



# AutoYaST Guide

---

openSUSE Leap 15.2



## AutoYaST Guide

openSUSE Leap 15.2

AutoYaST is a system for unattended mass deployment of openSUSE Leap systems. AutoYaST installations are performed using an AutoYaST control file (also called a “profile”) with your customized installation and configuration data.

Publication Date: December 16, 2020

SUSE LLC

1800 South Novell Place

Provo, UT 84606

USA

<https://documentation.suse.com> ↗

Copyright © 2006– 2020 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 <https://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

<b>1</b>	<b>Introduction to AutoYaST 1</b>
1.1	Motivation 1
1.2	Overview and Concept 1
I UNDERSTANDING AND CREATING THE AUTOYAST CONTROL FILE 4	
<b>2</b>	<b>The AutoYaST Control File 5</b>
2.1	Introduction 5
2.2	Format 5
2.3	Structure 6
	Resources and Properties 7 • Nested Resources 7 • Attributes 8
<b>3</b>	<b>Creating an AutoYaST Control File 9</b>
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 12
II AUTOYAST CONFIGURATION EXAMPLES 14	
<b>4</b>	<b>Configuration and Installation Options 15</b>
4.1	General Options 15
	The Mode Section 16 • Configuring the Installation Settings Screen 20 • The Self-Update Section 20 • The Semi-Automatic Section 22 • The Signature Handling Section 22 • The Wait Section 24 • Examples for the general Section 26
4.2	Reporting 28

- 4.3 The Boot Loader 29
  - Loader Type 29 • Globals 30 • Device map 35
- 4.4 Partitioning 35
  - Automatic Partitioning 36 • Guided Partitioning 36 • Expert Partitioning 37 • Advanced Partitioning Features 54 • Logical Volume Manager (LVM) 58 • Software RAID 60 • Multipath Support 66 • bcache Configuration 67 • Multi-device Btrfs Configuration 69 • NFS Configuration 71
- 4.5 iSCSI Initiator Overview 72
- 4.6 Fibre Channel over Ethernet Configuration (FCoE) 73
- 4.7 Country Settings 74
- 4.8 Software 76
  - Package Selection with Patterns and Packages Sections 76 • Deploying Images 77 • Installing Additional/Customized Packages or Products 77 • Kernel Packages 84 • Removing Automatically Selected Packages 84 • Installing Recommended Packages/Patterns 85 • Installing Packages in Stage 2 85 • Installing Patterns in Stage 2 86 • Online Update in Stage 2 86
- 4.9 Upgrade 86
- 4.10 Services and Targets 88
- 4.11 Network Configuration 89
  - Interfaces 92 • Persistent Names of Network Interfaces 96 • Domain Name System 97 • Routing 98 • s390 Options 99 • Proxy 100
- 4.12 NIS Client and Server 100
- 4.13 NIS Server 101
- 4.14 Hosts Definition 103
- 4.15 Windows Domain Membership 104
- 4.16 Samba Server 105
- 4.17 Authentication Client 106

- 4.18 NFS Client and Server 107
- 4.19 NTP Client 108
- 4.20 Mail Server Configuration 109
- 4.21 Apache HTTP Server Configuration 111
- 4.22 Squid Server 120
- 4.23 FTP Server 127
- 4.24 TFTP Server 131
- 4.25 Firstboot Workflow 132
- 4.26 Security Settings 132
  - Password Settings Options 133 • Boot Settings 134 • Login Settings 134 • New user settings (**useradd** settings) 134
- 4.27 Linux Audit Framework (LAF) 134
- 4.28 Users and Groups 137
  - Users 137 • User Defaults 142 • Groups 143 • Login Settings 144
- 4.29 Custom User Scripts 145
  - Pre-Install Scripts 145 • Post-partitioning Scripts 146 • Chroot Environment Scripts 147 • Post-Install Scripts 147 • Init Scripts 147 • Script XML Representation 149 • Script Example 152
- 4.30 System Variables (Sysconfig) 154
- 4.31 Adding Complete Configurations 155
- 4.32 Ask the User for Values during Installation 157
  - Default Value Scripts 166 • Scripts 167
- 4.33 Kernel Dumps 172
  - Memory Reservation 173 • Dump Saving 174 • E-Mail Notification 177 • Kdump Kernel Settings 178 • Expert Settings 180
- 4.34 DNS Server 180
- 4.35 DHCP Server 183

- 4.36 Firewall Configuration 186
  - General Firewall Configuration 187 • Firewall Zones Configuration 188 • A Full Example 189
- 4.37 Miscellaneous Hardware and System Components 190
  - Printer 190 • Sound devices 191
- 4.38 Importing SSH Keys and Configuration 192
- 4.39 Configuration Management 193
  - Connecting to a Configuration Management Server 193 • Running in Stand-alone Mode 195 • SUSE Manager Salt Formulas Support 196

### III MANAGING MASS INSTALLATIONS WITH RULES AND CLASSES 197

## 5 Rules and Classes 198

- 5.1 Rule-based Automatic Installation 198
  - Rules File Explained 199 • Custom Rules 202 • Match Types for Rules 202 • Combine Attributes 203 • Rules File Structure 203 • Predefined System Attributes 204 • Rules with Dialogs 206
- 5.2 Classes 209
- 5.3 Mixing Rules and Classes 211
- 5.4 Merging of Rules and Classes 211

### IV UNDERSTANDING THE AUTO-INSTALLATION PROCESS 214

## 6 The Auto-Installation Process 215

- 6.1 Introduction 215
  - X11 Interface (graphical) 215 • Serial Console 215 • Text-based YaST Installation 215
- 6.2 Choosing the Right Boot Medium 216
  - Booting from a Flash Disk (for example, a USB stick) 216 • Booting from the SUSE Linux Enterprise Installation Media 217 • Booting via PXE over the Network 217

6.3	Invoking the Auto-Installation Process	218
	Command Line Options	218
	Auto-installing a Single System	224
	Combining the <b>linuxrc</b> info File with the AutoYaST Control File	225
6.4	System Configuration	225
	Post-Install and System Configuration	226
	System Customization	226
V	USES FOR AUTOYAST ON INSTALLED SYSTEMS	227
7	Running AutoYaST in an Installed System	228
VI	APPENDICES	230
A	Handling Rules	231
B	AutoYaST FAQ—Frequently Asked Questions	232
C	Advanced <b>linuxrc</b> Options	236
C.1	Passing Parameters to <b>linuxrc</b>	236
C.2	info File Format	237
C.3	Advanced Network Setup	239
D	Differences Between AutoYaST Profiles in SLE 42.3 and 15	241
D.1	Partitioning	241
	GPT Becomes the Default Partition Type on AMD64/Intel 64	241
	Setting Partition Numbers	241
	Forcing Primary Partitions	242
	Btrfs: Default Subvolume Name	242
	Btrfs: Disabling Subvolumes	242
	Reading an Existing /etc/fstab Is No Longer Supported	243
	Setting for Aligning Partitions Has Been Dropped	243
	Using the type to Define an Volume Group	243
D.2	Firewall Configuration	243
	Assigning Interfaces to Zones	245
	Opening Ports	247
	Opening firewalld Services	248
	For More Information	249

- D.3 NTP Configuration 249
- D.4 AutoYaST Packages Are Needed for the Second Stage 250
- D.5 The CA Management Module Has Been Dropped 250
- D.6 Upgrade 251
  - Software 251



# 1 Introduction to AutoYaST

## 1.1 Motivation

Standard installations of openSUSE Leap are based on a wizard workflow. This is user-friendly and efficient when installing on few machines. However, it becomes repetitive and time-consuming when installing on many machines.

To avoid this, you could do mass deployments by copying the hard disk of the first successful installation. Unfortunately, that leads to the issue that even minute configuration changes between each machine need to later be dealt with individually. For example, when using static IP addresses, these IP addresses would need to be reset for each machine.

A regular installation of openSUSE Leap 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 openSUSE Leap 15.2 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 packages `autoyast2` and `autoyast2-installation` need to be installed to run the second stage in the installed system correctly. Otherwise an error will be shown before booting into the installed system.

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:

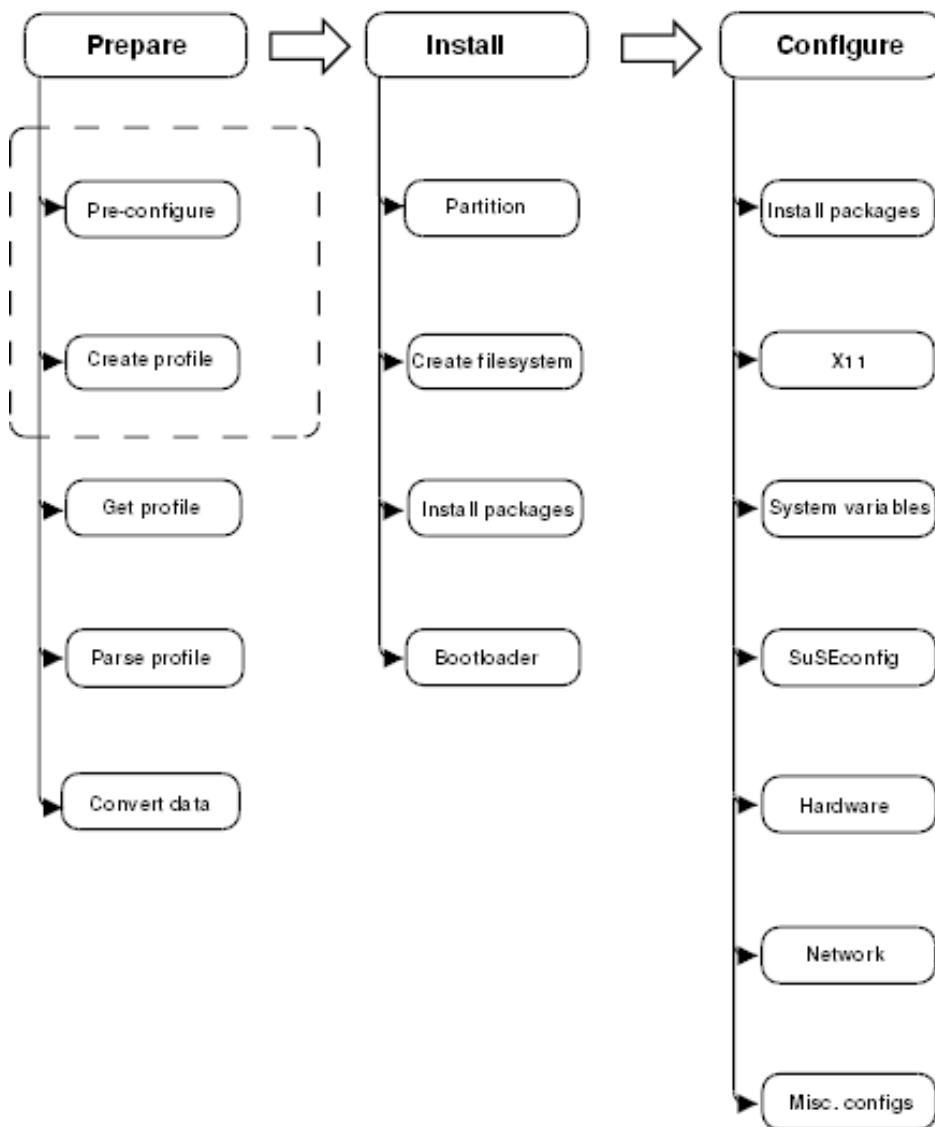


FIGURE 1.1: AUTO-INSTALLATION PROCESS

# I Understanding and Creating the AutoYaST Control File

- 2 The AutoYaST Control File 5
- 3 Creating an AutoYaST Control File 9

## 2 The AutoYaST Control File

### 2.1 Introduction

The control file 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 OS Releases Older Than SLES 12 GA and openSUSE 42.0 Are Incompatible

Many major changes were introduced with SLES 12 and openSUSE Leap 42.0, such as the switch to systemd and GRUB 2. These changes also required fundamental changes in AutoYaST. Therefore you cannot use AutoYaST control files created on SLES 11 to install openSUSE Leap 15.2 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.

An empty element, such as `<foo></foo>` or `<bar/>`, will *not* be present in the parsed data model. Usually this is interpreted as wanting a sensible default value. In cases where you need an explicitly empty string instead, use a CDATA section: `<foo><![CDATA[]]></foo>`.

### 2.3.2 Nested Resources

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

There are two kinds of nested resources: maps and lists. Maps, also known as associative arrays, hashes, or dictionaries, contain mixed contents, identified by their tag names. Lists, or arrays, have all items of the same type.

#### EXAMPLE 2.3: NESTED RESOURCES

```
...  
<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>
....
```

In the example above, the `drive` resource is a map consisting of a `device` property and a `partitions` resource. The `partitions` resource is a list containing multiple instances of the `partition` resource. Each `partition` resource is a map containing a `size` and `mount` property.

The default type of a nested resource is map. Lists must be marked as such using the `config:type="list"` attribute.

### 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.

The `config:type` attribute determines the type of the resource or property in the parsed data model. For resources, lists need a `list` type whereas a map is the default type that does not need an attribute. For properties, `boolean`, `symbol`, and `integer` can be used, the default being a string.

Attributes are not optional. It may appear that attributes are optional, because various parts of the schema are not very consistent in their usage of data types. In some places an enumeration is represented by a symbol, elsewhere a string is required. One resource needs `config:type="integer"`, another will parse the number from a string property. Some resources use `config:type="boolean"`, others want `yes` or even `1`. If in doubt, consult the schema file.



## 3 Creating an AutoYaST 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)

Also verify that both autoyast2-installation and autoyast2 are installed.

