<KDUMP_COPY_KERNEL>false</KDUMP_COPY_KERNEL>`	
KDUMP_FREE_DISK_SIZE	Disk space in megabytes that must remain free after saving the dump. If not enough space is available, the dump will not be saved. `<KDUMP_FREE_DISK_SIZE>64</KDUMP_FREE_DISK_SIZE>`	optional
KDUMP_KEEP_OLD_DUMPS	The number of dumps that are kept (not deleted) if KDUMP_SAVEDIR points to a local directory. Specify 0 if you do not want any dumps to be automatically deleted, specify -1 if all dumps except the current one should be deleted. `<KDUMP_KEEP_OLD_DUMPS>4</KDUMP_KEEP_OLD_DUMPS>`	optional

4.34.3 E-Mail Notification

Configure e-mail notification if you want to be informed when a machine crashes and a dump is saved.

Because Kdump runs in the initrd, a local mail server cannot send the notification e-mail. An SMTP server needs to be specified (see below).

You need to provide exactly one address in KDUMP_NOTIFICATION_TO. More addresses can be specified in KDUMP_NOTIFICATION_CC. Only use e-mail addresses in both cases, not a real name.

Specify KDUMP_SMTP_SERVER and (if the server needs authentication) KDUMP_SMTP_USER and KDUMP_SMTP_PASSWORD. Support for TLS/SSL is not available but may be added in the future.

TABLE 4.8: E-MAIL NOTIFICATION SETTINGS: XML REPRESENTATION

Element	Description	Comment
KDUMP_NOTIFICATION_TO	Exactly one e-mail address to which the e-mail should be sent. Additional recipients can be specified in KDUMP_NOTIFICATION_CC. `<KDUMP_NOTIFICATION_TO >tux@example.com</ KDUMP_NOTIFICATION_TO>`	optional (notification disabled if empty)
KDUMP_NOTIFICATION_CC	Zero, one or more recipients that are in the cc line of the notification e-mail. `<KDUMP_NOTIFICATION_CC >wilber@example.com geeko@example.com</ KDUMP_NOTIFICATION_CC>`	optional
KDUMP_SMTP_SERVER	Host name of the SMTP server used for mail delivery. SMTP authentication is supported (see KDUMP_SMTP_USER and KDUMP_SMTP_PASSWORD) but TLS/SSL are not. `<KDUMP_SMTP_SERVER>email.suse.de</ KDUMP_SMTP_SERVER>`	optional (notification disabled if empty)
KDUMP_SMTP_USER	User name used together with KDUMP_SMTP_PASSWORD for SMTP authentication. `<KDUMP_SMTP_USER>bwalle</ KDUMP_SMTP_USER>`	optional

Element	Description	Comment
KDUMP_SMTP_PASSWORD	Password used together with KDUMP_SMTP_USER for SMTP authentication. <KDUMP_SMTP_PASSWORD>geheim</KDUMP_SMTP_PASSWORD>	optional

4.34.4 Kdump Kernel Settings

As already mentioned, a special kernel is booted to save the dump. If you do not want to use the auto-detection mechanism to find out which kernel is used (see the kdump(5) manual page that describes the algorithm which is used to find the kernel), you can specify the version of a custom kernel in KDUMP_KERNELVER. If you set it to foo, then the kernel located in /boot/vmlinuz-foo or /boot/vmlinux-foo (in that order on platforms that have a vmlinuz file) will be used.

You can specify the command line used to boot the Kdump kernel. Normally the boot command line is used, minus settings that are not relevant for Kdump (like the crashkernel parameter) plus some settings needed by Kdump (see the manual page kdump(5)). To specify additional parameters, use KDUMP_COMMANDLINE_APPEND. If you know what you are doing and you want to specify the entire command line, set KDUMP_COMMANDLINE.

TABLE 4.9: KERNEL SETTINGS: XML REPRESENTATION

Element	Description	Comment
KDUMP_KERNELVER	Version string for the kernel used for Kdump. Leave it empty to use the auto-detection mechanism (strongly recommended). <KDUMP_KERNELVER>4.12.14-94.37-default</KDUMP_KERNELVER>	optional (auto-detection if empty)

Element	Description	Comment
KDUMP_COMMANDLINE_APPEND	Additional command line parameters for the Kdump kernel. <KDUMP_COMMANDLINE_APPEND>console=ttyS0,57600</KDUMP_COMMANDLINE_APPEND>	optional
KDUMP_Command Line	Overwrite the automatically generated Kdump command line. Use with care. Usually, KDUMP_COMMANDLINE_APPEND should suffice. <KDUMP_COMMANDLINE_APPEND>root=/dev/sda5 maxcpus=1 irqpoll</KDUMP_COMMANDLINE>	optional

4.34.5 Expert Settings

TABLE 4.10: EXPERT SETTINGS: XML REPRESENTATIONS

Element	Description	Comment
KDUMP_IMMEDIATE_REBOOT	true if the system should be rebooted automatically after the dump has been saved, false otherwise. The default is to reboot the system automatically. <KDUMP_IMMEDIATE_REBOOT>true</KDUMP_IMMEDIATE_REBOOT>	optional

Element	Description	Comment
KDUMP_VERBOSE	Bitmask that specifies how verbose the Kdump process should be. Read kdump(5) for details. <KDUMP_VERBOSE>3</KDUMP_VERBOSE>	optional
KEXEC_OPTIONS	Additional options that are passed to kexec when loading the Kdump kernel. Normally empty. <KEXEC_OPTIONS>--noio</KEXEC_OPTIONS>	optional

4.35 DNS Server

The Bind DNS server can be configured by adding a dns-server resource. The three more straightforward properties of that resource can have a value of 1 to enable them or 0 to disable.

Attribute	Value	Description
chroot	0 / 1	The DNS server must be jailed in a chroot.
start_service	0 / 1	Bind is enabled (executed on system start).
use_ldap	0 / 1	Store the settings in LDAP instead of native configuration files.

EXAMPLE 4.57: BASIC DNS SERVER SETTINGS

```
<dns-server>
  <chroot>0</chroot>
```

```
  <start_service>1</start_service>
  <use_ldap>0</use_ldap>
</dns-server>
```

In addition to those basic settings, there are three properties of type list that can be used to fine-tune the service configuration.

List	Description
`logging`	Options of the DNS server logging.
`options`	Bind options like the files and directories to use, the list of forwarders and other configuration settings.
`zones`	List of DNS zones known by the server, including all the settings, records and SOA records.

EXAMPLE 4.58: CONFIGURING DNS SERVER ZONES AND ADVANCED SETTINGS

```
<dns-server>
  <logging config:type="list">
    <listentry>
      <key>channel</key>
      <value>log_syslog { syslog; }</value>
    </listentry>
  </logging>
  <options config:type="list">
    <option>
      <key>forwarders</key>
      <value>{ 10.10.0.1; }</value>
    </option>
  </options>
  <zones config:type="list">
    <listentry>
      <is_new>1</is_new>
      <modified>1</modified>
      <options config:type="list"/>
      <records config:type="list">
        <listentry>
          <key>mydom.uwe.</key>
          <type>MX</type>
          <value>0 mail.mydom.uwe.</value>
```

```
        </listentry>
        <listentry>
          <key>mydom.uwe.</key>
          <type>NS</type>
          <value>ns.mydom.uwe.</value>
        </listentry>
      </records>
      <soa>
        <expiry>1w</expiry>
        <mail>root.aaa.aaa.cc.</mail>
        <minimum>1d</minimum>
        <refresh>3h</refresh>
        <retry>1h</retry>
        <serial>2005082300</serial>
        <server>aaa.aaa.cc.</server>
        <zone>@</zone>
      </soa>
      <soa_modified>1</soa_modified>
      <ttl>2d</ttl>
      <type>master</type>
      <update_actions config:type="list">
        <listentry>
          <key>mydom.uwe.</key>
          <operation>add</operation>
          <type>NS</type>
          <value>ns.mydom.uwe.</value>
        </listentry>
      </update_actions>
      <zone>mydom.uwe</zone>
    </listentry>
  </zones>
</dns-server>
```

4.36 DHCP Server

The `dhcp-server` resource makes it possible to configure all the settings of a DHCP server by means of the six following properties.

Element	Value	Description
`chroot`	0 / 1	A value of 1 means that the DHCP server must be jailed in a chroot.

Element	Value	Description
start_service	0 / 1	Set this to 1 to enable the DHCP server (that is, run it on system startup).
use_ldap	0 / 1	If set to 1, the settings will be stored in LDAP instead of native configuration files.
other_options	Text	String with parameters that will be passed to the DHCP server executable when started. For example, use "-p 1234" to listen on a non-standard 1234 port. For all possible options, consult the dhcpd manual page. If left blank, default values will be used.
allowed_interfaces	List	List of network cards in which the DHCP server will be operating. See the example below for the exact format.
settings	List	List of settings to configure the behavior of the DHCP server. The configuration is defined in a tree-like structure where the root represents the global options, with subnets and host nested from there. The children, parent_id and parent_type properties are used to rep-

Element	Value	Description
		resent that nesting. See the example below for the exact format.