### 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 openSUSE Leap is installed.

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

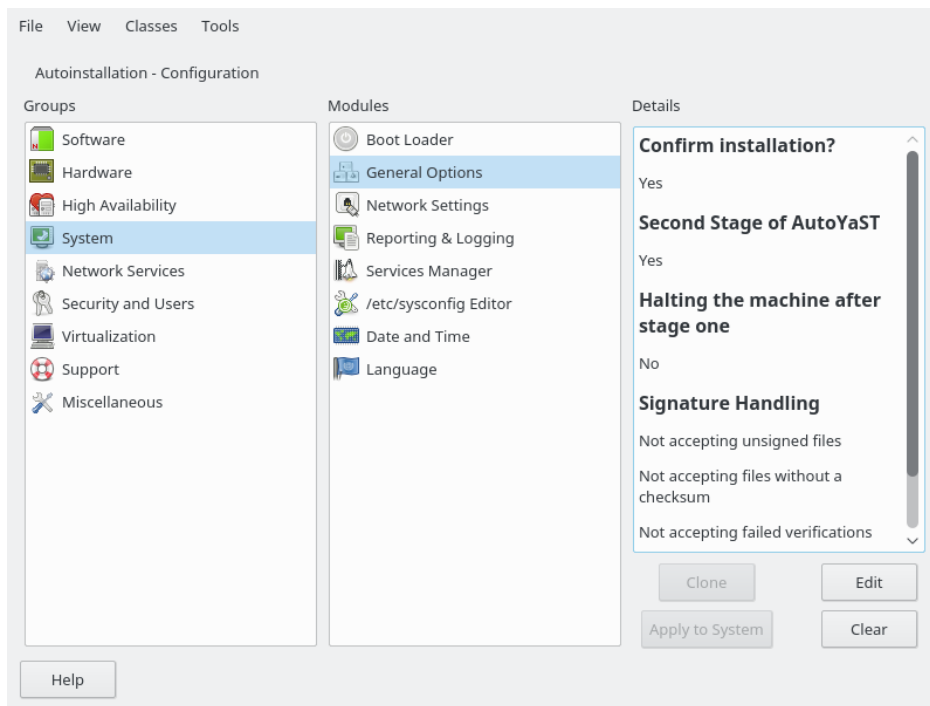


FIGURE 3.1: CONFIGURATION SYSTEM

### 3.2.1 Creating a New Control File

The easiest way to create an AutoYaST profile is to use an existing openSUSE Leap 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.

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.

```
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 often, you should consider scripting 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. This means that if there already is a domain element in the DNS section, you will get a second one, which will cause conflicts.

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

## II AutoYaST Configuration Examples

4 Configuration and Installation Options **15**

## 4 Configuration and Installation Options

This section contains configuration examples for services, registration, user and group management, upgrades, partitioning, configuration management, SSH key management, firewall configuration, and other 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>①
      ...
    </ask-list>
    <cio_ignore>
      ...
    </cio_ignore>
    <mode>②
      ...
    </mode>
    <proposals>③
      ...
    </proposals>
```

```

<self_update> ④
  ...
</self_update>
<self_update_url>
  ...
</self_update_url>
<semi-automatic config:type="list"> ⑤
  ...
</semi-automatic>
<signature-handling> ⑥
  ...
</signature-handling>
<storage> ⑦
  ...
</storage>
<wait> ⑧
  ...
</wait>
</general>
</profile>

```

- ① *Section 4.32, "Ask the User for Values during Installation"*
- ② *Section 4.1.1, "The Mode Section"*
- ③ *Section 4.1.2, "Configuring the Installation Settings Screen"*
- ④ *Section 4.1.3, "The Self-Update Section"*
- ⑤ *Section 4.1.4, "The Semi-Automatic Section"*
- ⑥ *Section 4.1.5, "The Signature Handling Section"*
- ⑦ *Section 4.4, "Partitioning"*
- ⑧ *Section 4.1.6, "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 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 4.8.3, "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

When set to true, the machine shuts down after everything is installed and configured at the end of the second stage. If you enable final\_halt, you do not need to set the final\_reboot option to true.

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

### final\_reboot

When set to true, the machine reboots after everything is installed and configured at the end of the second stage. If you enable final\_reboot, you do not need to set the final\_halt option 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>
```

### forceboot

Some openSUSE releases use Kexec to avoid the reboot after the first stage. They immediately boot into the installed system. You can force a reboot by setting this to true. Setting this value is optional. The default is set by the product.

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

```
</general>
```

## Important: Drivers May Need a Reboot

Some drivers, for example the proprietary drivers for Nvidia and ATI graphics cards, need a reboot and will not work properly when using Kexec. Therefore the default on openSUSE Leap products is to always do a proper reboot.

### 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 openSUSE Leap 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.

## ! Important: Quarterly Media Update: Self-Update Disabled

The installer self-update is only available if you use the GM images of the Unified Installer and Packages ISOs. If you install from the ISOs published as quarterly updates (they can be identified by the string QU in the name), the installer cannot update itself, because this feature has been disabled in the update media.

Use the following tags to configure the YaST self-update:

### self\_update

If set to true or false, this option enables or disables 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 to give the 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 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 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 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>
    ...
  </wait>
</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>
    ...
  </wait>
</general>

```

## 4.1.7 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-modules 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/configs">
  <general>
    <!-- Use cio_ignore on &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>
  </general>
</profile>
```

```

</accept_non_trusted_gpg_key>
</signature-handling>
<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 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>
```

### 4.3.1 Loader Type

This defines which boot loader (UEFI or BIOS/legacy) to use. Not all architectures support both legacy and EFI variants of the boot loader. The safest (`default`) option is to leave the decision up to the installer.

```
<loader_type>LOADER_TYPE</loader_type>
```

Possible values for `LOADER_TYPE` are:

- `default`: The installer chooses the correct boot loader. This is the default when no option is defined.
- `grub2`: Use the legacy BIOS boot loader.
- `grub2-efi`: Use the EFI boot loader.
- `none`: The boot process is not managed and configured by the installer.

### 4.3.2 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
<code>activate</code>	Set the boot flag on the boot partition. The boot partition can be <code>/</code> if there is no separate <code>/boot</code> partition. If the boot partition is on a logical partition, the boot flag is set to the extended partition. <pre>&lt;activate config:type="boolean"&gt;true&lt;/activate&gt;</pre>
<code>append</code>	Kernel parameters added at the end of boot entries for normal and recovery mode.

Attribute	Description
	<pre data-bbox="805 235 1414 286">&lt;append&gt;nomodeset vga=0x317&lt;/append&gt;</pre>
<u>boot_boot</u>	<p data-bbox="805 322 1414 454">Write GRUB 2 to a separate <code>/boot</code> partition. If no separate <code>/boot</code> partition exists, GRUB 2 will be written to <code>/</code>.</p> <pre data-bbox="805 488 1414 539">&lt;boot_boot&gt;&gt;false&lt;/boot_boot&gt;</pre>
<u>boot_custom</u>	<p data-bbox="805 580 1414 618">Write GRUB 2 to a custom device.</p> <pre data-bbox="805 651 1414 703">&lt;boot_custom&gt;/dev/sda3&lt;/boot_custom&gt;</pre>
<u>boot_extended</u>	<p data-bbox="805 743 1414 1016">Write GRUB 2 to the extended partition (important if you want to use generic boot code and the <code>/boot</code> partition is logical). Note: if the boot partition is logical, you should use <code>boot_mbr</code> (write GRUB 2 to MBR) rather than <code>generic_mbr</code>.</p> <pre data-bbox="805 1050 1414 1102">&lt;boot_extended&gt;&gt;false&lt;/boot_extended&gt;</pre>
<u>boot_mbr</u>	<p data-bbox="805 1146 1414 1272">Write GRUB 2 to the MBR of the first disk in the order (device.map includes order of disks).</p> <pre data-bbox="805 1305 1414 1357">&lt;boot_mbr&gt;&gt;false&lt;/boot_mbr&gt;</pre>
<u>boot_root</u>	<p data-bbox="805 1404 1414 1442">Write GRUB 2 to <code>/</code> partition.</p> <pre data-bbox="805 1476 1414 1527">&lt;boot_root&gt;&gt;false&lt;/boot_root&gt;</pre>
<u>generic_mbr</u>	<p data-bbox="805 1568 1414 1653">Write generic boot code to the MBR (will be ignored if <code>boot_mbr</code> is set to <code>true</code>).</p> <pre data-bbox="805 1686 1414 1765">&lt;generic_mbr config:type="boolean"&gt;&gt;false&lt;/generic_mbr&gt;</pre>

Attribute	Description
<u>gfxmode</u>	<p>Graphical resolution of the GRUB 2 screen (requires <code>&lt;terminal&gt;</code> to be set to <code>gfxterm</code>. Valid entries are <code>auto</code>, <code>HORIZONTALxVERTICAL</code>, or <code>HORIZONTALxVERTICALxCOLOR DEPTH</code>. You can see the screen resolutions supported by GRUB 2 on a particular system by using the <code>vbeinfo</code> command at the GRUB 2 command line in the running system.</p> <pre data-bbox="810 696 1402 752" style="border: 1px solid #ccc; padding: 5px;">&lt;gfxmode&gt;1280x1024x24&lt;/gfxmode&gt;</pre>
<u>os_prober</u>	<p>If set to <code>true</code>, automatically searches for operating systems already installed and generates boot entries for them during the installation</p> <pre data-bbox="810 1003 1402 1093" style="border: 1px solid #ccc; padding: 5px;">&lt;os_prober config:type="boolean"&gt;false&lt;/os_prober&gt;</pre>
<u>cpu_mitigations</u>	<p>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:</p> <p><code>auto</code>. 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.</p> <p><code>nosmt</code>. Provides the full set of available security mitigations. Enables all mitigations required for your CPU model. In addition, it disables Simultaneous Multithreading (SMT)</p>



Attribute	Description
	<p>to avoid side-channel attacks across multiple CPU threads. This setting may further impact performance, depending on the workload.</p> <p><u>off</u>. Disables all mitigations. Side-channel attacks against your CPU are possible, depending on the CPU model. This setting has no impact on performance.</p> <p><u>manual</u>. Does not set any mitigation level. Specify your CPU mitigations manually by using the kernel command line options.</p> <pre data-bbox="809 779 1412 840" style="border: 1px solid #ccc; padding: 5px;">&lt;cpu_mitigations&gt;auto&lt;/cpu_mitigations&gt;</pre> <p>If not set in AutoYaST, the respective settings can be changed via kernel command line. By default, the (product-specific) settings in the <code>/control.xml</code> file on the installation medium are used (if nothing else is specified).</p>
<u>suse_btrfs</u>	<p>Obsolete and no longer used. Booting from Btrfs snapshots is automatically enabled.</p>
<u>serial</u>	<p>Command to execute if the GRUB 2 terminal mode is set to <u>serial</u>.</p> <pre data-bbox="809 1424 1412 1592" style="border: 1px solid #ccc; padding: 5px;">&lt;serial&gt;   serial --speed=115200 --unit=0 --word=8   --parity=no --stop=1 &lt;/serials&gt;</pre>
<u>secure_boot</u>	<p>If set to <u>false</u>, then UEFI secure boot is disabled. Works only for <u>grub2-efi</u> boot loader.</p> <pre data-bbox="809 1794 1412 1854" style="border: 1px solid #ccc; padding: 5px;">&lt;secure_boot"&gt;false&lt;/secure_boot&gt;</pre>

Attribute	Description
<u>terminal</u>	<p>Specify the GRUB 2 terminal mode to use, Valid entries are <u>console</u>, <u>gfxterm</u>, and <u>serial</u>. If set to <u>serial</u>, the serial command needs to be specified with <code>&lt;serial&gt;</code>, too.</p> <pre data-bbox="809 506 1410 562">&lt;terminal&gt;serial&lt;/terminal&gt;</pre>
<u>timeout</u>	<p>The timeout in seconds until the default boot entry is booted automatically.</p> <pre data-bbox="809 719 1410 810">&lt;timeout config:type="integer"&gt;10&lt;/timeout&gt;</pre>
<u>trusted_boot</u>	<p>If set to <u>true</u>, then Trusted GRUB is used. Trusted GRUB supports Trusted Platform Module (TPM). Works only for <u>grub2</u> boot loader.</p> <pre data-bbox="809 1061 1410 1117">&lt;trusted_boot"&gt;true&lt;/trusted_boot&gt;</pre>
<u>vgamode</u>	<p>Adds the kernel parameter <u>vga=VALUE</u> to the boot entries.</p> <pre data-bbox="809 1274 1410 1330">&lt;vgamode&gt;0x317&lt;/vgamode&gt;</pre>
<u>xen_append</u>	<p>Kernel parameters added at the end of boot entries for Xen guests.</p> <pre data-bbox="809 1487 1410 1576">&lt;xen_append&gt;nomodeset vga=0x317&lt;/xen_append&gt;</pre>
<u>xen_kernel_append</u>	<p>Kernel parameters added at the end of boot entries for Xen kernels on the VM Host Server.</p>

Attribute	Description
	<pre data-bbox="818 239 1286 304">&lt;xen_kernel_append&gt;dom0_mem=768M&lt;/xen_kernel_append&gt;</pre>

### 4.3.3 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>
```

## 4.4 Partitioning

When it comes to partitioning, we can categorize AutoYaST use cases into three different levels:

- Automatic partitioning. The user does not care about the partitioning and trusts in AutoYaST to do the right thing.
- Guided partitioning. The user would like to set some basic settings. For example, a user would like to use LVM but has no idea about how to configure partitions, volume groups, and so on.
- Expert partitioning. The user specifies how the layout should look. However, a complete definition is not required, and AutoYaST should propose reasonable defaults for missing parts.

To some extent, it is like using the regular installer. You can skip the partitioning screen and trust in YaST, use the *Guided Proposal*, or define the partitioning layout through the *Expert Partitioner*.

## 4.4.1 Automatic Partitioning

AutoYaST can come up with a sensible partitioning layout without any user indication. Although it depends on the selected product to install, AutoYaST usually proposes a Btrfs root file system, a separate `/home` using XFS and a swap partition. Additionally, depending on the architecture, it adds any partition that might be needed to boot (like BIOS GRUB partitions).

However, these defaults might change depending on factors like the available disk space. For example, having a separate `/home` depends on the amount of available disk space.

If you want to influence these default values, you can use the approach described in [Section 4.4.2, “Guided Partitioning”](#).

## 4.4.2 Guided Partitioning

Although AutoYaST can come up with a partitioning layout without any user indication, sometimes it is useful to set some generic parameters and let AutoYaST do the rest. For example, you may be interested in using LVM or encrypting your file systems without having to deal with the details. It is similar to what you would do when using the guided proposal in a regular installation.

The `storage` section in [Example 4.3, “LVM-based Guided Partitioning”](#) instructs AutoYaST to set up a partitioning layout using LVM and deleting all Windows partitions, no matter whether they are needed.

### EXAMPLE 4.3: LVM-BASED GUIDED PARTITIONING

```
<general>
  <storage>
    <proposal>
      <lvm config:type="boolean">true</lvm>
      <windows_delete_mode config:type="symbol">all</windows_delete_mode>
    </proposal>
  </storage>
</general>
```

#### **lvm**

Creates an LVM-based proposal. The default is `false`.

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

#### **resize\_windows**

When set to true, AutoYaST resizes Windows partitions if needed to make room for the installation.

```
<resize_windows config:type="boolean">>false</resize_windows>
```

#### windows\_delete\_mode

- none does not remove Windows partitions.
- ondemand removes Windows partitions if needed.
- all removes all Windows partitions.

```
<windows_delete_mode config:type="symbol">ondemand</windows_delete_mode>
```

#### linux\_delete\_mode

- none does not remove Linux partitions.
- ondemand removes Linux partitions if needed.
- all removes all Linux partitions.

```
<linux_delete_mode config:type="symbol">ondemand</linux_delete_mode>
```

#### other\_delete\_mode

- none does not remove other partitions.
- ondemand removes other partitions if needed.
- all removes all other partitions.

```
<other_delete_mode config:type="symbol">ondemand</other_delete_mode>
```

#### encryption\_password

Enables encryption using the specified password. By default, encryption is disabled.

```
<encryption_password>some-secret</encryption_password>
```

### 4.4.3 Expert Partitioning

As an alternative to guided partitioning, AutoYaST allows to describe the partitioning layout through a partitioning section. However, AutoYaST does not need to know every single detail and is able to build a sensible layout from a rather incomplete specification.

The `partitioning` section is a list of `drive` elements. Each of these sections describes an element of the partitioning layout like a disk, an LVM volume group, a RAID, a multi-device Btrfs file system, and so on.

*Example 4.4, "Creating /, /home and swap partitions"*, asks AutoYaST to create a `/`, a `/home` and a `swap` partition using the whole disk. Note that some information is missing, like which file systems each partition should use. However, that is not a problem, and AutoYaST will propose sensible values for them.

#### EXAMPLE 4.4: CREATING /, /home AND swap PARTITIONS

```
<partitioning config:type="list">
  <drive>
    <use>all</use>
    <partitions config:type="list">
      <partition>
        <mount>/</mount>
        <size>20GiB</size>
      </partition>
      <partition>
        <mount>/home</mount>
        <size>max</size>
      </partition>
      <partition>
        <mount>swap</mount>
        <size>1GiB</size>
      </partition>
    </partitions>
  </drive>
```



### Tip: Proposing a Boot Partition

AutoYaST checks whether the layout described in the profile is bootable or not. If it is not, it adds the missing partitions. So, if you are unsure about which partitions are needed to boot, you can rely on AutoYaST to make the right decision.

#### 4.4.3.1 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
<u>device</u>	<p>The device you want to configure in this section. You can use persistent device names via ID, like <u>/dev/disk/by-id/ata-WDC_WD3200AAKS-75L9A0_WD-WMAV27368122</u> or <u>by-path</u>, like <u>/dev/disk/by-path/pci-0001:00:03.0-scsi-0:0:0:0</u>.</p> <pre>&lt;device&gt;/dev/sda&lt;/device&gt;</pre> <p>In case of volume groups, software RAID or <u>bcache</u> devices, the name in the installed system may be different (to avoid clashes with existing devices). See <a href="#">Section 4.4.7, "Multipath Support"</a> for further information about dealing with multipath devices.</p>	<p>Optional. If left out, AutoYaST tries to guess the device. See <a href="#">Tip: Skipping Devices</a> on how to influence guessing.</p> <p>If set to <u>ask</u>, AutoYaST will ask the user which device to use during installation.</p>
<u>initialize</u>	<p>If set to <u>true</u>, the partition table gets wiped out before AutoYaST starts the partition calculation.</p>	<p>Optional. The default is <u>false</u>.</p>

Attribute	Values	Description
	<pre data-bbox="603 235 987 351">&lt;initialize   config:type="boolean"&gt;true&lt;/ initialize&gt;</pre>	
<p data-bbox="186 398 352 427"><u>partitions</u></p>	<p data-bbox="603 398 987 524">A list of <code>&lt;partition&gt;</code> entries (see <a href="#">Section 4.4.3.2, "Partition Configuration"</a>).</p> <pre data-bbox="603 562 987 790">&lt;partitions   config:type="list"&gt;   &lt;partition&gt;...&lt;/ partition&gt;   ... &lt;/partitions&gt;</pre>	<p data-bbox="1019 398 1350 667">Optional. If no partitions are specified, AutoYaST will create a reasonable partitioning (see <a href="#">Section 4.4.3.5, "Filling the Gaps"</a>).</p>
<p data-bbox="186 840 288 869"><u>pesize</u></p>	<p data-bbox="603 840 987 913">This value only makes sense with LVM.</p> <pre data-bbox="603 952 987 1003">&lt;pesize&gt;8M&lt;/pesize&gt;</pre>	<p data-bbox="1019 840 1378 913">Optional. Default is 4M for LVM volume groups.</p>
<p data-bbox="186 1052 240 1081"><u>use</u></p>	<p data-bbox="603 1052 987 1178">Specifies the strategy AutoYaST will use to partition the hard disk.</p> <p data-bbox="603 1205 823 1234">Choose between:</p> <ul data-bbox="644 1279 995 1572" style="list-style-type: none"> <li data-bbox="644 1279 995 1404">• <u>all</u> (uses the whole device while calculating the new partitioning),</li> <li data-bbox="644 1449 995 1572">• <u>linux</u> (only existing Linux partitions are used),</li> </ul>	<p data-bbox="1019 1052 1358 1126">This parameter should be provided.</p>



Attribute	Values	Description
	<ul style="list-style-type: none"> <li>• <u>free</u> (only unused space on the device is used, no other partitions are touched),</li> <li>• 1,2,3 (a list of comma separated partition numbers to use).</li> </ul>	
<u>type</u>	<p>Specify the type of the <u>drive</u>,</p> <p>Choose between:</p> <ul style="list-style-type: none"> <li>• <u>CT_DISK</u> for physical hard disks (default),</li> <li>• <u>CT_LVM</u> for LVM volume groups,</li> <li>• <u>CT_RAID</u> for software RAID devices,</li> <li>• <u>CT_BCACHE</u> for software <u>bcache</u> devices.</li> </ul> <pre data-bbox="600 1319 994 1453" style="border: 1px solid gray; padding: 5px;"> &lt;type   config:type="symbol"&gt;CT_LVM&lt;/ type&gt;</pre>	Optional. Default is <u>CT_DISK</u> for a normal physical hard disk.
<u>disklabel</u>	<p>Describes the type of the partition table.</p> <p>Choose between:</p> <ul style="list-style-type: none"> <li>• <u>msdos</u></li> <li>• <u>gpt</u></li> <li>• <u>none</u></li> </ul>	Optional. By default YaST decides what makes sense. If a partition table of a different type already exists, it will be recreated with the given type only if it does not include any partition that should be kept or reused.

Attribute	Values	Description
	<pre data-bbox="603 230 987 286">&lt;disklabel&gt;gpt&lt;/disklabel&gt;</pre>	<p data-bbox="1019 230 1362 365">To use the disk without creating any partition, set this element to <u>none</u>.</p>
<p data-bbox="186 405 432 439"><u>keep_unknown_lv</u></p>	<p data-bbox="603 405 987 954">This value only makes sense for type = CT_LVM drives. If you are reusing a logical volume group and you set this to <u>true</u>, all existing logical volumes in that group will not be touched unless they are specified in the &lt;partitioning&gt; section. So you can keep existing logical volumes without specifying them.</p> <pre data-bbox="603 992 987 1160">&lt;keep_unknown_lv   config:type="boolean"   &gt;false&lt;/ keep_unknown_lv&gt;</pre>	<p data-bbox="1019 405 1334 483">Optional. The default is <u>false</u>.</p>
<p data-bbox="186 1196 448 1229"><u>enable_snapshots</u></p>	<p data-bbox="603 1196 987 1420">Enables snapshots on Btrfs file systems mounted at <u>/</u> (does not apply to other file systems, or Btrfs file systems not mounted at <u>/</u>).</p> <pre data-bbox="603 1458 987 1626">&lt;enable_snapshots   config:type="boolean"   &gt;false&lt;/ enable_snapshots&gt;</pre>	<p data-bbox="1019 1196 1334 1274">Optional. The default is <u>true</u>.</p>

## 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.

## 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>`s, run `yast2 ayast_probe` on a system that has already been installed.

### 4.4.3.2 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
<u>create</u>	Specify if this partition or logical volume must be created or if it already exists.  <pre>&lt;create config:type="boolean" &gt;false&lt;/create&gt;</pre>	If set to <u>false</u> , you also need to set one of <u>partition_nr</u> , <u>lv_name</u> , <u>label</u> or <u>uuid</u> to tell AutoYaST which device to use.
<u>crypt_method</u>	Partition will be encrypted using one of these methods: <ul style="list-style-type: none"> <li>• <u>luks1</u>: regular LUKS1 encryption.</li> <li>• <u>pervasive_luks2</u>: pervasive volume encryption.</li> <li>• <u>protected_swap</u>: encryption with volatile protected key.</li> <li>• <u>secure_swap</u>: encryption with volatile secure key.</li> <li>• <u>random_swap</u>: encryption with volatile random key.</li> </ul> <pre>&lt;crypt_method config:type="symbol"&gt;luks1&lt;/crypt_method&gt;</pre>	Optional. Encryption method selection was introduced in openSUSE Leap 15.2. To mimic the behavior of previous versions, use <u>luks1</u> .  See <u>crypt_key</u> element to find out how to specify the encryption password if needed.

Attribute	Values	Description
<u>crypt_fs</u>	<p>Partition will be encrypted. This element is deprecated. Use <u>crypt_method</u> instead.</p> <pre>&lt;crypt_fs config:type="boolean"&gt;true&lt;/crypt_fs&gt;</pre>	Default is <u>false</u> .
<u>crypt_key</u>	<p>Encryption key</p> <pre>&lt;crypt_key&gt;xxxxxxx&lt;/crypt_key&gt;</pre>	Needed if <u>crypt_method</u> has been set to a method that requires a password (i.e., <u>luks1</u> or <u>pervasive_luks2</u> ).
<u>mount</u>	<p>The mount point of this partition.</p> <pre>&lt;mount&gt;/&lt;/mount&gt; &lt;mount&gt;swap&lt;/mount&gt;</pre>	You should have at least a root partition (/) and a swap partition.
<u>fstopt</u>	<p>Mount options for this partition.</p> <pre>&lt;fstopt&gt; ro,noatime,user,data=ordered,acl,user_xattr &lt;/fstopt&gt;</pre>	See <u>man mount</u> for available mount options.
<u>label</u>	<p>The label of the partition. Useful when formatting the device (especially if the <u>mountby</u> parameter is set to <u>label</u>) and for identifying a device that already exists (see <u>create</u> above).</p> <pre>&lt;label&gt;mydata&lt;/label&gt;</pre>	See <u>man e2label</u> for an example.
<u>uuid</u>	<p>The uuid of the partition. Only useful for identifying an existing device (see <u>create</u> above). The uuid cannot be enforced for new devices.</p> <pre>&lt;uuid</pre>	See <u>man uuidgen</u> .