EXAMPLE 4.59: EXAMPLE DHCP-SERVER SECTION

```
<dhcp-server>
  <allowed_interfaces config:type="list">
    <allowed_interface>eth0</allowed_interface>
  </allowed_interfaces>
  <chroot>0</chroot>
  <other_options>-p 9000</other_options>
  <start_service>1</start_service>
  <use_ldap>0</use_ldap>

  <settings config:type="list">
    <settings_entry>
      <children config:type="list"/>
      <directives config:type="list">
        <listentry>
          <key>fixed-address</key>
          <type>directive</type>
          <value>192.168.0.10</value>
        </listentry>
        <listentry>
          <key>hardware</key>
          <type>directive</type>
          <value>ethernet d4:00:00:bf:00:00</value>
        </listentry>
      </directives>
      <id>static10</id>
      <options config:type="list"/>
      <parent_id>192.168.0.0 netmask 255.255.255.0</parent_id>
      <parent_type>subnet</parent_type>
      <type>host</type>
    </settings_entry>
    <settings_entry>
      <children config:type="list">
        <child>
          <id>static10</id>
          <type>host</type>
        </child>
      </children>
      <directives config:type="list">
        <listentry>
```

```
          <key>range</key>
          <type>directive</type>
          <value>dynamic-bootp 192.168.0.100 192.168.0.150</value>
        </listentry>
        <listentry>
          <key>default-lease-time</key>
          <type>directive</type>
          <value>14400</value>
        </listentry>
        <listentry>
          <key>max-lease-time</key>
          <type>directive</type>
          <value>86400</value>
        </listentry>
      </directives>
      <id>192.168.0.0 netmask 255.255.255.0</id>
      <options config:type="list"/>
      <parent_id/>
      <parent_type/>
      <type>subnet</type>
    </settings_entry>
    <settings_entry>
      <children config:type="list">
        <child>
          <id>192.168.0.0 netmask 255.255.255.0</id>
          <type>subnet</type>
        </child>
      </children>
      <directives config:type="list">
        <listentry>
          <key>ddns-update-style</key>
          <type>directive</type>
          <value>none</value>
        </listentry>
        <listentry>
          <key>default-lease-time</key>
          <type>directive</type>
          <value>14400</value>
        </listentry>
      </directives>
      <id/>
      <options config:type="list"/>
      <parent_id/>
      <parent_type/>
      <type/>
    </settings_entry>
  </settings>
```

```
</dhcp-server>
```

4.37 SUSE Firewall

SUSE Firewall can be configured using the `firewall` resource. All the properties in this resource are optional and all of them are of type text (except the boolean `start_firewall` and `enable_firewall` properties).

4.37.1 General Firewall Configuration

The following properties are intended to configure the general settings of SUSE Firewall. Most of them are a direct representation of the corresponding setting at `/etc/sysconfig/SuSEfirewall2`. Check the comments in that file for further information.

Attribute	Value	Description
`start_firewall`	Boolean	Whether SUSE Firewall should be started right after applying the configuration.
`enable_firewall`	Boolean	Whether SUSE Firewall should be started on every system startup.
`FW_LOG_ACCEPT_ALL`	yes / no	Log every accepted package. If set to "yes" the value of `FW_LOG_ACCEPT_CRIT` becomes irrelevant.
`FW_LOG_ACCEPT_CRIT`	yes / no	Log accepted critical packages.
`FW_LOG_DROP_ALL`	yes / no	Log every dropped package. If set to "yes" the value of `FW_LOG_DROP_CRIT` becomes irrelevant.

Attribute	Value	Description
FW_LOG_DROP_CRIT	yes / no	Log dropped critical packages.
FW_ALLOW_PING_FW	yes / no	Allow the firewall to reply to ICMP echo requests.
FW_MASQUERADE	yes / no	Used to enable network masquerading, which allows to transparently redirect ports from one interface in the external zone to ports of another interface in a different zone. Masquerading needs at least one external interface and one other (not external) interface. Since routing is needed for masquerading, the values of this property and FW_ROUTE are connected.
FW_FORWARD_MASQ	Space delimited list of rules	Rules for network masquerading, with each rule following this format: source_network,ip_to_forward_to,protocol,port[,redirect_port,[destination_ip]]
FW_FORWARD_ALWAYS_INOUT_DEV	Space separated list of interface names	Bridge interfaces without IP address.
FW_IPSEC_TRUST	yes / no / int /ext / dmz	Trust level of IPsec packets.

Attribute	Value	Description
FW_LOAD_MODULES	Space delimited list	Additional kernel modules to load at startup.
FW_ROUTE	yes / no	Whether routing between external, dmz and internal network should be activated. Related to FW_MASQUERADE.
FW_PROTECT_FROM_INT	yes / no / notrack	Whether to protect the firewall from the internal network.

4.37.2 Firewall Zones Configuration

The configuration of SUSE Firewall is based on the existence of three network zones. The behavior of each zone can be tweaked in several ways. Therefore, there are many almost equivalent AutoYaST properties that differ only by name and the zone to which they apply. For example, FW_DEV_DMZ for the demilitarized zone, FW_DEV_EXT for the external zone and FW_DEV_INT for the internal one.

Attributes	Value	Description
FW_ALLOW_FW_BROADCAST_{DMZ/EXT/INT}	yes / no	Allow IP broadcasts in that zone.
FW_CONFIGURATIONS_{DMZ/EXT/INT}	Space delimited list	Services that should be accessible from that zone.
FW_DEV_{DMZ/EXT/INT}	Space delimited list	Name of the interfaces that are considered to belong to the zone. The special keyword "any" means that packets arriving on interfaces not

Attributes	Value	Description
		explicitly configured as int, ext or dmz will be considered to belong to this zone.
`FW_IGNORE_FW_BROADCAST_{DMZ/EXT/INT}`	yes / no	Suppress logging of dropped broadcast packets. Useful if you do not allow broadcasts on a LAN interface.
`FW_SERVICES_ACCEPT_{DMZ/EXT/INT}`	Space separated list of rules	Services to allow. Each rule following the format `net,protocol[,dport[,sport[,flags]]]`. For example, "0/0,tcp,22".
`FW_SERVICES_ACCEPT_RELATED_{DMZ/EXT/INT}`	Space delimited list of rules	Services to allow that are considered RELATED by the connection tracking engine. Format of each rule: `net,protocol[,sport[,dport]]`. For example, to allow Samba broadcast replies marked as related by `nf_conntrack_netbios_ns` from a certain network: "192.168.1.0/24,udp,137".
`FW_SERVICES_{DMZ/EXT/INT}_IP`	Space separated list	Which IP services should be accessible from the zone. Every entry in the list can be a port, a port range or a well known protocol name. This setting has precedence over `FW_SERVICES_ACCEPT_*`.

Attributes	Value	Description
FW_SERVICES_{DMZ/EXT/IN-T}_RPC	Space delimited list	RPC services that should be accessible from the zone. This setting has precedence over FW_SERVICES_ACCEP-T_*.
FW_SERVICES_{DMZ/EXT/IN-T}_TCP	Space separated list	Which TCP services should be accessible from the zone. Every entry in the list can be a port, a port range or a well known protocol name. This setting has precedence over FW_SERVICES_ACCEPT_*.
FW_SERVICES_{DMZ/EXT/IN-T}_UDP	Space separated list	Which UDP services should be accessible from the zone. Every entry in the list can be a port, a port range or a well known protocol name. This setting has precedence over FW_SERVICES_ACCEPT_*.

4.37.3 A Full Example

A full example of the firewall section, including general and zone specific properties could look like this.

EXAMPLE 4.60: EXAMPLE FIREWALL SECTION

```
<firewall>
  <FW_ALLOW_FW_BROADCAST_DMZ>no</FW_ALLOW_FW_BROADCAST_DMZ>
  <FW_ALLOW_FW_BROADCAST_EXT>no</FW_ALLOW_FW_BROADCAST_EXT>
  <FW_ALLOW_FW_BROADCAST_INT>no</FW_ALLOW_FW_BROADCAST_INT>
  <FW_DEV_DMZ></FW_DEV_DMZ>
  <FW_DEV_EXT>any eth0</FW_DEV_EXT>
  <FW_DEV_INT></FW_DEV_INT>
  <FW_FORWARD_ALWAYS_INOUT_DEV></FW_FORWARD_ALWAYS_INOUT_DEV>
```

```
  <FW_FORWARD_MASQ></FW_FORWARD_MASQ>
  <FW_IGNORE_FW_BROADCAST_DMZ>no</FW_IGNORE_FW_BROADCAST_DMZ>
  <FW_IGNORE_FW_BROADCAST_EXT>yes</FW_IGNORE_FW_BROADCAST_EXT>
  <FW_IGNORE_FW_BROADCAST_INT>no</FW_IGNORE_FW_BROADCAST_INT>
  <FW_IPSEC_TRUST>no</FW_IPSEC_TRUST>
  <FW_LOAD_MODULES>nf_conntrack_netbios_ns</FW_LOAD_MODULES>
  <FW_LOG_ACCEPT_ALL>no</FW_LOG_ACCEPT_ALL>
  <FW_LOG_ACCEPT_CRIT>yes</FW_LOG_ACCEPT_CRIT>
  <FW_LOG_DROP_ALL>no</FW_LOG_DROP_ALL>
  <FW_LOG_DROP_CRIT>yes</FW_LOG_DROP_CRIT>
  <FW_MASQUERADE>no</FW_MASQUERADE>
  <FW_PROTECT_FROM_INT>no</FW_PROTECT_FROM_INT>
  <FW_ROUTE>no</FW_ROUTE>
  <enable_firewall config:type="boolean">true</enable_firewall>
  <start_firewall config:type="boolean">true</start_firewall>
</firewall>
```

4.38 Miscellaneous Hardware and System Components

In addition to the core component configuration, like network authentication and security, AutoYaST offers a wide range of hardware and system configuration options, the same as available by default on any system installed manually and in an interactive way. For example, it is possible to configure printers, sound devices, TV cards and any other hardware components which have a module within YaST.

Any new configuration options added to YaST will be automatically available in AutoYaST.

4.38.1 Printer

AutoYaST support for printing is limited to basic settings defining how CUPS is used on a client for printing via the network.

There is no AutoYaST support for setting up local print queues. Modern printers are usually connected via USB. CUPS accesses USB printers by a model-specific device URI like `usb://ACME/FunPrinter?serial=1a2b3c`. Usually it is not possible to predict the correct USB device URI in advance, because it is determined by the CUPS back-end `usb` during runtime. Therefore it is not possible to set up local print queues with AutoYaST.

Basics on how CUPS is used on a client workstation to print via network:

On client workstations application programs submit print jobs to the CUPS daemon process (`cupsd`). `cupsd` forwards the print jobs to a CUPS print server in the network where the print jobs are processed. The server sends the printer specific data to the printer device.