Attribute	Values	Description
	<pre>&gt;1b4e28ba-2fa1-11d2-883f-b9a761bde3fb&lt;/ uuid&gt;</pre>	
<u>size</u>	<p>The size of the partition, for example 4G, 4500M, etc. The /boot partition and the swap partition can have <u>auto</u> as size. Then AutoYaST calculates a reasonable size. One partition can have the value <u>max</u> to use all remaining space.</p> <p>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.</p> <pre>&lt;size&gt;10G&lt;/size&gt;</pre>	<p>Starting with openSUSE Leap 15, all values (including <u>auto</u> and <u>max</u>) can be used for resizing partitions as well.</p>
<u>format</u>	<p>Specify if AutoYaST should format the partition.</p> <pre>&lt;format config:type="boolean"&gt;&gt;false&lt;/ format&gt;</pre>	<p>If you set <u>create</u> to <u>true</u>, then you likely want this option set to <u>true</u> as well.</p>
<u>file system</u>	<p>Specify the file system to use on this partition:</p> <ul style="list-style-type: none"> <li>• <u>btrfs</u></li> <li>• <u>ext2</u></li> <li>• <u>ext3</u></li> <li>• <u>ext4</u></li> <li>• <u>fat</u></li> <li>• <u>xf</u>s</li> <li>• <u>swap</u></li> </ul> <pre>&lt;filesystem config:type="symbol"</pre>	<p>Optional. The default is <u>btrfs</u> for the root partition (/) and <u>xf</u>s for data partitions.</p>

Attribute	Values	Description
	<pre>&gt;ext3&lt;/filesystem&gt;</pre>	
<u>mkfs_options</u>	<p>Specify an option string that is added to the mkfs command.</p> <pre>&lt;mkfs_options&gt;-I 128&lt;/mkfs_options&gt;</pre>	Optional. Only use this when you know what you are doing.
<u>partition_nr</u>	<p>The <u>partition_nr</u> of this partition. If you have set <u>create=false</u> or if you use LVM, then you can specify the partition via <u>partition_nr</u>. You can force AutoYaST to only create primary partitions by assigning numbers below 5.</p> <pre>&lt;partition_nr config:type="integer" &gt;2&lt;/partition_nr&gt;</pre>	Usually, numbers 1 to 4 are primary partitions while 5 and higher are logical partitions.
<u>partition_id</u>	<p>The <u>partition_id</u> sets the id of the partition. If you want different identifiers than 131 for Linux partition or 130 for swap, configure them with <u>partition_id</u>.</p> <pre>&lt;partition_id config:type="integer" &gt;131&lt;/partition_id&gt;</pre> <p>Possible values are:</p> <p>Swap: <u>130</u>  Linux: <u>131</u>  LVM: <u>142</u>  MD RAID: <u>253</u>  EFI partition: <u>259</u></p>	The default is <u>131</u> for Linux partition and <u>130</u> for swap.
<u>partition_type</u>	<p>When using an <u>msdos</u> partition table, this element sets the type of the partition. The value can be <u>primary</u> or <u>logical</u>. This</p>	Optional. Allowed values are <u>primary</u> (default) and <u>logical</u> .

Attribute	Values	Description
	<p>value is ignored when using a <code>gpt</code> partition table, because such a distinction does not exist in that case.</p> <pre data-bbox="391 392 995 448">&lt;partition_type&gt;primary&lt;/partition_type&gt;</pre>	
<u>mountby</u>	<p>Instead of a partition number, you can tell AutoYaST to mount a partition by <u>device</u>, <u>label</u>, <u>uuid</u>, <u>path</u> or <u>id</u>, which are the udev path and udev id (see <code>/dev/disk/...</code>).</p> <pre data-bbox="391 750 995 840">&lt;mountby config:type="symbol"&gt;label&lt;/mountby&gt;</pre>	<p>See <u>label</u> and <u>uuid</u> documentation above. The default depends on YaST and usually is <u>id</u>.</p>
<u>subvolumes</u>	<p>List of subvolumes to create for a file system of type Btrfs. This key only makes sense for file systems of type Btrfs. See <a href="#">Section 4.4.3.3, "Btrfs subvolumes"</a> for more information.</p> <pre data-bbox="391 1086 995 1507">&lt;subvolumes config:type="list"&gt;   &lt;path&gt;tmp&lt;/path&gt;   &lt;path&gt;opt&lt;/path&gt;   &lt;path&gt;srv&lt;/path&gt;   &lt;path&gt;var/crash&lt;/path&gt;   &lt;path&gt;var/lock&lt;/path&gt;   &lt;path&gt;var/run&lt;/path&gt;   &lt;path&gt;var/tmp&lt;/path&gt;   &lt;path&gt;var/spool&lt;/path&gt;   ... &lt;/subvolumes&gt;</pre>	<p>If no <u>subvolumes</u> section has been defined for a partition description, AutoYaST will create a predefined set of subvolumes for the given mount point.</p>
<u>create_subvolumes</u>	<p>Determine whether Btrfs subvolumes should be created or not.</p>	<p>It is set to <u>true</u> by default. When set to <u>false</u>, no subvolumes will be created.</p>



Attribute	Values	Description
<u>subvolumes_prefix</u>	<p>Set the Btrfs subvolumes prefix name. If no prefix is wanted, it must be set to an empty value:</p> <pre>&lt;subvolumes_prefix&gt;&lt;![CDATA[]]&gt;&lt;/subvolumes_prefix&gt;</pre>	It is set to <u>@</u> by default.
<u>lv_name</u>	<p>If this partition is on a logical volume in a volume group, specify the logical volume name here (see the <u>type</u> parameter in the drive configuration).</p> <pre>&lt;lv_name&gt;opt_lv&lt;/lv_name&gt;</pre>	
<u>stripes</u>	<p>An integer that configures LVM striping. Specify across how many devices you want to stripe (spread data).</p> <pre>&lt;stripes config:type="integer"&gt;2&lt;/stripes&gt;</pre>	
<u>stripesize</u>	<p>Specify the size of each block in KB.</p> <pre>&lt;stripesize config:type="integer"&gt;4&lt;/stripesize&gt;</pre>	
<u>lvm_group</u>	<p>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.</p> <pre>&lt;lvm_group&gt;system&lt;/lvm_group&gt;</pre>	
<u>pool</u>	<p><u>pool</u> must be set to <u>true</u> if the LVM logical volume should be an LVM thin pool.</p> <pre>&lt;pool config:type="boolean"&gt;&gt;false&lt;/pool&gt;</pre>	

Attribute	Values	Description
<u>used_pool</u>	<p>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.</p> <pre>&lt;used_pool&gt;my_thin_pool&lt;/used_pool&gt;</pre>	
<u>raid_name</u>	<p>If this physical volume is part of a RAID, specify the name of the RAID.</p> <pre>&lt;raid_name&gt;/dev/md/0&lt;/raid_name&gt;</pre>	
<u>raid_options</u>	<p>Specify RAID options, see below.</p> <pre>&lt;raid_options&gt;...&lt;/raid_options&gt;</pre>	<p>Setting the RAID options at <u>partition</u> level is deprecated. See <a href="#">Section 4.4.6, "Software RAID"</a>.</p>
<u>bcache_backing_for</u>	<p>If this device is used as a <u>bcache backing device</u>, specify the name of the <u>bcache</u> device.</p> <pre>&lt;bcache_backing_for&gt;/dev/bcache0&lt;/bcache_backing_for&gt;</pre>	<p>See <a href="#">Section 4.4.8, "bcache Configuration"</a> for further details.</p>
<u>bcache_caching_for</u>	<p>If this device is used as a <u>bcache caching device</u>, specify the names of the <u>bcache</u> devices.</p> <pre>&lt;bcache_caching_for config:type="list"&gt;   &lt;listentry&gt;/dev/bcache0&lt;/listentry&gt; &lt;/bcache_caching_for&gt;</pre>	<p>See <a href="#">Section 4.4.8, "bcache Configuration"</a> for further details.</p>
<u>resize</u>	<p>This boolean must be <u>true</u> if an existing partition should be resized. In this case, you need to tell AutoYaST to reuse the device (see <u>create</u>) and specify a <u>size</u>.</p>	<p>Starting with openSUSE Leap 15 resizing works with physical disk partitions and with LVM volumes.</p>

Attribute	Values	Description
	<code>&lt;resize config:type="boolean"&gt;false&lt;/resize&gt;</code>	

### 4.4.3.3 Btrfs subvolumes

As mentioned in [Section 4.4.3.2, "Partition Configuration"](#), it is possible to define a set of subvolumes for each Btrfs file system. In its simplest form, this is 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 openSUSE Leap), the name of this default subvolume is automatically prefixed to the names in this list. This behavior can be disabled by setting the `subvolumes_prefix`.

```
<subvolumes_prefix><![CDATA[]]></subvolumes_prefix>
```

#### 4.4.3.4 Using the Whole Disk

AutoYaST allows to use a whole disk without creating any partition by setting the `disklabel` to `none` as described in [Section 4.4.3.1, "Drive Configuration"](#). In such cases, the configuration in the first `partition` from the `drive` will be applied to the whole disk.

In the example below, we are using the second disk (`/dev/sdb`) as the `/home` file system.

##### EXAMPLE 4.5: USING A WHOLE DISK AS A FILE SYSTEM

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <partitions config:type="list">
      <partition>
        <create config:type="boolean">true</create>
        <format config:type="boolean">true</format>
        <mount>/</mount>
        <size>max</size>
      </partition>
    </partitions>
  </drive>
  <drive>
    <device>/dev/sdb</device>
    <disklabel>none</disklabel>
    <partitions config:type="list">
      <partition>
        <format config:type="boolean">true</format>
        <mount>/home</mount>
      </partition>
    </partitions>
  </drive>
```

In addition, the whole disk can be used as an LVM physical volume or as a software RAID member. See [Section 4.4.5, "Logical Volume Manager \(LVM\)"](#) and [Section 4.4.6, "Software RAID"](#) for further details about setting up an LVM or a software RAID.

For backward compatibility reasons, it is possible to achieve the same result by setting the `<partition_nr>` element to `0`. However, this usage of the `<partition_nr>` element is deprecated from openSUSE Leap 15.

### 4.4.3.5 Filling the Gaps

When using the Expert Partitioner approach, AutoYaST can create a partition plan from a rather incomplete profile. The following profiles show how you can describe some details of the partitioning layout and let AutoYaST do the rest.

#### EXAMPLE 4.6: AUTOMATED PARTITIONING ON SELECTED DRIVES

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 4.7: INSTALLING ON MULTIPLE DRIVES

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <use>all</use>
    <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>free</use>
    <partitions config:type="list">
      <partition>
        <filesystem config:type="symbol">ext4</filesystem>
        <mount>/data1</mount>
        <size>15G</size>
      </partition>
    </partitions>
  </drive>
</partitioning>
```

```
</partition>
<partition>
  <filesystem config:type="symbol">xfs</filesystem>
  <mount>/data2</mount>
  <size>auto</size>
</partition>
</partitions>
</drive>
</partitioning>
```

## 4.4.4 Advanced Partitioning Features

### 4.4.4.1 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.

### 4.4.4.2 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 a new partition using a label, use the `label` property to specify its value. 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 4.8: MOUNT OPTIONS

```
<partitions config:type="list">
  <partition>
    <filesystem config:type="symbol">ext4</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>
```



## Note: Check Supported File System Options

Different file system types support different options. Check the documentation carefully before setting them.

### 4.4.4.3 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 specific 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 4.9: 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>
    </partitions>
  </drive>
</partitioning>
```



```

</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.

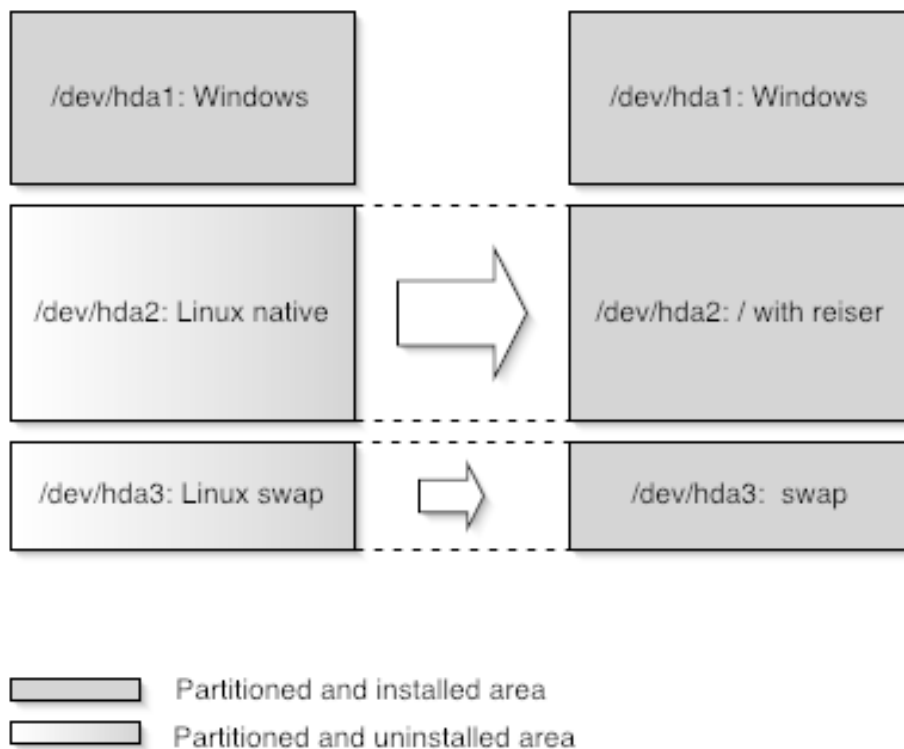


FIGURE 4.1: KEEPING PARTITIONS

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 4.10: 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>
```



### Note: Keeping Encrypted Devices

When AutoYaST is probing the storage devices, the partitioning section from the profile is not yet analyzed. In some scenarios, it is not clear which key should be used to unlock a device. For example, this can happen when more than one encryption key is defined. To solve this problem, AutoYaST will try all defined keys on all encrypted devices until a working key is found.

## 4.4.5 Logical Volume Manager (LVM)

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

#### EXAMPLE 4.11: 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 4.12: 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>
    <type config:type="symbol">CT_LVM</type>
    <partitions config:type="list">
      <partition>
        <filesystem config:type="symbol">ext4</filesystem>
        <lv_name>user_lv</lv_name>
        <mount>/usr</mount>
        <size>15G</size>
      </partition>
    </partitions>
  </drive>
```

```

<filesystem config:type="symbol">ext4</filesystem>
<lv_name>opt_lv</lv_name>
<mount>/opt</mount>
<size>10G</size>
</partition>
<partition>
  <filesystem config:type="symbol">ext4</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.

#### 4.4.6 Software RAID

The support for software RAID devices has been greatly improved in openSUSE Leap 15.2.

If needed, see [Section 4.4.6.1, "Using the Deprecated Syntax"](#) to find out further details about the old way of specifying a software RAID, which is still supported for backward compatibility.

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.

Similar to LVM, a software RAID definition in an AutoYaST profile is composed of two different parts:

- Determining which disks or partitions are going to be used as RAID members. In order to do that, you need to set the `raid_name` element in such devices.
- Defining the RAID itself by using a dedicated `drive` section.

The following example shows a RAID1 configuration that uses a partition from the first disk and another one from the second disk as RAID members:

### EXAMPLE 4.13: RAID1 CONFIGURATION

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <partitions config:type="list">
      <partition>
        <mount>/</mount>
        <size>20G</size>
      </partition>
      <partition>
        <raid_name>/dev/md/0</raid_name>
        <size>max</size>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
  <drive>
    <device>/dev/sdb</device>
    <disklabel>none</disklabel>
    <partitions config:type="list">
      <partition>
        <raid_name>/dev/md/0</raid_name>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
  <drive>
```

```

<device>/dev/md/0</device>
<partitions config:type="list">
  <partition>
    <mount>/home</mount>
    <size>40G</size>
  </partition>
  <partition>
    <mount>/srv</mount>
    <size>10G</size>
  </partition>
</partitions>
<raid_options>
  <chunk_size>4</chunk_size>
  <parity_algorithm>left_asymmetric</parity_algorithm>
  <raid_type>raid1</raid_type>
</raid_options>
<use>all</use>
</drive>
</partitioning>

```

If you do not want to create partitions in the software RAID, set the `disklabel` to `none` as you would do for a regular disk. In the example below, only the RAID `drive` section is shown for simplicity's sake:

#### EXAMPLE 4.14: RAID1 WITHOUT PARTITIONS

```

<drive>
  <device>/dev/md/0</device>
  <disklabel>none</disklabel>
  <partitions config:type="list">
    <partition>
      <mount>/home</mount>
      <size>40G</size>
    </partition>
  </partitions>
  <raid_options>
    <chunk_size>4</chunk_size>
    <parity_algorithm>left_asymmetric</parity_algorithm>
    <raid_type>raid1</raid_type>
  </raid_options>
  <use>all</use>
</drive>

```

#### 4.4.6.1 Using the Deprecated Syntax

If the installer self-update feature is enabled, it is possible to partition a software RAID for openSUSE Leap 15. However, that scenario was not supported in previous versions and hence the way to define a software RAID was slightly different.

This section defines what the old-style configuration looks like because it is still supported for backward compatibility.

Keep the following in mind when configuring a RAID using this deprecated syntax:

- 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/md/0 is configured. Adding several partition sections means that you want to have multiple software RAIDs (/dev/md/0, /dev/md/1, etc.).
- All RAID-specific options are contained in the raid\_options resource.

##### EXAMPLE 4.15: OLD STYLE 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>raid1</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>
```

```

    <size>4gb</size>
  </partition>
</partitions>
<use>all</use>
</drive>
<drive>
  <device>/dev/md</device>
  <partitions config:type="list">
    <partition>
      <filesystem config:type="symbol">ext4</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_options>
    </partition>
  </partitions>
  <use>all</use>
</drive>
</partitioning>

```

#### 4.4.6.2 RAID Options

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

```

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

```

Attribute	Values	Description
<u>chunk_size</u>	<code>&lt;chunk_size&gt;4&lt;/chunk_size&gt;</code>	
<u>parity_algorithm</u>	Possible values are: <u>left_asymmetric</u> , <u>left_symmetric</u> , <u>right_asymmetric</u> , <u>right_symmetric</u> .	



Attribute	Values	Description
	<p>For RAID6 and RAID10 the following values can be used:</p> <p><u>parity_first</u>,  <u>parity_last</u>,  <u>left_asymmetric_6</u>,  <u>left_symmetric_6</u>,  <u>right_asymmetric_6</u>,  <u>right_symmetric_6</u>,  <u>parity_first_6</u>, <u>n2</u>, <u>o2</u>,  <u>f2</u>, <u>n3</u>, <u>o3</u>, <u>f3</u></p> <pre data-bbox="603 741 987 907">&lt;parity_algorithm &gt;left_asymmetric&lt;/ parity_algorithm&gt;</pre>	
<u>raid_type</u>	<p>Possible values are: <u>raid0</u>, <u>raid1</u>, <u>raid5</u>, <u>raid6</u> and <u>raid10</u>.</p> <pre data-bbox="603 1108 987 1200">&lt;raid_type&gt;raid1&lt;/ raid_type&gt;</pre>	The default is <u>raid1</u> .
<u>device_order</u>	<p>This list contains the order of the physical devices:</p> <pre data-bbox="603 1355 987 1662">&lt;device_order config:type="list"&gt;   &lt;device&gt;/dev/ sdb2&lt;/device&gt;   &lt;device&gt;/dev/ sda1&lt;/device&gt;   ... &lt;/device_order&gt;</pre>	This is optional and the default is alphabetical order.

## 4.4.7 Multipath Support

AutoYaST can handle multipath devices. In order to take advantage of them, you need to enable multipath support, as shown in *Example 4.16, "Using Multipath Devices"*. Alternatively, you can use the following parameter on the Kernel command line: `LIBSTORAGE_MULTIPATH_AUTOSTART=ON`. Unlike SUSE Linux Enterprise 12, it is not required to set the drive section type to `CT_DMMULTIPATH`. You should use `CT_DISK`, although for historical reasons, both values are equivalent.

EXAMPLE 4.16: 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_DISK</type>
    <use>all</use>
  </drive>
</partitioning>
```

If you want to specify the device, you could use the World Wide Identifier (WWID), its device name (for example, `/dev/dm-0`), any other path under `/dev/disk` that refers to the multipath device or any of its paths.

For example, given the `multipath` listing from *Example 4.17, "Listing multipath devices"*, you could use `/dev/mapper/149455400000000000f86756dce9286158be4c6e3567e75ba5`, `/dev/dm-3`, any other corresponding path under `/dev/disk` (as shown in *Example 4.18, "Using the WWID to Identify a Multipath Device"*), or any of its paths (`/dev/sda` or `/dev/sdb`).

EXAMPLE 4.17: LISTING MULTIPATH DEVICES

```
# multipath -l
```

```

149455400000000000f86756dce9286158be4c6e3567e75ba5 dm-3 ATA,VIRTUAL-DISK
size=40G features='0' hwhandler='0' wp=rw
|+- policy='service-time 0' prio=1 status=active
| ` 2:0:0:0 sda 8:0 active ready running
`+- policy='service-time 0' prio=1 status=enabled
  ` 3:0:0:0 sdb 8:16 active ready running

```

#### EXAMPLE 4.18: USING THE WWID TO IDENTIFY A MULTIPATH DEVICE

```

<drive>
  <partitions config:type="list">
    <device>/dev/mapper/149455400000000000f86756dce9286158be4c6e3567e75ba5</device>
    <partition>
      <size>20G</size>
      <mount>/</mount>
      <filesystem config:type="symbol">ext4</filesystem>
    </partition>
  </partitions>
  <type config:type="symbol">CT_DISK</type>
  <use>all</use>
</drive>

```

## 4.4.8 bcache Configuration

bcache is a caching system which allows the use of multiple fast drives to speed up the access to one or more slower drives. For example, you can improve the performance of a large (but slow) drive by using a fast one as a cache.

For more information about bcache on openSUSE Leap, also see the blog post at <https://www.suse.com/c/combine-the-performance-of-solid-state-drive-with-the-capacity-of-a-hard-drive-with-bcache-and-yast/>.

To set up a bcache device, AutoYaST needs a profile that specifies the following:

- To set a (slow) block device as *backing device*, use the bcache\_backing\_for element.
- To set a (fast) block device as *caching device*, use the bcache\_caching\_for element. You can use the same device to speed up the access to several drives.
- To specify the layout of the bcache device, use a drive section and set the type element to CT\_BCACHE. The layout of the bcache device may contain partitions.

#### EXAMPLE 4.19: bcache DEFINITION

```

<partitioning config:type="list">
  <drive>

```

```

<device>/dev/sda</device>
<type config:type="symbol">CT_DISK</type>
<use>all</use>
<enable_snapshots config:type="boolean">>true</enable_snapshots>
<partitions config:type="list">
  <partition>
    <filesystem config:type="symbol">btrfs</filesystem>
    <mount>/</mount>
    <create config:type="boolean">>true</create>
    <size>max</size>
  </partition>
  <partition>
    <filesystem config:type="symbol">swap</filesystem>
    <mount>swap</mount>
    <create config:type="boolean">>true</create>
    <size>2GiB</size>
  </partition>
</partitions>
</drive>

<drive>
  <type config:type="symbol">CT_DISK</type>
  <device>/dev/sdb</device>
  <disklabel>msdos</disklabel>
  <use>all</use>
  <partitions config:type="list">
    <partition>
      <!-- It can serve as caching device for several bcachees -->
      <bcache_caching_for config:type="list">
        <listentry>/dev/bcache0</listentry>
      </bcache_caching_for>
      <size>max</size>
    </partition>
  </partitions>
</drive>

<drive>
  <type config:type="symbol">CT_DISK</type>
  <device>/dev/sdc</device>
  <use>all</use>
  <disklabel>msdos</disklabel>
  <partitions config:type="list">
    <partition>
      <!-- It can serve as backing device for one bcache -->
      <bcache_backing_for>/dev/bcache0</bcache_backing_for>
    </partition>
  </partitions>

```

```

</drive>

<drive>
  <type config:type="symbol">CT_BCACHE</type>
  <device>/dev/bcache0</device>
  <bcache_options>
    <cache_mode>writethrough</cache_mode>
  </bcache_options>
  <use>all</use>
  <partitions config:type="list">
    <partition>
      <mount>/data</mount>
      <size>20GiB</size>
    </partition>
    <partition>
      <mount>swap</mount>
      <filesystem config:type="symbol">swap</filesystem>
      <size>1GiB</size>
    </partition>
  </partitions>
</drive>
</partitioning>

```

For the time being, the only supported option in the `bcache_options` section is `cache_mode`, described in the table below.

Attribute	Values	Description
<code>cache_mode</code>	<pre>&lt;cache_mode&gt;writethrough&lt;/cache_mode&gt;</pre>	Cache mode for <code>bcache</code> . Possible values are <code>writethrough</code> , <code>writeback</code> , <code>writearound</code> and <code>none</code> .

#### 4.4.9 Multi-device Btrfs Configuration

Btrfs supports creating a single volume that spans more than one storage device, offering similar features to software RAID implementations such as the Linux kernel's built-in `mdraid` subsystem. *Multi-device Btrfs* offers advantages over some other RAID implementations. For example, you can dynamically migrate a multi-device Btrfs volume from one RAID level to another, RAID levels can be set on a per-file basis, and more. However, not all of these features are fully supported yet in openSUSE Leap 15 SP 2.

With AutoYaST, a multi-device Btrfs can be configured by specifying a drive with the `CT_BTRFS` type. The `device` property is used as an arbitrary name to identify each multi-device Btrfs.

As with RAID, you need to create all block devices first (e.g., partitions, LVM logical volumes, etc.) and assign them to the Btrfs file system you want to create over such block devices.

The following example shows a simple multi-device Btrfs configuration:

#### EXAMPLE 4.20: MULTI-DEVICE BTRFS CONFIGURATION

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <disklabel>none</disklabel>
    <partitions>
      <partition>
        <btrfs_name>root_fs</btrfs_name>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
  <drive>
    <device>/dev/sdb</device>
    <disklabel>gpt</disklabel>
    <partitions>
      <partition>
        <partition_nr>1</partition_nr>
        <size>4gb</size>
        <filesystem>ext4</filesystem>
        <btrfs_name>root_fs</btrfs_name>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
  <drive>
    <device>root_fs</device>
    <type config:type="symbol">CT_BTRFS</type>
    <partitions>
      <partition config:type="list">
        <mount>/</mount>
      </partition>
    </partitions>
    <btrfs_options>
      <raid_level>raid1</raid_level>
      <metadata_raid_level>raid1</metadata_raid_level>
    </btrfs_options>
  </drive>
```

```
</partitioning>
```

The supported data and meta-data RAID levels are: default, single, dup, raid0, raid1, and raid10. By default, file system meta-data is mirrored across two devices and data is striped across all of the devices. If only one device is present, meta-data will be duplicated on that one device.

Keep the following in mind when configuring a multi-device Btrfs file system:

- Devices need to indicate the btrfs\_name property to be included into a multi-device Btrfs file system.
- All Btrfs-specific options are contained in the btrfs\_options resource of a CT\_BTRFS drive.

#### 4.4.10 NFS Configuration

AutoYaST allows to install openSUSE Leap onto *Network File System* (NFS) shares. To do so, you have to create a drive with the CT\_NFS type and provide the NFS share name (SERVER:PATH) as device name. The information relative to the mount point is included as part of its first partition section. Note that for an NFS drive, only the first partition is taken into account.

For more information on how to configure an NFS client and server after the system has been installed, refer to [Section 4.4.10, "NFS Configuration"](#).

##### EXAMPLE 4.21: NFS SHARE DEFINITION