If there is only a single CUPS print server in the network, there is no need to have a CUPS daemon running on each client workstation. Instead it is simpler to specify the CUPS server in `/etc/cups/client.conf` and access it directly (only one CUPS server entry can be set). In this case application programs that run on client workstations submit print jobs directly to the specified CUPS print server.

Example 4.61, "Printer configuration" shows a `printer` configuration section. The `cupsd_conf_content` entry contains the whole verbatim content of the `cupsd` configuration file `/etc/cups/cupsd.conf`. The `client_conf_content` entry contains the whole verbatim content of `/etc/cups/client.conf`. The `printer` section contains the `cupsd` configuration but it does not specify whether the cupsd should run.

EXAMPLE 4.61: PRINTER CONFIGURATION

```
  <printer>
    <client_conf_content>
      <file_contents><![CDATA[
... verbatim content of /etc/cups/client.conf ...
]]></file_contents>
    </client_conf_content>
    <cupsd_conf_content>
      <file_contents><![CDATA[
... verbatim content of /etc/cups/cupsd.conf ...
]]></file_contents>
    </cupsd_conf_content>
  </printer>
```

Note: `/etc/cups/cups-files.conf`

With release 1.6 the CUPS configuration file has been split into two files: `cupsd.conf` and `cups-files.conf`. As of SUSE Linux Enterprise Server 12 SP5, AutoYaST only supports modifying `cupsd.conf` since the default settings in `cups-files.conf` are sufficient for usual printing setups.

4.38.2 Sound devices

An example of the sound configuration created using the configuration system is shown below.

EXAMPLE 4.62: SOUND CONFIGURATION

```
<sound>
  <autoinstall config:type="boolean">true</autoinstall>
  <modules_conf config:type="list">
    <module_conf>
      <alias>snd-card-0</alias>
      <model>M5451, ALI</model>
      <module>snd-ali5451</module>
      <options>
        <snd_enable>1</snd_enable>
        <snd_index>0</snd_index>
        <snd_pcm_channels>32</snd_pcm_channels>
      </options>
    </module_conf>
  </modules_conf>
  <volume_settings config:type="list">
    <listentry>
      <Master config:type="integer">75</Master>
    </listentry>
  </volume_settings>
</sound>
```

4.39 Importing SSH Keys and Configuration

YaST allows to import SSH keys and server configuration from previous installations. The behavior of this feature can also be controlled through an AutoYaST profile.

EXAMPLE 4.63: IMPORTING SSH KEYS AND CONFIGURATION FROM /DEV/SDA2

```
<ssh_import>
  <import config:type="boolean">true</import>
  <copy_config config:type="boolean">true</copy_config>
  <device>/dev/sda2</device>
</ssh_import>
```

Attributes	Value	Description
`import`	true / false	SSH keys will be imported. If set to `false`, nothing will be imported.
`copy_config`	true / false	Additionally, SSH server configuration will be imported. This setting will not have effect if `import` is set to `false`.
`device`	Partition	Partition to import keys and configuration from. If it is not set, the partition which contains the most recently accessed key is used.

4.40 Configuration Management

AutoYaST allows delegating part of the configuration to a configuration management tool like Salt:

- AutoYaST takes care of system installation (partitioning, network setup, etc.)
- System configuration can be delegated to a configuration management tool

This module configures the connection to a configuration management tool and uploads SSH keys which are needed for establishing connections. At the end of the installation, the configuration management Master will be contacted to retrieve state files and other resources.

EXAMPLE 4.64: CONFIGURING SALT MANAGER

```
<configuration_management>
    <type>salt</type>
    <master>linux-addc</master>
    <auth_attempts config:type="integer">5</auth_attempts>
    <auth_time_out config:type="integer">10</auth_time_out>
    <keys_url>http://keys.example.de/keys</keys_url>
```

```
</configuration_management>
```

Attributes	Value	Description
`type`	Configuration management type	Configuration management name. Currently only `salt` is supported.
`master`	Host name	Host name or IP address of the configuration management master.
`auth_attempts`	Integer	At the end of installation, YaST connects to the configuration management master with maximum `auth_attempts` attempts. The default is 3 attempts.
`auth_time_out`	Integer	Time between the configuration management master connection attempts. The default is 15 seconds.
`keys_url`	URL of used key	Path to an HTTP server, hard disk, USB drive or similar with the files `default.key` and `default.pub`. This key has to be known to the configuration management master.
`enable_services`	True/false	Enables the configuration management services on the client side. Default is `true`.

5 Rules and Classes

5.1 Rules-based Automatic Installation

Rules offer the possibility to configure a system depending on system attributes by merging multiple control files during installation. The rules-based installation is controlled by a rules file. This is useful to install, for example, systems in two departments in one go. Assume a scenario where machines in department A need to be installed as office desktops, whereas machines in department B need to be installed as developer workstations. You would create a rules file with two different rules. For each rule, you could use different system parameters to distinguish the installations from one another. Each rule would also contain a link to an appropriate profile for each department.

The rules file is an XML file containing rules for each group of systems (or single systems) that you want to automatically install. A set of rules distinguish a group of systems based on one or more system attributes. After passing all rules, each group of systems is linked to a control file. Both the rules file and the control files must be located in a pre-defined and accessible location.

The rules file is retrieved only if no specific control file is supplied using the `autoyast` keyword. For example, if the following is used, the rules file will not be evaluated:

```
autoyast=http://10.10.0.1/profile/myprofile.xml
autoyast=http://10.10.0.1/profile/rules/rules.xml
```

Instead use:

```
autoyast=http://10.10.0.1/profile/
```

which will load `http://10.10.0.1/profile/rules/rules.xml` (the slash at the end of the directory name is important).

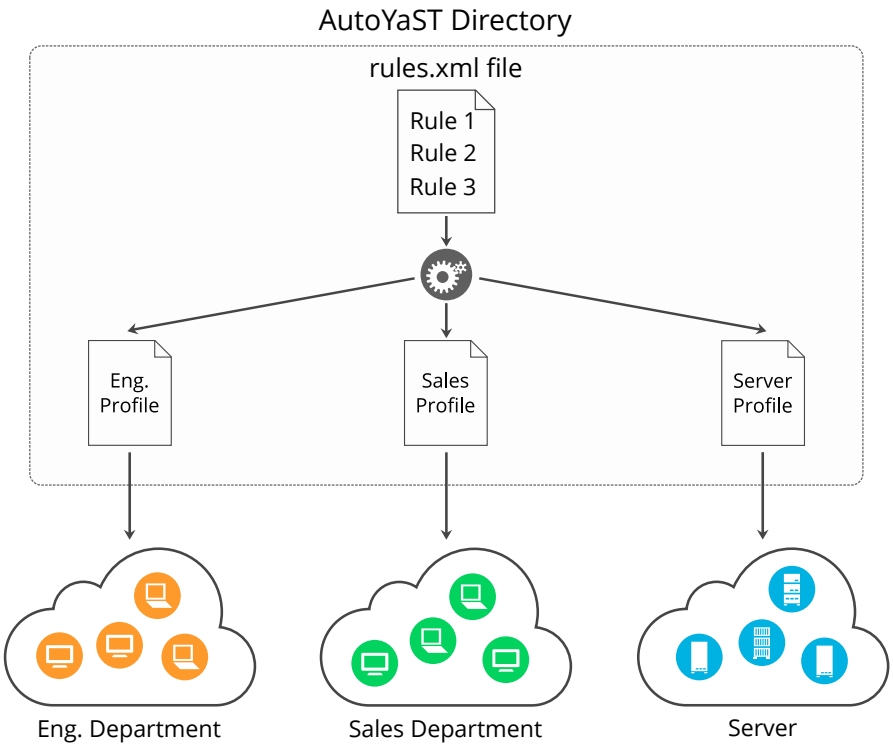

FIGURE 5.1: RULES

If more than one rule applies, the final control file for each group is generated on the fly using a merge script. The merging process is based on the order of the rules and later rules override configuration data in earlier rules. Note that the names of the top sections in the merged xml files need to be in alphabetical order for the merge to succeed.

The use of a rules file is optional. If the rules file is not found, system installation proceeds in the standard way by using the supplied control file or by searching for the control file depending on the MAC or the IP address of the system.

5.1.1 Rules File Explained

EXAMPLE 5.1: SIMPLE RULES FILE

The following simple example illustrates how the rules file is used to retrieve the configuration for a client with known hardware.

```
<?xml version="1.0"?>
<!DOCTYPE autoinstall>
<autoinstall xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://
www.suse.com/1.0/configns">
```

```
  <rules config:type="list">
    <rule>
       <disksize>
            <match>/dev/sdc 1000</match>
            <match_type>greater</match_type>
       </disksize>
       <result>
            <profile>department_a.xml</profile>
            <continue config:type="boolean">false</continue>
        </result>
    </rule>
    <rule>
       <disksize>
            <match>/dev/sda 1000</match>
            <match_type>greater</match_type>
       </disksize>
       <result>
            <profile>department_b.xml</profile>
            <continue config:type="boolean">false</continue>
        </result>
    </rule>
  </rules>
</autoinstall>
```

The last example defines two rules and provides a different control file for every rule. The rule used in this case is `disksize`. After parsing the rules file, YaST attempts to match the target system with the rules in the `rules.xml` file. A rule match occurs when the target system matches all system attributes defined in the rule. When the system matches a rule, the respective resource is added to the stack of control files AutoYaST will use to create the final control file. The `continue` property tells AutoYaST whether it should continue with other rules after a match has been found.

If the first rule does not match, the next rule in the list is examined until a match is found.

Using the `disksize` attribute, you can provide different configurations for systems with hard disks of different sizes. The first rule checks if the device `/dev/sdc` is available and if it is greater than 1 GB in size using the `match` property.

A rule must have at least one attribute to be matched. If you need to check more attributes, such as memory or architectures, you can add more attributes in the rule resource as shown in the next example.