```
<partitioning config:type="list">
  <drive>
    <device>192.168.1.1:/exports/root_fs</device>
    <type config:type="symbol">CT_NFS</type>
    <use>all</use>
    <partitions config:type="list">
      <partition>
        <mount>/</mount>
        <fstopt>nolock</fstopt>
      </partition>
    </partitions>
  </drive>
</partitioning>
```

## 4.5 iSCSI Initiator Overview

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

### EXAMPLE 4.22: 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
<u>initiatorname</u>	<u>InitiatorName</u> is a value from <code>/etc/iscsi/initiatorname.iscsi</code> . In case you have iBFT, this value will be added from there and you are only able to change it in the BIOS setup.
<u>version</u>	Version of the YaST module. Default: 1.0
<u>targets</u>	List of targets. Each entry contains: <u>authmethod</u> Authentication method: None/CHAP <u>portal</u> Portal address <u>startup</u> Value: manual/onboot <u>target</u> Target name <u>iface</u> Interface name



## 4.6 Fibre Channel over Ethernet Configuration (FCoE)

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

### EXAMPLE 4.23: 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
<code>fcoe_cfg</code>	<p><code>DEBUG</code> is used to enable or disable debugging messages from the fcoe service script and fcoemon.</p> <p><code>USE_SYSLOG</code> messages are sent to the system log if set to yes.</p>	yes/no

Attribute	Description	Values
<u>interfaces</u>	List of network cards including the status of VLAN and FCoE configuration.	
<u>service_start</u>	<p>Enable or disable the start of the services <u>fcoe</u> and <u>lldpad</u> boot time.</p> <p>Starting the <u>fcoe</u> service means starting the Fibre Channel over Ethernet service daemon <u>fcoemon</u> which controls the FCoE interfaces and establishes a connection with the <u>lldpad</u> daemon.</p> <p>The <u>lldpad</u> service provides the Link Layer Discovery Protocol agent daemon <u>lldpad</u>, which informs <u>fcoemon</u> about DCB (Data Center Bridging) features and configuration of the interfaces.</p>	yes/no

## 4.7 Country Settings

Language, timezone, and keyboard settings.

### EXAMPLE 4.24: LANGUAGE

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

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

If the configured value for the primary language is unknown, it will be reset to the default, en\_US.

#### EXAMPLE 4.25: TIMEZONE

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

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

#### EXAMPLE 4.26: KEYBOARD

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

Attribute	Description	Values
<u>keymap</u>	Keyboard layout	Keymap-code values or keymap-alias values are valid. A list of available entries can be found in <u>/usr/share/YaST2/data/keyboards.rb</u> . For example, <u>english-us, us, english-uk, uk.</u>

## 4.8 Software

### 4.8.1 Package Selection with Patterns and Packages Sections

Patterns or packages are configured like this:

EXAMPLE 4.27: PACKAGE SELECTION IN THE CONTROL FILE WITH PATTERNS AND PACKAGES SECTIONS

```
<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>
```



#### Note: Package and Pattern Names

The values are real package or pattern names. If the package name has been changed because of an upgrade, you will need to adapt these settings too.

## 4.8.2 Deploying Images

You can use images during installation to speed up the installation.

### EXAMPLE 4.28: ACTIVATING IMAGE DEPLOYMENT

```
<!-- note! this is not in the software section! -->
<deploy_image>
  <image_installation config:type="boolean">false</image_installation>
</deploy_image>
```

## 4.8.3 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 4.29: CREATING A PACKAGE DATABASE WITH THE ADDITIONAL PACKAGE INST-SOURCE-UTILS.RPM

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

```
tux > cp /tmp/inst-source-utils-2016.7.26-1.2.noarch.rpm /usr/local/DVDs/LATEST/
suse/noarch
tux > cd /usr/local/DVDs/LATEST/suse
tux > 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 4.30: `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/configs">
  <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>
```

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

Even repositories which do not have any product or module information can be added during the installation. These are called other add-ons.

EXAMPLE 4.31: **ADDING THE SDK EXTENSION AND A USER DEFINED REPOSITORY**

```
<add-on>
  <add_on_products config:type="list">
    <listentry>
      <media_url>cd:///sdk</media_url>
      <product>sle-sdk</product>
      <alias>SLE 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_others config:type="list">
    <listentry>
```

```

    <media_url>https://download.opensuse.org/repositories/YaST:/Head/
openSUSE_Leap_15.2/</media_url>
    <alias>yast2_head</alias>
    <priority config:type="integer">30</priority>
    <name>Latest YaST2 packages from OBS</name>
  </listentry>
</add_on_others>
</add-on>

```

The add\_on\_others and add\_on\_products sections support the same values:

Attribute	Values
<u>media_url</u>	<p>Product URL. Can have the prefix <u>cd:///</u>, <u>http://</u>, <u>ftp://</u>, etc. This entry is mandatory.</p> <p>If you use a multi-product medium such as the SUSE Linux Enterprise Packages DVD, then the URL path should point to the root directory of the multi-product medium. The specific product directory is selected using the <u>product_dir</u> value (see below).</p>
<u>product</u>	<p>Internal product name if the add-on is a product. The command <b><u>zypper products</u></b> shows the names of installed products.</p>
<u>alias</u>	<p>Repository alias name. Defined by the user.</p>
<u>product_dir</u>	<p>Optional subpath. This should only be used for multi-product media such as the SUSE Linux Enterprise Packages DVD.</p>
<u>priority</u>	<p>Sets the repository libzypp priority. Priority of 1 is the highest. The higher the number, the lower the priority. Default is 99.</p>
<u>ask_on_error</u>	<p>AutoYaST can ask the user to make add-on products, modules or extensions available instead of reporting a time-out error when</p>

Attribute	Values
	no repository can be found at the given location. Set <code>ask_on_error</code> to <code>true</code> (the default is <code>false</code> ).
<code>confirm_license</code>	The user needs to confirm the license. Default is <code>false</code> .
<code>name</code>	Repository name. The command <code>zypper lr</code> 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
<code>accept_unsigned_file</code>	<p>If set to <code>true</code>, AutoYaST will accept unsigned files like the content file.</p> <pre>&lt;accept_unsigned_file   config:type="boolean"</pre>



Attribute	Values
	<pre data-bbox="805 224 1418 280">&gt;true&lt;/accept_unsigned_file&gt;</pre>
<u>accept_file_without_checksum</u>	<p data-bbox="805 309 1418 387">If set to <u>true</u>, AutoYaST will accept files without a checksum in the content file.</p> <pre data-bbox="805 427 1418 551">&lt;accept_file_without_checksum   config:type="boolean" &gt;true&lt;/accept_file_without_checksum&gt;</pre>
<u>accept_verification_failed</u>	<p data-bbox="805 591 1418 719">If set to <u>true</u>, AutoYaST will accept signed files even when the verification of the signature failed.</p> <pre data-bbox="805 763 1418 887">&lt;accept_verification_failed   config:type="boolean" &gt;true&lt;/accept_verification_failed&gt;</pre>
<u>accept_unknown_gpg_key</u>	<p data-bbox="805 922 1418 1050">If set to <u>true</u>, AutoYaST will accept new GPG keys of the installation sources, for example the key used to sign the content file.</p> <pre data-bbox="805 1095 1418 1218">&lt;accept_unknown_gpg_key   config:type="boolean" &gt;true&lt;/accept_unknown_gpg_key&gt;</pre>
<u>accept_non_trusted_gpg_key</u>	<p data-bbox="805 1254 1418 1332">Set this option to <u>true</u> to accept known keys you have not yet trusted.</p> <pre data-bbox="805 1377 1418 1500">&lt;accept_non_trusted_gpg_key   config:type="boolean" &gt;true&lt;/accept_non_trusted_gpg_key&gt;</pre>
<u>import_gpg_key</u>	<p data-bbox="805 1536 1418 1664">If set to <u>true</u>, AutoYaST will accept and import new GPG keys on the installation source in its database.</p> <pre data-bbox="805 1709 1418 1800">&lt;import_gpg_key config:type="boolean" &gt;true&lt;/import_gpg_key&gt;</pre>

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
<u>accept_unsigned_file</u>	<p>If set to <u>true</u>, AutoYaST will accept unsigned files like the content file for this add-on product.</p> <pre data-bbox="810 645 1414 779" style="border: 1px solid #ccc; padding: 5px;"> &lt;accept_unsigned_file   config:type="boolean" &gt;true&lt;/accept_unsigned_file&gt;</pre>
<u>accept_file_without_checksum</u>	<p>If set to <u>true</u>, AutoYaST will accept files without a checksum in the content file for this add-on.</p> <pre data-bbox="810 976 1414 1111" style="border: 1px solid #ccc; padding: 5px;"> &lt;accept_file_without_checksum   config:type="boolean" &gt;true&lt;/accept_file_without_checksum&gt;</pre>
<u>accept_verification_failed</u>	<p>If set to <u>true</u>, AutoYaST will accept signed files even when the verification of the signature fails.</p> <pre data-bbox="810 1308 1414 1442" style="border: 1px solid #ccc; padding: 5px;"> &lt;accept_verification_failed   config:type="boolean" &gt;true&lt;/accept_verification_failed&gt;</pre>
<u>accept_unknown_gpg_key</u>	<p>If <u>all</u> is set to <u>true</u>, AutoYaST will accept new GPG keys on the installation source.</p> <pre data-bbox="810 1592 1414 1727" style="border: 1px solid #ccc; padding: 5px;"> &lt;accept_unknown_gpg_key&gt;   &lt;all config:type="boolean"&gt;true&lt;/all&gt; &lt;/accept_unknown_gpg_key&gt;</pre> <p>Otherwise you can define single keys too.</p> <pre data-bbox="810 1821 1414 1859" style="border: 1px solid #ccc; padding: 5px;"> &lt;accept_unknown_gpg_key&gt;</pre>

Attribute	Values
	<pre data-bbox="818 228 1374 398">&lt;all config:type="boolean"&gt;false&lt;/all&gt; &lt;keys config:type="list"&gt;   &lt;keyid&gt;3B3011B76B9D6523&lt;/keyid&gt; &lt;/keys&gt; &lt;/accept_unknown_gpg_key&gt;</pre>
<p data-bbox="186 456 608 488"><u>accept_non_trusted_gpg_key</u></p>	<p data-bbox="809 456 1374 533">This means, the key is known, but it is not trusted by you.</p> <p data-bbox="809 562 1241 593">You can trust all keys by adding:</p> <pre data-bbox="818 640 1358 741">&lt;accept_non_trusted_gpg_key&gt;   &lt;all config:type="boolean"&gt;&gt;true&lt;/all&gt; &lt;/accept_non_trusted_gpg_key&gt;</pre> <p data-bbox="809 788 1206 819">Or you can trust specific keys:</p> <pre data-bbox="818 869 1370 1077">&lt;accept_non_trusted_gpg_key&gt;   &lt;all config:type="boolean"&gt;&gt;false&lt;/all&gt;   &lt;keys config:type="list"&gt;     &lt;keyid&gt;3B3011B76B9D6523&lt;/keyid&gt;   &lt;/keys&gt; &lt;/accept_non_trusted_gpg_key&gt;</pre>
<p data-bbox="186 1133 416 1164"><u>import_gpg_key</u></p>	<p data-bbox="809 1133 1385 1254">If <u>all</u> is set to <u>true</u>, AutoYaST will accept and import all new GPG keys on the installation source into its database.</p> <pre data-bbox="818 1305 1358 1406">&lt;import_gpg_key&gt;   &lt;all config:type="boolean"&gt;&gt;true&lt;/all&gt; &lt;/import_gpg_key&gt;</pre> <p data-bbox="809 1453 1329 1485">This can be done for specific keys only:</p> <pre data-bbox="818 1534 1370 1742">&lt;import_gpg_key&gt;   &lt;all config:type="boolean"&gt;&gt;false&lt;/all&gt;   &lt;keys config:type="list"&gt;     &lt;keyid&gt;3B3011B76B9D6523&lt;/keyid&gt;   &lt;/keys&gt; &lt;/import_gpg_key&gt;</pre>

## 4.8.4 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 4.32: KERNEL SELECTION IN THE CONTROL FILE

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

## 4.8.5 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 openSUSE Leap. The following example shows how this can be done:

### EXAMPLE 4.33: 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 4.8.1, “Package Selection with Patterns and Packages Sections”](#)). 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.
```

## 4.8.6 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.

## 4.8.7 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>
```

## 4.8.8 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>
```

## 4.8.9 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>
```

## 4.9 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 4.34: 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
<u>stop_on_solver_conflict</u>	Halt installation if there are package dependency issues.	
<u>modified</u>	Create backup of modified files.	
<u>sysconfig</u>	Create backup of <u>/etc/sysconfig</u> directory.	
<u>remove_old</u>	Remove backups from previous updates.	

To start the AutoYaST upgrade mode, you need:

PROCEDURE 4.1: STARTING AUTOYAST IN OFFLINE UPGRADE MODE

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

PROCEDURE 4.2: STARTING AUTOYAST IN ONLINE UPGRADE MODE

1. Boot the system from the installation media.

2. Select the Installation menu item.
3. On the command line, set netsetup=dhcp autoupgrade=1 autoyast=http://192.169.3.1/autoyast.xml.  
Here, network will be set up via DHCP.
4. Press  to start the upgrade process.

## 4.10 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 and how to start them.

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.

To specify the set of services that should be started on boot, use the enable and disable lists. To start a service, add its name to the enable list. To make sure that the service is not started on boot, add it to the disable list.

If a service is not listed as enabled or disabled, a default setting is used. The default setting may be either disabled or enabled.

Finally, some services like cups support on-demand activation (socket activated services). If you want to take advantage of such a feature, list the names of those services in the on\_demand list instead of enable.

### EXAMPLE 4.35: CONFIGURING SERVICES AND TARGETS

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



## 4.11 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).



### Note: The `linuxrc` program

For a detailed description of how `linuxrc` works and its keywords, see [Appendix C, Advanced `linuxrc` Options](#).

AutoYaST settings have higher priority than any configuration files already present. YaST will write `ifcfg-*` files based on the entries in the control file without removing old ones. If there is an empty or absent DNS and routing section, YaST will keep any pre-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, `<networking>` is used to store the whole network configuration.

#### EXAMPLE 4.36: 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>
</dns>
<interfaces config:type="list">
  <interface>
    <bootproto>dhcp</bootproto>
    <name>eth0</name>
    <startmode>auto</startmode>
  </interface>
</interfaces>
<ipv6 config:type="boolean">true</ipv6>
<keep_install_network config:type="boolean">>false</keep_install_network>
#false means use Wicked, true means use NetworkManager
<managed config:type="boolean">>false</managed>
<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>192.168.2.1/24</destination>
      <name>eth0</name>
      <extrapara>foo</extrapara>
      <gateway>-</gateway>
    </route>
    <route>
      <destination>default</destination>
      <name>eth0</name>
      <gateway>192.168.1.1</gateway>
    </route>
    <route>

```

```

    <destination>default</destination>
    <name>lo</name>
    <gateway>192.168.5.1</gateway>
  </route>
</routes>
</routing>
</networking>

```

As shown in the example above, the `<networking>` section can be composed of a few subsections:

- `interfaces` describes the configuration of the network interfaces, including their IP addresses, how they are started, etc.
- `dns` specifies DNS related settings, such as the host name, the list of name servers, etc.
- `routing` defines the routing rules.
- `s390-devices` covers s390-specific device settings.
- `net-udev` enumerates the udev rules used to set persistent names.