EXAMPLE 5.2: SIMPLE RULES FILE

The following example illustrates how the rules file is used to retrieve the configuration for a client with known hardware.

```
<?xml version="1.0"?>
<!DOCTYPE autoinstall>
<autoinstall xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://
www.suse.com/1.0/configns">
  <rules config:type="list">
    <rule>
       <disksize>
            <match>/dev/sdc 1000</match>
            <match_type>greater</match_type>
       </disksize>
       <memsize>
            <match>1000</match>
            <match_type>greater</match_type>
       </memsize>
       <result>
            <profile>department_a.xml</profile>
            <continue config:type="boolean">false</continue>
        </result>
    </rule>
    <rule>
       <disksize>
            <match>/dev/shda 1000</match>
            <match_type>greater</match_type>
       </disksize>
       <memsize>
            <match>256</match>
            <match_type>greater</match_type>
       </memsize>
       <result>
            <profile>department_b.xml</profile>
            <continue config:type="boolean">false</continue>
        </result>
    </rule>
  </rules>
</autoinstall>
```

The rules directory must be located in the same directory specified via the `autoyast` keyword at boot time. If the client was booted using `autoyast=http://10.10.0.1/profiles/`, AutoYaST will search for the rules file at http://10.10.0.1/profiles/rules/rules.xml.

5.1.2 Custom Rules

If the attributes AutoYaST provides for rules are not enough for your purposes, use custom rules. Custom rules contain a shell script. The output of the script (STDOUT, STDERR is ignored) can be evaluated.

Here is an example for the use of custom rules:

```
<rule>
  <custom1>
    <script>
if grep -i intel /proc/cpuinfo > /dev/null; then
echo -n "intel"
else
echo -n "non_intel"
fi;
    </script>
    <match>*</match>
    <match_type>exact</match_type>
  </custom1>
  <result>
    <profile>@custom1@.xml</profile>
    <continue config:type="boolean">true</continue>
  </result>
</rule>
```

The script in this rule can echo either `intel` or `non_intel` to STDOUT (the output of the grep command must be directed to /dev/null in this case). The output of the rule script will be filled between the two '@' characters, to determine the file name of the control file to fetch. AutoYaST will read the output and fetch a file with the name `intel.xml` or `non_intel.xml`. This file can contain the AutoYaST profile part for the software selection; for example, in case you want a different software selection on intel hardware than on others.

The number of custom rules is limited to five. So you can use `custom1` to `custom5`.

5.1.3 Match Types for Rules

You can use five different match_types:

- `exact` (default)
- `greater`
- `lower`

- `range`
- `regex` (a simple `=~` operator like in Bash)

If using `exact`, the string must match exactly as specified. `regex` can be used to match substrings like `ntel` will match Intel, intel and intelligent. `greater` and `lower` can be used for `memsize` or `totaldisk` for example. They can match only with rules that return an integer value. A range is only possible for integer values too and has the form of `value1-value2`, for example `512-1024`.

5.1.4 Combine Attributes

Multiple attributes can be combined via a logical operator. It is possible to let a rule match if `disksize` is greater than 1GB or `memsize` is exactly 512MB.

You can do this with the `operator` element in the rules.xml file. `and` and `or` are possible operators, `and` being the default. Here is an example:

```
<rule>
  <disksize>
    <match>/dev/sda 1000</match>
    <match_type>greater</match_type>
  </disksize>
  <memsize>
    <match>256</match>
    <match_type>greater</match_type>
  </memsize>
  <result>
    <profile>machine2.xml</profile>
    <continue config:type="boolean">false</continue>
  </result>
  <operator>or</operator>
</rule>
```

5.1.5 Rules File Structure

The `rules.xml` file needs to:

- have at least one rule,
- have the name `rules.xml`,

- be located in the directory `rules` in the profile repository,
- have at least one attribute to match in the rule.

5.1.6 Predefined System Attributes

The following table lists the predefined system attributes you can match in the rules file.

If you are unsure about a value on your system, run **`/usr/lib/YaST/bin/y2base ayast_probe ncurses`**. The text box displaying the detected values can be scrolled. Note that this command will not work while another YaST process that requires a lock (for example the installer) is running. Therefore you cannot run it during the installation.

TABLE 5.1: SYSTEM ATTRIBUTES

Attribute	Values	Description
`hostaddress`	IP address of the host	This attribute must always match exactly.
`host name`	The name of the host	This attribute must always match exactly.
`domain`	Domain name of host	This attribute must always match exactly.
`installed_product`	The name of the product to be installed.	This attribute must always match exactly.
`installed_product_version`	The version of the product to be installed.	This attribute must always match exactly.
`network`	network address of host	This attribute must always match exactly.
`mac`	MAC address of host	This attribute must always match exactly (the MAC addresses should have the form `0080c8f6484c`).

Attribute	Values	Description
`linux`	Number of installed Linux partitions on the system	This attribute can be 0 or more.
`others`	Number of installed non-Linux partitions on the system	This attribute can be 0 or more.
`xserver`	X Server needed for graphic adapter	This attribute must always match exactly.
`memsize`	Memory available on host in MBytes	All match types are available.
`totaldisk`	Total disk space available on host in MBytes	All match types are available.
`hostid`	Hex representation of the IP address	Exact match required
`arch`	Architecture of host	Exact match required
`karch`	Kernel Architecture of host (for example SMP kernel, Xen kernel)	Exact match required
`disksize`	Drive device and size	All match types are available.
`product`	The hardware product name as specified in SMBIOS	Exact match required
`product_vendor`	The hardware vendor as specified in SMBIOS	Exact match required
`board`	The system board name as specified in SMBIOS	Exact match required
`board_vendor`	The system board vendor as specified in SMBIOS	Exact match required

Attribute	Values	Description
`custom1-5`	Custom rules using shell scripts	All match types are available.

5.1.7 Rules with Dialogs

You can use dialog pop-ups with check boxes to select rules you want matched.

The elements listed below must be placed within the following XML structure in the `rules.xml` file:

```
<rules config:type="list">
  <rule>
    <dialog>
      ...
    </dialog>
  </rule>
</rules>
```

Attribute	Values	Description
`dialog_nr`	All rules with the same `dialog_nr` are presented in the same pop-up dialog. The same `dialog_nr` can appear in multiple rules. `<dialog_nr config:type="integer">3</dialog_nr>`	This element is optional and the default for a missing dialog_nr is always `0`. To use one pop-up for all rules, you do not need to specify the `dialog_nr`.
`element`	Specify a unique ID. Even if you have more than one dialog, you must not use the same id twice. Using id `1` on dialog 1 and id `1` on dialog 2 is not supported. (This behavior is contrary to the `ask`	Optional. If left out, AutoYaST adds its own ids internally. Then you cannot specify conflicting rules (see below).

Attribute	Values	Description
	dialog, where you can have the same ID for multiple dialogs.) `<element config:type="integer">3</element>`	
`title`	Caption of the pop-up dialog `<title>Desktop Selection</title>`	Optional
`question`	Question shown in the pop-up behind the check box. `<question>GNOME Desktop</question>`	Optional. If you do not configure a text here, the name of the XML file that is triggered by this rule will be shown instead.
`timeout`	Timeout in seconds after which the dialog will automatically “press” the okay button. Useful for a non-blocking installation in combination with rules dialogs. `<timeout config:type="integer">30</timeout>`	Optional. A missing timeout will stop the installation process until the dialog is confirmed by the user.
`conflicts`	A list of element ids (rules) that conflict with this rule. If this rule matches or is selected by the user, all conflicting rules are deselected and	`optional`

Attribute	Values	Description
	disabled in the pop-up. Take care that you do not create deadlocks. `<conflicts config:type="list"> <element config:type="integer">1</element> <element config:type="integer">5</element> ... </conflicts>`	

Here is an example of how to use dialogs with rules:

```
<rules config:type="list">
  <rule>
    <custom1>
      <script>
echo -n 100
      </script>
      <match>100</match>
      <match_type>exact</match_type>
    </custom1>
    <result>
      <profile>rules/gnome.xml</profile>
      <continue config:type="boolean">true</continue>
    </result>
    <dialog>
      <element config:type="integer">0</element>
      <question>GNOME Desktop</question>
      <title>Desktop Selection</title>
      <conflicts config:type="list">
        <element config:type="integer">1</element>
      </conflicts>
      <dialog_nr config:type="integer">0</dialog_nr>
    </dialog>
  </rule>
  <rule>
    <custom1>
      <script>
echo -n 100
```

```
      </script>
      <match>101</match>
      <match_type>exact</match_type>
    </custom1>
    <result>
      <profile>rules/gnome.xml</profile>
      <continue config:type="boolean">true</continue>
    </result>
    <dialog>
      <element config:type="integer">1</element>
      <dialog_nr config:type="integer">0</dialog_nr>
      <question>Gnome Desktop</question>
      <conflicts config:type="list">
        <element config:type="integer">0</element>
      </conflicts>
    </dialog>
  </rule>
  <rule>
    <custom1>
      <script>
echo -n 100
      </script>
      <match>100</match>
      <match_type>exact</match_type>
    </custom1>
    <result>
      <profile>rules/all_the_rest.xml</profile>
      <continue config:type="boolean">false</continue>
    </result>
  </rule>
</rules>
```

5.2 Classes

Classes represent configurations for groups of target systems. Unlike rules, classes need to be configured in the control file. Then classes can be assigned to target systems.

Here is an example of a class definition:

```
<classes config:type="list">
  <class>
    <class_name>TrainingRoom</class_name>
    <configuration>Software.xml</configuration>
  </class>
</classes>
```

In the example above, the file `Software.xml` must be placed in the subdirectory `classes/TrainingRoom/` It will be fetched from the same place the AutoYaST control file and rules were fetched from.

If you have multiple control files and those control files share parts, better use classes for common parts. You can also use XIncludes.

Using the configuration management system, you can define a set of classes. A class definition consists of the following variables:

- Name: class name
- Description:
- Order: order (or priority) of the class in the stack of migration

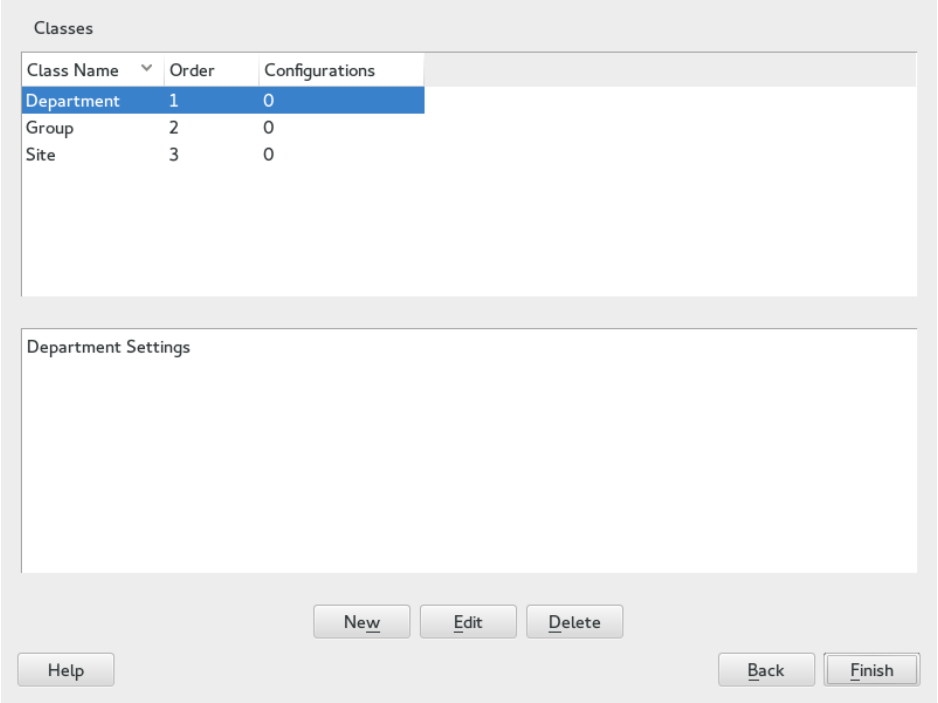

FIGURE 5.2: DEFINING CLASSES

You can create as many classes as you need, however it is recommended to keep the set of classes as small as possible to keep the configuration system concise. For example, the following sets of classes can be used:

- site: classes describing a physical location or site,
- machine: classes describing a type of machine,

- role: classes describing the function of the machine,
- group: classes describing a department or a group within a site or a location.

A file saved in a class directory can have the same syntax and format as a regular control file but represents a subset of the configuration. For example, to create a new control file for a computer with a specific network interface, you only need the control file resource that controls the configuration of the network. Having multiple network types, you can merge the one needed for a special type of hardware with other class files and create a new control file which suits the system being installed.

5.3 Mixing Rules and Classes

It is possible to mix rules and classes during an auto-installation session. For example you can identify a system using rules which contain class definitions in them. The process is described in the figure *Figure A.1, "Rules Retrieval Process"*.

After retrieving the rules and merging them, the generated control file is parsed and checked for class definitions. If classes are defined, then the class files are retrieved from the original repository and a new merge process is initiated.

5.4 Merging of Rules and Classes

With classes and with rules, multiple XML files get merged into one resulting XML file. This merging process is often confusing for people, because it behaves different than one would expect. First of all, it is important to note that the names of the top sections in the merged XML files must be in alphabetical order for the merge to succeed.

For example, the following two XML parts should be merged:

```
<partitioning config:type="list">
  <drive>
    <partitions config:type="list">
      <partition>
        <filesystem config:type="symbol">swap</filesystem>
        <format config:type="boolean">true</format>
        <mount>swap</mount>
        <partition_id config:type="integer">130</partition_id>
        <size>2000mb</size>
      </partition>
```

```
      <partition>
        <filesystem config:type="symbol">xfs</filesystem>
        <partition_type>primary</partition_type>
        <size>4Gb</size>
        <mount>/data</mount>
      </partition>
    </partitions>
  </drive>
</partitioning>
```

```
<partitioning config:type="list">
  <drive>
    <initialize config:type="boolean">false</initialize>
    <partitions config:type="list">
      <partition>
        <format config:type="boolean">true</format>
        <filesystem config:type="symbol">xfs</filesystem>
        <mount>/</mount>
        <partition_id config:type="integer">131</partition_id>
        <partition_type>primary</partition_type>
        <size>max</size>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
</partitioning>
```

You might expect the control file to contain 3 partitions. This is not the case. You will end up with two partitions and the first partition is a mix up of the swap and the root partition. Settings configured in both partitions, like `mount` or `size`, will be used from the second file. Settings that only exist in the first or second partition, will be copied to the merged partition too.

In this example, you do not want a second `drive`. The two drives should be merged into one. With regard to partitions, three separate ones should be defined. Using the `dont_merge` method solves the merging problem:

```
<classes config:type="list">
  <class>
    <class_name>swap</class_name>
    <configuration>largeswap.xml</configuration>
    <dont_merge config:type="list">
      <element>partition</element>
    </dont_merge>
  </class>
</classes>
```

```
<rule>
  <board_vendor>
    <match>ntel</match>
    <match_type>regex</match_type>
  </board_vendor>
  <result>
    <profile>classes/largeswap.xml</profile>
    <continue config:type="boolean">true</continue>
    <dont_merge config:type="list">
      <element>partition</element>
    </dont_merge>
  </result>
  <board_vendor>
    <match>PowerEdge [12]850</match>
    <match_type>regex</match_type>
  </board_vendor>
  <result>
    <profile>classes/smallswap.xml</profile>
    <continue config:type="boolean">true</continue>
    <dont_merge config:type="list">
      <element>partition</element>
    </dont_merge>
  </result>
</rule>
```

6 The Auto-Installation Process

6.1 Introduction

After the system has booted into an automatic installation and the control file has been retrieved, YaST configures the system according to the information provided in the control file. All configuration settings are summarized in a window that is shown by default and should be deactivated if a fully automatic installation is needed.

By the time YaST displays the summary of the configuration, YaST has only probed hardware and prepared the system for auto-installation. Nothing has been changed in the system yet. In case of any error, you can still abort the process.

A system should be automatically installable without the need to have any graphic adapter or monitor. Having a monitor attached to the client machine is nevertheless recommended so you can supervise the process and to get feedback in case of errors. Choose between the graphical and the text-based Ncurses interfaces. For headless clients, system messages can be monitored using the serial console.

6.1.1 X11 Interface (graphical)

This is the default interface while auto-installing. No special variables are required to activate it.

6.1.2 Serial console

Start installing a system using the serial console by adding the keyword `console` (for example `console=ttyS0`) to the command line of the kernel. This starts linuxrc in console mode and later YaST in serial console mode.

6.1.3 Text-based YaST Installation

This option can also be activated on the command line. To start YaST in text mode, add `textmode=1` on the command line.

Starting YaST in the text mode is recommended when installing a client with less than 64 MB or when X11 should not be configured, especially on headless machines.

6.2 Choosing the Right Boot Medium

There are different methods for booting the client. The computer can boot from its network interface card (NIC) to receive the boot images via DHCP or TFTP. Alternatively a suitable kernel and initrd image can be loaded from a flash disk or a bootable DVD-ROM.

YaST will check for `autoinst.xml` in the root directory of the boot medium or the initrd upon start-up and switch to an automated installation if it was found. In case the control file is named differently or located elsewhere, specify its location on the kernel command line with the parameter `AutoYaST=`*`URL`*.

6.2.1 Booting from a Flash Disk

For testing/rescue purposes or because the NIC does not have a PROM or PXE you can build a bootable flash disk to use with AutoYaST. Flash disks can also store the control file.

Tip: Creating a Bootable Flash Disk

To create a bootable flash disk, copy either the SUSE Linux Enterprise Server ISO image of DVD1 or the Mini CD ISO image to the disk using the **dd** command (the flash disk must not be mounted, all data on the device will be erased):

```
dd if=PATH_TO_ISO_IMAGE of=USB_STORAGE_DEVICE bs=4M
```

6.2.2 Booting from DVD-ROM

You can use the original SUSE Linux Enterprise Server DVD-ROM number one in combination with other media. For example, the control file can be provided via a flash disk or a specified location on the network. Alternatively, create a customized DVD-ROM that includes the control file.

6.2.3 Booting via PXE over the Network

Booting via PXE requires a DHCP and a TFTP server in your network. The computer will then boot without a physical medium. For instructions on setting up the required infrastructure, see *Book “*Deployment Guide*”, Chapter 10 “Remote Installation”*.

If you do installation via PXE, the installation will run into an endless loop. This happens because after the first reboot, the machine performs the PXE boot again and restarts the installation instead of booting from the hard disk for the second stage of the installation.

There are several ways to solve this problem. You can use an HTTP server to provide the AutoYaST control file. Alternatively, instead of a static control file, run a CGI script on the Web server that provides the control file and changes the TFTP server configuration for your target host. This way, the next PXE boot of the machine will be from the hard disk by default.

Another way is to use AutoYaST to upload a new PXE boot configuration for the target host via the control file:

```
<pxe>
  <pxe_localboot config:type="boolean">true</pxe_localboot>
  <pxelinux-config>
    DEFAULT linux
    LABEL linux
    localboot 0
  </pxelinux-config>
  <tftp-server>192.168.1.115</tftp-server>
  <pxelinux-dir>/pxelinux.cfg</pxelinux-dir>
  <filename>__MAC__</filename>
</pxe>
```

This entry will upload a new configuration for the target host to the TFTP server shortly before the first reboot happens. In most installations the TFTP daemon runs as user `nobody`. You need to make sure this user has write permissions to the `pxelinux.cfg` directory. You can also configure the file name that will be uploaded. If you use the "magic" `__MAC__` file name, the file name will be the MAC address of your machine like, for example `01-08-00-27-79-49-ee`. If the file name setting is missing, the IP address will be used for the file name.

To do another auto-installation on the same machine, you need to remove the file from the TFTP server.