#### EXAMPLE 4.37: 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>192.168.1.100</ipaddr>
    <netmask>255.255.255.0</netmask>
    <network>192.168.1.0</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>`

## 4.11.1 Interfaces

The `interfaces` section allows the user to define the configuration of interfaces, including how they are started, their IP addresses, networks, and more. The following elements must be enclosed in `<interfaces>...</interfaces>` tags.

Element	Description	Comment
<u>bootproto</u>	<p>Boot protocol used by the interface. Possible values:</p> <ul style="list-style-type: none"><li>• <u>static</u> for statically assigned addresses. It is required to specify the IP using the <u>ipaddr</u> element.</li><li>• <u>dhcp4</u>, <u>dhcp6</u> or <u>dhcp</u> for setting the IP address with DHCP (IPv4, IPv6 or any).</li><li>• <u>dhcp+autoip</u> to get the IPv4 configuration from <i>Zeroconf</i> and get IPv6 from DHCP.</li><li>• <u>autoip</u> to get the IPv4 configuration from <i>Zeroconf</i>.</li></ul>	Required.

Element	Description	Comment
	<ul style="list-style-type: none"> <li>• <u>ibft</u> to get the IP address using the iBFT protocol.</li> <li>• <u>none</u> to skip setting an address. This value is used for bridges and bonding slaves.</li> </ul>	
<u>broadcast</u>	Broadcast IP address.	Used only with <u>static</u> boot protocol.
<u>device</u>	Device name.	Deprecated. Use <u>name</u> instead.
<u>name</u>	Device name, for example: <u>eth0</u> .	Required.
<u>ipaddr</u>	IP address assigned to the interface.	Used only with <u>static</u> boot protocol. It can include a network prefix, for example: <u>192.168.1.1/24</u> .
<u>remote_ipaddr</u>	Remote IP address for point-to-point connections.	Used only with <u>static</u> boot protocol.
<u>netmask</u>	Network mask, for example: <u>255.255.255.0</u> .	Deprecated. Use <u>prefixlen</u> instead or include the network prefix in the <u>ipaddr</u> element.
<u>network</u>	Network IP address.	Deprecated. Use <u>ipaddr</u> with <u>prefixlen</u> instead.
<u>prefixlen</u>	Network prefix, for example: <u>24</u> .	Used only with <u>static</u> boot protocol.

Element	Description	Comment
<u>startmode</u>	<p>When to bring up an interface. Possible values are:</p> <ul style="list-style-type: none"> <li>• <u>hotplug</u> when the device is plugged in. Useful for USB network cards, for example.</li> <li>• <u>auto</u> when the system boots. <u>onboot</u> is a deprecated alias.</li> <li>• <u>ifplugd</u> when the device is managed by the <u>ifplugd</u> daemon.</li> <li>• <u>manual</u> when the device is supposed to be started manually.</li> <li>• <u>nfsroot</u> when the device is needed to mount the root file system, for example, when <u>/</u> is on an NFS volume.</li> <li>• <u>off</u> to never start the device.</li> </ul>	
<u>ifplugd_priority</u>	Priority for <u>ifplugd</u> daemon. It determines in which order the devices are activated.	Used only with <u>ifplugd</u> start mode.
<u>usercontrol</u>	Parameter is no longer used.	Deprecated.

Element	Description	Comment
<u>bonding_slaveX</u>	Name of the bonding device.	Required for bonding devices. <u>X</u> is replaced by a number starting from 0, for example <u>bonding_slave0</u> . Each slave needs to have a unique number.
<u>bonding_module_opts</u>	Options for bonding device.	Used only with <u>bond</u> device.
<u>mtu</u>	Maximum transmission unit for the interface.	Optional.
<u>ethtool_options</u>	Ethtool options during device activation.	Optional.
<u>zone</u>	Firewall zone name which the interface is assigned to.	Optional.
<u>vlan_id</u>	Identifier used for this VLAN.	Used only with a <u>vlan</u> device.
<u>etherdevice</u>	Device to which VLAN is attached.	Used only with a <u>vlan</u> device and required for it.
<u>bridge</u>	<u>yes</u> if interface is a bridge.	Deprecated. It is inferred from other attributes.
<u>bridge_ports</u>	Space-separated list of bridge ports, for example, <u>eth0</u> <u>eth1</u> .	Used only with a <u>bridge</u> device and required for it.
<u>bridge_stp</u>	Spanning tree protocol. Possible values are <u>on</u> (when enabled) and <u>off</u> (when disabled).	Used only with a <u>bridge</u> device.

Element	Description	Comment
<u>bridge_forward_delay</u>	Forward delay for bridge, for example: <u>15</u> .	Used only with <u>bridge</u> devices. Valid values are between <u>4</u> and <u>30</u> .

## 4.11.2 Persistent Names of Network Interfaces

The `net-udev` element allows to specify a set of udev rules that can be used to assign persistent names to interfaces.

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



### Tip: Handling Collisions in Device Names

When creating an incomplete `udev` rule set, the chosen device name can collide with existing device names. For example, when renaming a network interface to `eth0`, a collision with a device automatically generated by the kernel can occur. AutoYaST tries to handle such cases in a best effort manner and renames colliding devices.

#### EXAMPLE 4.38: ASSIGNING A PERSISTENT NAME USING THE MAC ADDRESS

```
<net-udev config:type="list">
  <rule>
    <name>eth1</name>
    <rule>ATTR{address}</rule>
    <value>52:54:00:68:54:fb</value>
```



```
</rule>
</net-udev>
```

### 4.11.3 Domain Name System

The `dns` section is used to define name-service related settings, such as the host name or name servers.

Element	Description	Comment
<u>hostname</u>	Host name, excluding the domain name part. For example: <code>foo</code> (instead of <code>foo.bar</code> ).	If a host name is not specified and is not taken from a DHCP server (see <code>dhcp_hostname</code> ), AutoYaST will generate a random one.
<u>nameservers</u>	List of name servers. Example: <pre>&lt;nameservers   config:type="list"&gt;   &lt;nameserver&gt;192.168.1.116&lt;/ nameserver&gt;   &lt;nameserver&gt;192.168.1.117&lt;/ nameserver&gt; &lt;/nameservers&gt;</pre>	
<u>searchlist</u>	Search list for host name lookup. <pre>&lt;searchlist   config:type="list"&gt;   &lt;search&gt;example.com&lt;/ search&gt; &lt;/searchlist&gt;</pre>	Optional.
<u>dhcp_hostname</u>	Specifies whether the host name must be taken from DHCP or not.	

Element	Description	Comment
	<pre>&lt;dhcp_hostname   config:type="boolean"&gt;true&lt;/ dhcp_hostname&gt;</pre>	

#### 4.11.4 Routing

The `routing` table allows to specify a list of routes and the packet forwarding settings for IPv4 and IPv6.

Element	Description	Comment
<code>ipv4_forward</code>	Whether IP forwarding must be enabled for IPv4.	Optional.
<code>ipv6_forward</code>	Whether IP forwarding must be enabled for IPv6.	Optional.
<code>routes</code>	List of routes.	Optional.

The following table describes how routes are defined.

Element	Description	Comment
<code>destination</code>	Route destination. An address prefix can be specified, for example: <code>192.168.100.0/24</code> .	Required.
<code>device</code>	Interface associated to the route.	Required.
<code>gateway</code>	Gateway's IP address.	Optional.
<code>netmask</code>	Destination's netmask.	Deprecated. Specifying the prefix as part of the <code>destination</code> value is preferred.

## 4.11.5 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 a colon (preferred) or a space <pre>&lt;chanids&gt;0.0.0700:0.0.0701:0.0.0702&lt;/chanids&gt;</pre>	
layer2	<pre>&lt;layer2 config:type="boolean"&gt;true&lt;/layer2&gt;</pre>	boolean; default: false
portname	QETH port name (deprecated since openSUSE 42.2)	
protocol	CTC / LCS protocol, a small number (as a string) <pre>&lt;protocol&gt;1&lt;/protocol&gt;</pre>	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.

## 4.11.6 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 4.39: 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 NIS Client and Server

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 4.40: 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>

```

## 4.13 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 4.41: 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_ybind config:type="boolean">>false</start_ybind>
  <start_yppasswdd config:type="boolean">>false</start_yppasswdd>
  <start_ypxfrd config:type="boolean">>false</start_ypxfrd>
</nis_server>

```

Attribute	Values	Description
<u>domain</u>	NIS domain name.	

Attribute	Values	Description
<u>maps_to_serve</u>	List of maps which are available for the server.	Values: auto.master, ethers, group, hosts, netgrp, networks, passwd, protocols, rpc, services, shadow
<u>merge_passwd</u>	Select if your passwd file should be merged with the shadow file (only possible if the shadow file exists).	Value: true/false
<u>mingid</u>	Minimum GID to include in the user maps.	
<u>minuid</u>	Minimum UID to include in the user maps.	
<u>nopush</u>	Do not push the changes to slave servers. (Useful if there are none).	Value: true/false
<u>pwd_chfn</u>	YPPWD_CHFN - allow changing the full name	Value: true/false
<u>pwd_chsh</u>	YPPWD_CHSH - allow changing the login shell	Value: true/false
<u>pwd_srcdir</u>	YPPWD_SRCDIR - source directory for passwd data	Default: <u>/etc</u>
<u>securenets</u>	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.

Attribute	Values	Description
		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.
<u>server_type</u>	Select whether to configure the NIS server as a master or a slave or not to configure a NIS server.	Values: master, slave, none
<u>slaves</u>	List of host names to configure as NIS server slaves.	
<u>start_yplib</u>	This host is also a NIS client (only when client is configured locally).	Value: true/false
<u>start_yppasswdd</u>	Also start the password daemon.	Value: true/false
<u>start_ypxfrd</u>	Also start the map transfer daemon. Fast Map distribution; it will speed up the transfer of maps to the slaves.	Value: true/false

## 4.14 Hosts Definition

Using the host resource, you can add more entries to the /etc/hosts file. Already existing entries will not be deleted. The following example shows details.

#### EXAMPLE 4.42: /ETC/HOSTS

```
<host>
  <hosts config:type="list">
    <hosts_entry>
      <host_address>133.3.0.1</host_address>
      <names config:type="list">
        <name>booking</name>
      </names>
    </hosts_entry>
    <hosts_entry>
      <host_address>133.3.0.5</host_address>
      <names config:type="list">
        <name>test-machine</name>
      </names>
    </hosts_entry>
  </hosts>
</host>
```

## 4.15 Windows Domain Membership

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

#### EXAMPLE 4.43: 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
<u>disable_dhcp_hostname</u>	Do not allow DHCP to change the host name.	Value: true/false



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

## 4.16 Samba Server

Configuration of a simple Samba server.

### EXAMPLE 4.44: 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
<u>accounts</u>	List of Samba accounts.	
<u>backend</u>	List of available back-ends	Value: true/false
<u>config</u>	Setting additional user-defined parameters in <u>/etc/samba/smb.conf</u> .	The example shows parameters in the <u>global</u> section of <u>/etc/samba/smb.conf</u> .
<u>service</u>	Samba service starts during boot.	Value: Enabled/Disabled
<u>trustdom/</u>	Trusted Domains.	A map of two maps (keys: <u>establish</u> , <u>revoke</u> ). Each map contains entries in the format <u>key: domainname</u> <u>value: password</u> .
<u>version</u>	Samba version.	Default: 2.11

## 4.17 Authentication Client

The configuration file must be in the JSON format. Verify that both autoyast2 and autoyast2-installation are installed. Use the *Autoinstallation Configuration* module in YaST to generate a valid JSON configuration file. Launch YaST and switch to the *Miscellaneous > Autoinstallation Configuration*. Choose *Network Services > User Logon Management*, click *Edit*, and configure the available settings. Click *OK* when done. To save the generated configuration file, use *File > Save*.



### Tip: Using ldaps://

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

## 4.18 NFS Client and Server

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

From openSUSE Leap 15.2 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.

For more information on how to install openSUSE Leap onto NFS shares, refer to [Section 4.4.10, "NFS Configuration"](#).