6.3 Invoking the Auto-Installation Process

6.3.1 Command Line Options

Adding the command line variable `autoyast` causes linuxrc to start in automated mode. linuxrc searches for a configuration file, which should be distinguished from the main control file in the following places:

- in the root directory of the initial RAM disk used for booting the system,
- in the root directory of the boot medium

The configuration file used by linuxrc can have the following keywords (for a detailed description of how linuxrc works and other keywords, see *Appendix C, Advanced Linuxrc Options*):

TABLE 6.1: KEYWORDS FOR LINUXRC

Keyword	Value
`autoupgrade`	Initiate an automatic upgrade using AutoYaST, see *Section 4.10, "Upgrade"*. For some use cases, you need the `autoyast` parameter (see *AutoYaST Control File Locations* for details).
`autoyast`	Location of the control file for automatic installation, see *AutoYaST Control File Locations* for details.
`autoyast2`	Location of the control file for automatic installation. Similar to `autoyast` option but linuxrc parses the provided value and, for example, tries to configure a network when needed. For information about differences between the AutoYaST and linuxrc URI syntax, see the documentation of linuxrc. AutoYaST's rules and classes are not supported.
`ifcfg`	Configure and start the network. Required if the AutoYaST is to be fetched from a remote location. See *Section C.3, "Advanced Network Setup"* for details.

Keyword	Value
`insmod`	Kernel modules to load
`install`	Location of the installation directory, for example `install=nfs://192.168.2.1/CDs/`.
`instmode`	Installation mode, for example `nfs`, `http` etc. (not needed if `install` is set).
`server`	Server (NFS) to contact for source directory
`serverdir`	Directory on NFS Server
`y2confirm`	Even with <confirm>no</confirm> in the control file, the confirm proposal comes up.

These variables and keywords will bring the system up to the point where YaST can take over with the main control file. Currently, the source medium is automatically discovered, which in some cases makes it possible to initiate the auto-install process without giving any instructions to linuxrc.

The traditional linuxrc configuration file (`info`) has the function of giving the client enough information about the installation server and the location of the sources. Usually, this file is not required, but it is needed in special network environments where DHCP and BOOTP are not used or when special kernel modules need to be loaded.

All linuxrc keywords can be passed to linuxrc using the kernel command line. The command line can also be set when creating network bootable images or it can be passed to the kernel using a specially configured DHCP server in combination with Etherboot or PXE.

The command line variable `autoyast` can be used in the format described in the following list.

AUTOYAST CONTROL FILE LOCATIONS

Format of URIs

The `autoyast` syntax for the URIs for your control file locations can be confusing. The format is SCHEMA://HOST/PATH-TO-FILE. The number of forward slashes to use varies. For remote locations of your control file, the URI looks like this example for an NFS server, with two slashes: `autoyast=nfs://SERVER/PATH`.

It is different when your control file is on a local filesystem. For example, `autoyast=usb:///profile.xml` is the same as `autoyast=usb://localhost/profile.xml`. You may omit the local host name, but you must keep the third slash. `autoyast=usb://profile.xml` will fail because `profile.xml` is interpreted as the host name.

When no control file specification is needed

For upgrades, no `autoyast` variable is needed for an automated offline upgrade.
For new installations, `autoyast` will be started if a file named `autoinst.xml` is in one of the following three locations:

1. The root directory of the installation flash disk (e.g. USB stick)
2. The root directory of the installation medium
3. Or the root directory of the initial RAM disk used to boot the system

`autoyast=file:///PATH`

Looks for control file in the specified path, relative to the source root directory, for example `file:///autoinst.xml` when the control file is in the top-level directory of any local filesystem, including mounted external devices such as a CD or USB drive. (This is the same as `file://localhost/autoinst.xml`.)

`autoyast=device://DEVICE/FILENAME`

Looks for the control file on a storage device. Do not specify the full path to the device, but the device name only (e.g. `device://vda1/autoyast.xml`). You may also omit specifying the device and trigger `autoyast` to search all devices, for example. `autoyast=device://localhost/autoinst.xml`, or `autoyast=device:///autoinst.xml`.

`autoyast=nfs://SERVER/PATH`

Looks for the control file on an NFS server.

`autoyast=http://[user:password@]SERVER/PATH`

Retrieves the control file from a Web server using the HTTP protocol. Specifying a user name and a password is optional.

`autoyast=https://[user:password@]SERVER/PATH`

Retrieves the control file from a Web server using HTTPS. Specifying a user name and a password is optional.

`autoyast=tftp://SERVER/PATH`

Retrieve the control file via TFTP.

`autoyast=ftp://[user:password@]SERVER/PATH`

Retrieve the control file via FTP. Specifying a user name and a password is optional.

`autoyast=usb:///PATH`

Retrieve the control file from USB devices (`autoyast` will search all connected USB devices).

`autoyast=relurl://PATH`

Retrieve the control file from the installation source. Either from the default installation source or from the installation source defined in `install=INSTALLATION_SOURCE_PATH`.

`autoyast=cifs://SERVER/PATH`

Looks for the control file on a CIFS server.

`autoyast=label://LABEL/PATH`

Searches for a control file on a device with the specified label.

Several scenarios for auto-installation are possible using different types of infrastructure and source media. The simplest way is to use the source media (DVD number one) of SUSE Linux Enterprise Server. But to initiate the auto-installation process, the auto-installation command-line variable should be entered at system boot-up and the control file should be accessible for YaST.

In a scripting context, you can use a serial console for your virtual machine, that allows you to work in text mode. Then you can pass the needed parameters from an expect script or equivalent.

The following list of scenarios explains how the control file can be supplied:

Using the Original SUSE Linux Enterprise Server DVD-ROM

When using the original DVD-ROM (DVD #1 is needed), the control file needs to be accessible via flash disk or network:

Flash Disk. Access the control file via the `autoyast=usb://PATH` option.

Network. Access the control file via the following commands: `autoyast=nfs://..`, `autoyast=ftp://..`, `autoyast=http://..`, `autoyast=https://..`, `autoyast=tftp://..`, or `autoyast=cifs://..`.

Using a Custom DVD-ROM

In this case, you can include the control file directly on the DVD-ROM. When placing it in the root directory and naming it `autoinst.xml`, it will automatically be found and used for the installation. Otherwise use `autoyast=file:///PATH` to specify the path to the control file.

When using a DVD-ROM for auto-installation, it is necessary to instruct the installer to use the DVD-ROM for installation instead of trying to find the installation files on the network. This can be done by adding the `instmode=cd` option to the kernel command line (this can be automated by adding the option to the `isolinux.cfg` file on the DVD).

Using a Network Installation Source

This option is the most important one because installations of multiple machines are usually done using SLP or NFS servers and other network services like BOOTP and DHCP. The easiest way to make the control file available is to place it in the root directory of the installation source naming it `autoinst.xml`. In this case it will automatically be found and used for the installation. The control file can also reside in the following places:

Flash Disk. Access the control file via the `autoyast=usb://`*`PATH`* option.

Network. Access the control file via the following commands: `autoyast=nfs://..`, `autoyast=ftp://..`, `autoyast=http://..`, `autoyast=https://..`, `autoyast=tftp://..`, or `autoyast=cifs://..`.

Note: Disabling Network and DHCP

To disable the network during installations where it is not needed or unavailable, for example when auto-installing from DVD-ROMs, use the linuxrc option `netsetup=0` to disable the network setup.

Note: Difference between the `autoyast` and `autoyast2` Options

The options `autoyast` and `autoyast2` are very similar but differ in one important point:

- When you use `autoyast=http://...`, you need to provide linuxrc with the network configuration.
- When you use `autoyast2=http://...`, linuxrc tries to configure the network for you.

If `autoyast=default` is defined, YaST will look for a file named `autoinst.xml` in the following three places:

1. the root directory of the flash disk,
2. the root directory of the installation medium,
3. the root directory of the initial RAM disk used to boot the system.

With all AutoYaST invocation options, excluding `default`, it is possible to specify the location of the control file in the following ways:

1. Specify the exact location of the control file:

   ```
   autoyast=http://192.168.1.1/control-files/client01.xml
   ```

2. Specify a directory where several control files are located:

   ```
   autoyast=http://192.168.1.1/control-files/
   ```

   In this case the relevant control file is retrieved using the hex digit representation of the IP as described below.

If only the path prefix variable is defined, YaST will fetch the control file from the specified location in the following way:

1. First, it will search for the control file using its own IP address in uppercase hexadecimal, for example `192.0.2.91 -> C000025B`.
2. If this file is not found, YaST will remove one hex digit and try again. This action is repeated until the file with the correct name is found. Ultimately, it will try looking for a file with the MAC address of the client as the file name (mac should have the following syntax: `0080C8F6484C`) and if not found a file named `default` (in lowercase).

As an example, for 192.0.2.91, the HTTP client will try:

```
C000025B
C000025
C00002
C0000
C000
C00
C0
```

```
C
0080C8F6484C
default
```

in that order.

To determine the hex representation of the IP address of the client, use the utility called `/usr/bin/gethostip` available with the `syslinux` package.

EXAMPLE 6.1: DETERMINE HEX CODE FOR AN IP ADDRESS

```
# /usr/bin/gethostip 10.10.0.1
10.10.0.1 10.10.0.1 0A0A0001
```

6.3.2 Auto-installing a Single System

The easiest way to auto-install a system without any network connection is to use the original SUSE Linux Enterprise Server DVD-ROMs and a flash disk. You do not need to set up an installation server nor the network environment.

Create the control file and name it `autoinst.xml`. Copy the file `autoinst.xml` to the flash disk.

6.3.3 Combining the linuxrc `info` file with the AutoYaST control file

If you choose to pass information to linuxrc using the `info` file, it is possible to integrate the keywords in the XML control file. In this case the file needs to be accessible to linuxrc and needs to be named `info`.

Linuxrc will look for a string (`start_linuxrc_conf` in the control file which represents the beginning of the file. If it is found, it will parse the content starting from that string and will finish when the string `end_linuxrc_conf` is found. The options are stored in the control file in the following way:

EXAMPLE 6.2: LINUXRC OPTIONS IN THE CONTROL FILE

```
....
  <install>
....
    <init>
      <info_file>
<![CDATA[
```

```
#
# Do not remove the following line:
# start_linuxrc_conf
#
install: nfs://192.168.1.1/CDs/full-i386
textmode: 1
autoyast: file:///info

# end_linuxrc_conf
# Do not remove the above comment
#
]]>

      </info_file>
    </init>
......
  </install>
....
```

Note that the `autoyast` keyword must point to the same file. If it is on a flash disk, then the option `usb:///` needs to be used. If the `info` file is stored in the initial RAM disk, the `file://` option needs to be used.

6.4 System Configuration

The system configuration during auto-installation is the most important part of the whole process. As you have seen in the previous chapters, almost anything can be configured automatically on the target system. In addition to the pre-defined directives, you can always use post-scripts to change other things in the system. Additionally you can change any system variables, and if required, copy complete configuration files into the target system.

6.4.1 Post-Install and System Configuration

The post-installation and system configuration are initiated directly after the last package is installed on the target system and continue after the system has booted for the first time.

Before the system is booted for the first time, AutoYaST writes all data collected during installation and writes the boot loader in the specified location. In addition to these regular tasks, AutoYaST executes the chroot-scripts as specified in the control file. Note that these scripts are executed while the system is not yet mounted.

If a different kernel than the default is installed, a hard reboot will be required. A hard reboot can also be forced during auto-installation, independent of the installed kernel. Use the `reboot` property of the `general` resource (see *Section 4.1, "General Options"*).

6.4.2 System Customization

Most of the system customization is done in the second stage of the installation. If you require customization that cannot be done using AutoYaST resources, use post-install scripts for further modifications.

You can define an unlimited number of custom scripts in the control file, either by editing the control file or by using the configuration system.

7 Running AutoYaST in an Installed System

In some cases it is useful to run AutoYaST in a running system.

In the following example, an additional software package (`foo`) is going to be installed. To run this software, a user needs to be added and an NTP client needs to be configured.

The respective AutoYaST profile needs to include a section for the package installation (*Section 4.9.6, "Installing Packages in Stage 2"*), a user (*Section 4.29.1, "Users"*) section and an NTP-client (*Section 4.20, "NTP Client"*) section:

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://www.suse.com/1.0/
configns">
  <ntp-client>
    <peers config:type="list">
      <peer>
        <address>us.pool.ntp.org</address>
        <comment/>
        <options> iburst</options>
        <type>server</type>
      </peer>
    </peers>
    <start_at_boot config:type="boolean">true</start_at_boot>
    <start_in_chroot config:type="boolean">false</start_in_chroot>
    <sync_interval config:type="integer">5</sync_interval>
    <synchronize_time config:type="boolean">false</synchronize_time>
  </ntp-client>
  <software>
    <post-packages config:type="list">
      <package>ntp</package>
      <package>yast2-ntp-client</package>
      <package>foo</package>
    </post-packages>
  </software>
  <users config:type="list">
    <user>
      <encrypted config:type="boolean">false</encrypted>
      <fullname>Foo user</fullname>
      <gid>100</gid>
      <home>/home/foo</home>
      <password_settings>
        <expire/>
        <flag/>
        <inact/>
```

```
        <max>99999</max>
        <min>0</min>
        <warn>7</warn>
      </password_settings>
      <shell>/bin/bash</shell>
      <uid>1001</uid>
      <user_password>linux</user_password>
      <username>foo</username>
    </user>
  </users>
</profile>
```

Store this file as `/tmp/install_foo.xml` and start the AutoYaST installation process by calling:

```
yast2 ayast_setup setup filename=/tmp/install_foo.xml dopackages="yes"
```

For more information, run **`yast2 ayast_setup longhelp`**

A Handling Rules

The following figure illustrates how rules are handled and the processes of retrieval and merge.

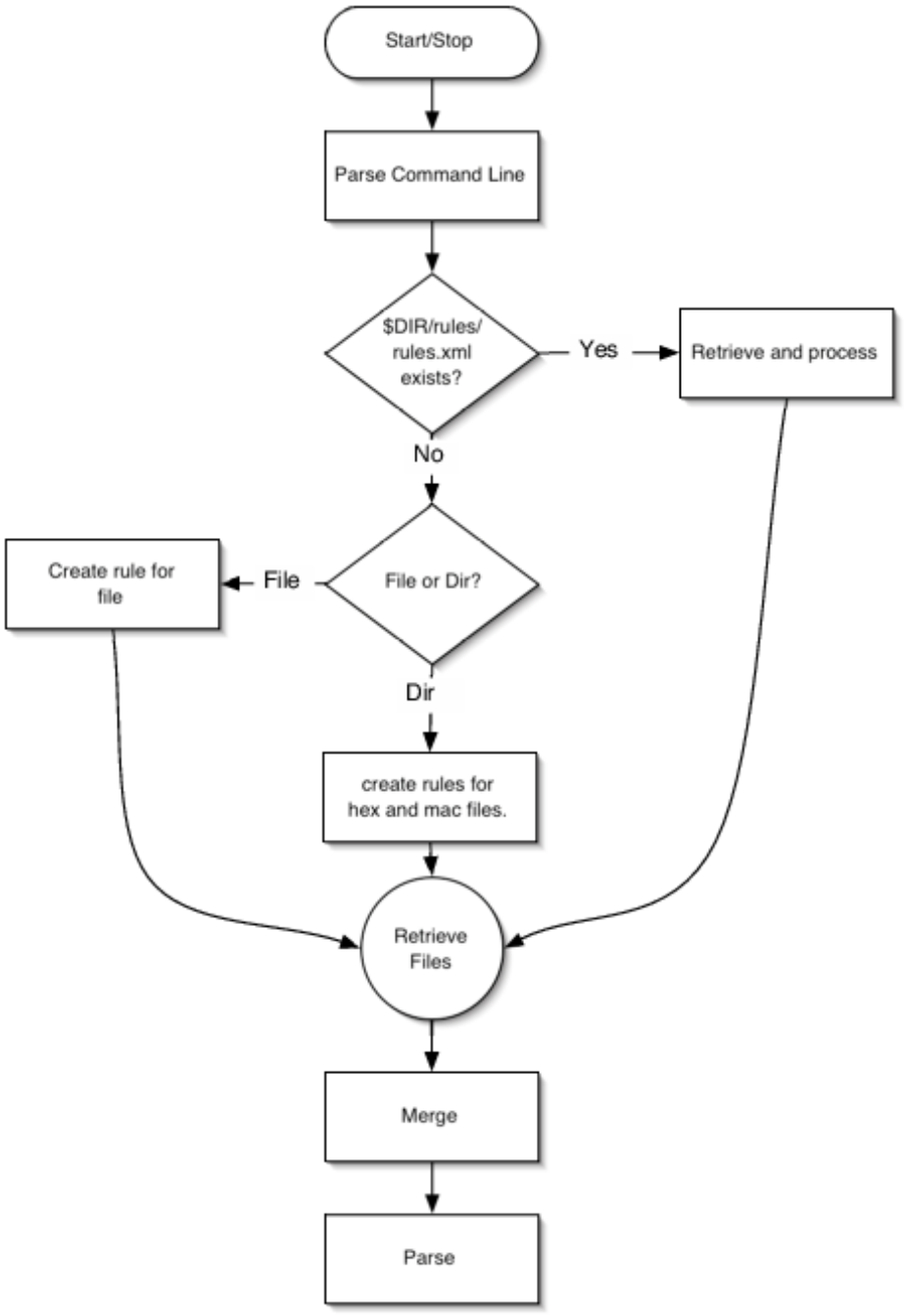

FIGURE A.1: RULES RETRIEVAL PROCESS

B AutoYaST FAQ - Frequently Asked Questions

B.1. *How do I invoke an AutoYaST installation?*

On all SUSE Linux Enterprise Server versions, the automatic installation gets invoked by adding `autoyast=<PATH_TO_PROFILE>` to the kernel parameter list. So for example adding `autoyast=http://MYSERVER/MYCONFIG.xml` will start an automatic installation where the profile with the AutoYaST configuration gets fetched from the Web server `myserver`. See *Section 6.3, "Invoking the Auto-Installation Process"* for more information.

B.2. *What is an AutoYaST profile?*

A profile is the AutoYaST configuration file. The content of the AutoYaST profile determines how the system will be configured and which packages will get installed. This includes partitioning, network setup, and software sources, to name but a few. Almost everything that can be configured with YaST in a running system can also be configured in an AutoYaST profile. The profile format is an ASCII XML file.

B.3. *How do I create an AutoYaST profile?*

The easiest way to create an AutoYaST profile is to use an existing SUSE Linux Enterprise Server system as a template. On an already installed system, start *YaST* › *Miscellaneous* › *Autoinstallation*. Now select *Tools* › *Create Reference Profile* from the menu. Choose the system components you want to include in the profile. Alternatively, create a profile containing the complete system configuration by running **`sudo yast clone_system`** from the command line.

Both methods will create the file `/root/autoinst.xml`. The version created on the command line can be used to set up an identical clone of the system on which the profile was created. However, usually you will want to adjust the file to make it possible to install several machines that are very similar, but not identical. This can be done by adjusting the profile using your favorite text/XML editor.

B.4. *How can I check the syntax of a created AutoYaST profile?*

The most efficient way to check your created AutoYaST profile is by using **`jing`** or **`xmllint`**.

See *Section 3.3, "Creating/Editing a Control File Manually"* for details.

B.5. *What is smallest AutoYaST profile that makes sense?*

If a section has not been defined in the AutoYaST profile the settings of the general YaST installation proposal will be used. However, you need to specify at least the `root` password to be able to log in to the machine after the installation.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://
www.suse.com/1.0/configns">
  <users config:type="list">
    <user>
      <encrypted config:type="boolean">false</encrypted>
      <user_password>linux</user_password>
      <username>root</username>
    </user>
  </users>
</profile>
```

B.6. *How do I do an automatic installation with autodetection of my sound card?*

Use the following `sound` section in your profile:

```
<sound>
  <autoinstall config:type="boolean">true</autoinstall>
  <configure_detected config:type="boolean">true</configure_detected>
</sound>
```

B.7. *I want to install from DVD only. Where do I put the AutoYaST profile?*

Put the profile in the root of the DVD. Refer to it with `file:///PROFILE.xml`.

B.8. *How can I test a merging process on the command line?*

To merge two profiles, `a.xml` with `base.xml`, run the following command:

```
/usr/bin/xsltproc --novalid --param replace "'false'" \
--param dontmerge1 "'package'" --param with "'a.xml'" --output out.xml \
/usr/share/autoinstall/xslt/merge.xslt base.xml
```

This requires sections in both profiles to be in alphabetical order (<software>, for example, needs to be listed after <add-on>). If you have created the profile with YaST, profiles are automatically sorted correctly.

The `dontmerge1` parameter is optional and an example of what to do when you use the `dont_merge` element in your profile. See *Section 5.4, "Merging of Rules and Classes"* for more information.

B.9. *May I call Zypper from scripts?*

Zypper can only be called from AutoYaST init scripts, because during the post-script phase, YaST still has an exclusive lock on the RPM database.

If you really need to use other script types (for example a post-script) you will need to break the lock at your own risk:

```
<post-scripts config:type="list">
  <script>
    <filename>yast_clone.sh</filename>
    <interpreter>shell</interpreter>
    <location/>
    <feedback config:type="boolean">false</feedback>
    <source><![CDATA[#!/bin/sh
mv /var/run/zypp.pid /var/run/zypp.sav
zypper in foo
mv /var/run/zypp.sav /var/run/zypp.pid
]]></source>
  </script>
</post-scripts>
```

B.10. *Is the order of sections in an AutoYaST profile important?*

Actually the order is not important. The order of sections in the profile has no influence on the AutoYaST workflow. However, if you want to *merge* different profiles, sections need to be in alphabetical order.

B.11. *linuxrc blocks the installation with* `File not signed`*. I need to manually interact.*

Linuxrc found some unsigned file (like a driver update). To use an unsigned file, you can suppress that message by passing `insecure=1` to the linuxrc parameter list (together with the `autoyast=...` parameter).

B.12. *I want to install from DVD/USB/HD but fetch the XML file from the network.*

You need to pass `ifcfg` to linuxrc. This is required to set up the network, otherwise AutoYaST cannot download the profile from remote. See *Section C.3, "Advanced Network Setup"* for more information.

B.13. *Is the installation on an NFS root (`/`) possible?*

Yes, but it is a little bit "tricky". You will need to set up the environment (DHCP, TFTP, etc.) very carefully. The AutoYaST profile needs to look like the following:

```
<?xml version="1.0"?>
<!DOCTYPE profile>
```

```
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://
www.suse.com/1.0/configns">
  <partitioning config:type="list">
    <drive>
      <device>/dev/nfs</device>
      <initialize config:type="boolean">false</initialize>
      <type config:type="symbol">CT_NFS</type>
      <partitions config:type="list">
        <partition>
          <filesystem config:type="symbol">nfs</filesystem>
          <fstopt>nolock</fstopt>
          <device>10.10.1.53:/tmp/m4</device>
          <mount>/</mount>
        </partition>
      </partitions>
      <use>all</use>
    </drive>
  </partitioning>
</profile>
```

B.14. *Where can I ask questions which have not been answered here?*

There is an AutoYaST mailing list where you can post your questions. Join us at http://lists.opensuse.org/opensuse-autoinstall/.

C Advanced Linuxrc Options

Linuxrc is a program used for setting up the kernel for installation purposes. It allows the user to load modules, start an installed system, a rescue system or an installation via YaST.

Linuxrc is designed to be as small as possible. Therefore, all needed programs are linked directly into one binary. So there is no need for shared libraries in the init disk.

Note: Running Linuxrc on an Installed System

If you run Linuxrc on an installed system, it will work slightly differently so as not to destroy your installation. As a consequence you cannot test all features this way.

C.1 Passing parameters to Linuxrc

Unless Linuxrc is in manual mode, it will look for an `info` file in these locations: first `/info` on the flash disk and if that does not exist, for `/info` in the initrd. After that it parses the kernel command line for parameters. You may change the `info` file Linuxrc reads by setting the `info` command line parameter. If you do not want Linuxrc to read the kernel command line (for example because you need to specify a kernel parameter that Linuxrc recognizes as well), use `linuxrc=nocmdline`.

Linuxrc will always look for and parse a file `/linuxrc.config`. Use this file to change default values if you need to. In general, it is better to use the `info` file instead. Note that `/linuxrc.config` is read before any `info` file, even in manual mode.

C.2 `info` file format

Lines starting with `#` are comments, valid entries are of the form:

```
key: value
```

Note that `value` extends to the end of the line and therefore may contain spaces. `key` is matched case-insensitive.

You can use the same key-value pairs on the kernel command line using the syntax `key=value`. Lines that do not have the form described above are ignored.

The table below lists important keys and example values. For a complete list of linuxrc parameters refer to https://en.opensuse.org/SDB:Linuxrc.

TABLE C.1:

TABLE C.2: ADVANCED LINUXRC KEYWORDS

Keyword: Example Value	Description
`addswap: 0|3|/dev/sda5`	If 0, never ask for swap; if the argument is a positive number `n`, activate the `n`'th swap partition; if the argument is a partition name, activate this swap partition.
`autoyast: ftp://AUTOYASTFILE`	Location of the auto installation file; activates auto installation mode. See *AutoYaST Control File Locations* for details.
`bootptimeout: 10`	10 seconds timeout for BOOTP requests.
`bootpwait: 5`	Sleep 5 seconds between network activation and starting bootp.
`display: color|mono|alt`	Set the menu color scheme.
`exec: COMMAND`	Run *command*.
`forceinsmod: 0|1`	Use the `-f` option (force) when running **`insmod`** commands.
`forcerootimage: 0|1`	Load the installation system into RAM disk.
`ifcfg: NETWORK_CONFIGURATION`	Set up and start the network. See *Section C.3, "Advanced Network Setup"* for more information.
`insmod: MODULE`	Load *`MODULE`*.
`install: URL`	Install from the repository specified with *`URL`*. For the syntax of *`URL`* refer to https://en.opensuse.org/SDB:Linuxrc#url_descr.
`keytable: de-lat1-nd`	Virtual console keyboard map to load.
`language: de_DE`	Language preselected for the installation.

Keyword: Example Value	Description
`loghost: 10.10.0.22`	Enable remote logging via syslog.
`memloadimage: 50000`	Load installation system into RAM disk if free memory is above 50000 KB.
`memlimit: 10000`	Ask for swap if free memory drops below 10000 KB.
`memYaST: 20000`	Run YaST in text mode if free memory is below 20000 KB.
`memYaSTText: 10000`	Ask for swap before starting YaST if free memory is below 10000 KB.
`proxy: 10.10.0.1`	Proxy (either FTP or HTTP).
`rescue: 1|nfs://server/dir`	Load the rescue system; the URL variant specifies the location of the rescue image explicitly.
`rescueimage: /suse/images/rescue`	Location of the rescue system image.
`rootimage: /suse/images/root`	Location of the installation system image.
`textmode: 1`	Start YaST in text mode.
`usbwait: 4`	Wait 4 seconds after loading the USB modules.
`y2confirm`	Overrides the confirm parameter in a control file and requests confirmation of installation proposal.

C.3 Advanced Network Setup

Even if parameters like `hostip`, `nameserver`, and `gateway` are passed to linuxrc, the network is only started when it is needed (for example, when installing via SSH or VNC). Since `autoyast` is not a linuxrc parameter (this parameter is ignored by linuxrc and only passed to YaST), the network will *not* be started automatically when specifying a remote location for the AutoYaST profile.

Therefore the network needs to be started explicitly. This used to be done with the linuxrc parameter `netsetup`. Starting with SUSE Linux Enterprise Server 12, the parameter `ifcfg` is available. It offers more configuration options, for example configuring more than one interface. `ifcfg` directly controls the content of the `/etc/sysconfig/network/ifcfg-*` files.

DHCP Network Configuration

The general syntax to configure DHCP is

```
ifcfg=INTERFACE=DHCP*,OPTION1=VALUE1,OPTION2=VALUE2
```

where *`INTERFACE`* is the interface name, for example `eth0`, or `eth*` for all interfaces. *`DHCP*`* can either be `dhcp` (IPv4 and IPv6), `dhcp4`, or `dhcp6`.

To set up DHCP for `eth0` use:

```
ifcfg=eth0=dhcp
```

To set up DHCP on all interfaces use:

```
ifcfg=eth*=dhcp
```

Static Network Configuration

The general syntax to configure a static network is

```
ifcfg=INTERFACE=IP_LIST,GATEWAY_LIST,NAMESERVER_LIST,DOMAINSEARCH_LIST,\
OPTION1=value1,...
```

where *`INTERFACE`* is the interface name, for example `eth0`. If using `eth*`, the first device available will be used. The other parameters need to be replaced with the respective values in the given order. Example:

```
ifcfg=eth0=192.168.2.100/24,192.168.5.1,192.168.1.116,example.com
```

When specifying multiple addresses for a parameter, use spaces to separate them and quote the complete string. The following example uses two name servers and a search list containing two domains.

```
ifcfg="eth0=192.168.2.100/24,192.168.5.1,192.168.1.116 192.168.1.117,example.com
 example.net"
```

For more information refer to https://en.opensuse.org/SDB:Linuxrc#Network_Configuration.

D GNU Licenses

This appendix contains the GNU Free Documentation License version 1.2.

GNU Free Documentation License

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. 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 non-commercially. 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 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.

1. 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.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, 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.

3. 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.

4. 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.

5. 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".

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

ADDENDUM: How to use this License for your documents

```
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.