### EXAMPLE 4.45: 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 4.46: 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>

```

## 4.19 NTP Client

### ! Important: NTP Client Profile Incompatible

Starting with openSUSE Leap 15, the NTP client profile has a new format and is *not* compatible with previous profiles. You need to update your NTP client profile used in prior openSUSE Leap versions to be compatible with version 15 and newer.

Following is an example of the NTP client configuration:

#### EXAMPLE 4.47: NETWORK CONFIGURATION: NTP CLIENT

```

<ntp-client>
<ntp_policy>auto</ntp_policy> ❶
<ntp_servers config:type="list">
  <ntp_server>
    <address>cz.pool.ntp.org</address> ❷
    <iburst config:type="boolean">>false</iburst> ❸
    <offline config:type="boolean">>false</offline> ❹
  </ntp_server>
</ntp_servers>
<ntp_sync>15</ntp_sync> ❺
</ntp-client>

```

- ❶ The `ntp_policy` takes the same values as the `NETCONFIG_NTP_POLICY` option in `/etc/sysconfig/network/config`. The most common options are 'static' and 'auto' (default). See [man 8 netconfig](#) for more details.
- ❷ URL of the time server or pool of time servers.

- ③ `iburst` speeds up the initial time synchronization for the specific time source after `chronyd` is started.
- ④ When the `offline` option is set to `true` it will prevent the client from polling the time server if it is not available when `chronyd` is started. Polling will not resume until it is started manually with `chronyc online`. This command does not survive a reboot. Setting it to `false` ensures that clients will always attempt to contact the time server, without administrator intervention.
- ⑤ For `ntp_sync`, enter 'systemd' (default) when running an NTP daemon, an *integer* interval in seconds to synchronize using cron, or 'manual' for no automatic synchronization.

The following example illustrates an IPv6 configuration. You may use the server's IP address, host name, or both:

```
<peer>
  <address>2001:418:3ff::1:53</address>
  <comment/>
  <options/>
  <type>server</type>
</peer>

<peer>
  <address>2.pool.ntp.org</address>
  <comment/>
  <options/>
  <type>server</type>
</peer>
```

## 4.20 Mail Server 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 4.48: 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>

```

## 4.21 Apache 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 4.49: 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 overridden 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>
</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 need 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>

```

```

<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 <u>Listen</u> settings
	PORT	port address
	ADDRESS	Network address. All addresses will be taken if this entry is empty.
hosts		List of Hosts configuration
	KEY	Host name; <code>&lt;KEY&gt;main&lt;/KEY&gt;</code> defines the main hosts, for example <code>&lt;KEY&gt;192.168.43.2/secondserver.suse.de&lt;/KEY&gt;</code>
	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 <u>true</u> .
	change	For historical reasons, it is always set to <u>enable</u> .

Element	Description	Comment
version	Version of used Apache server	Only for information. Default 2.9

Element	Description	Comment
service	Enable Apache service	Optional. Default: false



## Note: Firewall

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

## 4.22 Squid Server

Squid is a caching and forwarding Web proxy.

EXAMPLE 4.50: **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>
```

```
</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
<u>acls</u>	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.
<u>http_accesses</u>	In the Access Control table, access can be denied or allowed to ACL Groups.	<p>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.</p> <p>The Access Control table is checked in the order listed here. The first matching entry is used.</p>
<u>http_ports</u>	Define all ports where Squid will listen for clients' HTTP requests.	<p><u>Host</u> can contain a host name or IP address or remain empty.</p> <p><u>transparent</u> disables PMTU discovery when transparent.</p>
<u>refresh_patterns</u>	Refresh patterns define how Squid treats the objects in the cache.	<p>The refresh patterns are checked in the order listed here. The first matching entry is used.</p> <p><u>Min</u> determines how long (in minutes) an object should be considered fresh if no explicit expiry time is given.</p> <p><u>Max</u> is the upper limit of how long objects without an</p>

Attribute	Values	Description
		explicit expiry time will be considered fresh. <u>Percent</u> is the percentage of the object's age (time since last modification). An object without an explicit expiry time will be considered fresh.
<u>settings</u>	Map of all available general parameters with default values.	Use the YaST Squid configuration module to get an overview about possible entries.
<u>service_enabled_on_startup</u>	Squid service start when booting.	Value: true/false

## 4.23 FTP Server

Configure your FTP Internet server settings.

EXAMPLE 4.51: **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>
<TLS>YES</TLS>
<Umask/>
<UmaskAnon/>
<UmaskLocal/>
<VerboseLogging>NO</VerboseLogging>
<VirtualUser>NO</VirtualUser>
</ftp-server>

```

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



<b>Element</b>	<b>Description</b>	<b>Comment</b>
<u>CertFile</u>	DSA certificate to use for SSL-encrypted connections	This option specifies the location of the DSA certificate to use for SSL-encrypted connections.
<u>ChrootEnable</u>	When enabled, local users will by default be placed in a <u>chroot</u> jail in their home directory after login.	Warning: This option has security implications. Values: YES/NO
<u>EnableUpload</u>	If enabled, FTP users can upload.	To allow anonymous users to upload, enable <u>AnonReadOnly</u> . Values: YES/NO
<u>FTPUser</u>	Defines the anonymous FTP user.	
<u>FtpDirAnon</u>	FTP directory for anonymous users.	Specify a directory which is used for anonymous FTP users.
<u>FtpDirLocal</u>	FTP directory for authenticated users.	Specify a directory which is used for FTP authenticated users.
<u>LocalMaxRate</u>	The maximum data transfer rate permitted for local authenticated users.	KB/s
<u>MaxClientsNumber</u>	The maximum number of clients allowed to connect.	

<b>Element</b>	<b>Description</b>	<b>Comment</b>
<u>MaxClientsPerIP</u>	Defines the maximum number of clients for one IP.	This limits the number of clients allowed to connect from a single source Internet address.
<u>MaxIdleTime</u>	The maximum time (timeout) a remote client may wait between FTP commands.	Minutes
<u>PasMaxPort</u>	Maximum value for a port range for passive connection replies.	<u>PassiveMode</u> needs to be set to YES.
<u>PasMinPort</u>	Minimum value for a port range for passive connection replies.	<u>PassiveMode</u> needs to be set to YES.
<u>PassiveMode</u>	Enable Passive Mode	Value: YES/NO
<u>SSL</u>	Security Settings	Disable SSL/TLS: 0; Accept SSL and TLS: 1; Refuse Connections Without SSL/TLS: 2
<u>SSLEnable</u>	If enabled, SSL connections are allowed.	Value: YES/NO
<u>SSLv2</u>	If enabled, SSL version 2 connections are allowed.	Value: YES/NO
<u>SSLv3</u>	If enabled, SSL version 3 connections are allowed.	Value: YES/NO
<u>StartDaemon</u>	How the FTP daemon will be started.	Manually: 0; when booting: 1; via <u>systemd</u> socket: 2

Element	Description	Comment
<u>TLS</u>	If enabled, TLS connections are allowed.	Value: YES/NO
<u>Umask</u>	File creation mask, in the format (umask for files): (umask for directories).	For example <u>177:077</u> if you feel paranoid.
<u>UmaskAnon</u>	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.
<u>UmaskLocal</u>	Umask for authenticated users.	To specify octal values, remember the "0" prefix, otherwise the value will be treated as a base-10 integer.
<u>VerboseLogging</u>	When enabled, all FTP requests and responses are logged.	Value: YES/NO
<u>VirtualUser</u>	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.

## 4.24 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 the systemd socket.

Note that TFTP and FTP are not the same.

EXAMPLE 4.52: 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).

## 4.25 Firstboot Workflow

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

EXAMPLE 4.53: ENABLING FIRSTBOOT WORKFLOW

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

## 4.26 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 4.54: 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>
```

### 4.26.1 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.

## 4.26.2 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.

## 4.26.3 Login Settings

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

## 4.26.4 New user settings (`useradd` settings)

Set the minimum and maximum possible user and group IDs.

## 4.27 Linux Audit Framework (LAF)

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

EXAMPLE 4.55: 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
<u>auditd/flush</u>	Describes how to write the data to disk.	If set to <u>INCREMENTAL</u> the Frequency parameter tells how many records to write before issuing an explicit flush to disk. <u>NONE</u> means: no special effort is made to flush data, <u>DATA</u> : keep data portion synchronized, <u>SYNC</u> : keep data and metadata fully synchronized.
<u>auditd/freq</u>	This parameter tells how many records to write before issuing an explicit flush to disk.	The parameter <u>flush</u> needs to be set to <u>INCREMENTAL</u> .
<u>auditd/log_file</u>	The full path name to the log file.	
<u>auditd/log_fomat</u>	How much information needs to be logged.	Set <u>RAW</u> to log all data (store in a format exactly as the kernel sends it) or <u>NOLOG</u> to discard all audit information instead of writing it to disk (does not affect data sent to the dispatcher).
<u>auditd/max_log_file</u>	How much information needs to be logged.	Unit: Megabytes
<u>auditd/num_logs</u>	Number of log files.	<u>max_log_file_action</u> needs to be set to <u>ROTATE</u>

Attribute	Values	Description
<u>auditd/</u> <u>max_log_file_action</u>	What happens if the log capacity has been reached.	If the action is set to <u>ROTATE</u> the Number of Log Files specifies the number of files to keep. Set to <u>SYSLOG</u> , the audit daemon will write a warning to the system log. With <u>SUSPEND</u> the daemon stops writing records to disk. <u>IGNORE</u> means do nothing, <u>KEEP_LOGS</u> is similar to <u>ROTATE</u> , but log files are not overwritten.
<u>auditd/name_format</u>	Computer Name Format describes how to write the computer name to the log file.	If <u>USER</u> is set, the user-defined name is used. <u>NONE</u> means no computer name is inserted. <u>HOSTNAME</u> uses the name returned by the 'gethostname' syscall. <u>FQD</u> uses the fully qualified domain name.
<u>rules</u>	Rules for auditctl	You can edit the rules manually, which we only recommend for advanced users. For more information about all options, see <u>man auditctl</u> .



## 4.28 Users and Groups

### 4.28.1 Users

A list of users can be defined in the `<users>` section. To be able to log in, make sure that either the `root` users are set up or `rootpassword` is specified as a `linuxrc` option.

#### EXAMPLE 4.56: 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 4.57: 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
        AAAAB3NzaC1yc2EAAAADAQABAAQDKLtlvnW2vTJpBp3VK91rFsBvpY97NljsVLdgUr1PbZ/
        L51FerQQ+djQ/ivDASQj0+567nMGqfYGFA/De1EGMMEoeShza67qjNi14L1HBGgVojaNajMR/
        NI2d1kDyvsgRy7D7FT5UGGUNT0d1cSD3b85zgwHeYLidgcGIoKeRi7HpVD00TyhwUv4sq3ubrPCWARgPe0LdVFfa9cLC8PTZdxSeKp4j
        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` file by not specifying the respective section in the AutoYaST control file.



### Note: Combine `rootpassword` and Root User Options

It is possible to specify `rootpassword` in `linuxrc` and have a user section for the `root` user. If this section is missing the password, then the password from `linuxrc` will be used. Passwords in profiles take precedence over `linuxrc` passwords.



### 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
<u>username</u>	Text <pre>&lt;username&gt;lukesw&lt;/username&gt;</pre>	Required. It should be a valid user name. Check <a href="#">man 8 useradd</a> if you are not sure.
<u>fullname</u>	Text <pre>&lt;fullname&gt;Tux Torvalds&lt;/fullname&gt;</pre>	Optional. User's full name.
<u>forename</u>	Text <pre>&lt;forename&gt;Tux&lt;/forename&gt;</pre>	Optional. User's forename.
<u>surname</u>	Text <pre>&lt;surname&gt;Skywalker&lt;/surname&gt;</pre>	Optional. User's surname.
<u>uid</u>	Number <pre>&lt;uid&gt;1001&lt;/uid&gt;</pre>	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 <i>Note: Specifying a User ID (uid)</i> for additional information.
<u>gid</u>	Number <pre>&lt;gid&gt;100&lt;/gid&gt;</pre>	Optional. Initial group ID. It must be a unique and non-negative number. Moreover it must refer to an existing group.
<u>home</u>	Path <pre>&lt;home&gt;/home/luke&lt;/home&gt;</pre>	Optional. Absolute path to the user's home directory. By default, <a href="#">/home/username</a>

Attribute	Values	Description
		will be used (for example, <u>alice</u> 's home directory will be <u>/home/alice</u> ).
<u>home_btrfs_subvolume</u>	Boolean <pre>&lt;home_btrfs_subvolume   config:type="boolean"&gt;true&lt;/ home_btrfs_subvolume&gt;</pre>	Optional. Generates the home directory in a Btrfs subvolume. Disabled by default.
<u>shell</u>	Path <pre>&lt;shell&gt;/usr/bin/zsh&lt;/shell&gt;</pre>	Optional. <u>/bin/bash</u> is the default value. If you choose another one, make sure that it is installed (adding the corresponding package to the <u>software</u> section).
<u>user_password</u>	Text <pre>&lt;user_password&gt;some- password&lt;/user_password&gt;</pre>	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 <u>mkpasswd</u> . Enter the password as written in <u>/etc/shadow</u> (second column). To enable or disable the use of encrypted passwords in the profile, see the <u>encrypted</u> parameter.
<u>encrypted</u>	Boolean	Optional. Considered <u>false</u> if not present. Indicates if the user's password in the

Attribute	Values	Description
	<pre data-bbox="598 235 992 353">&lt;encrypted   config:type="boolean"&gt;true&lt;/ encrypted&gt;</pre>	<p data-bbox="1015 235 1410 405">profile is encrypted or not. AutoYaST supports standard encryption algorithms (see <a href="#">man 3 crypt</a>).</p>
<p data-bbox="181 454 464 483"><u>password_settings</u></p>	<p data-bbox="598 454 834 483">Password settings</p> <pre data-bbox="598 526 992 719">&lt;password_settings&gt;   &lt;expire/&gt;   &lt;max&gt;60&lt;/max&gt;   &lt;warn&gt;7&lt;/warn&gt; &lt;/password_settings&gt;</pre>	<p data-bbox="1015 454 1410 1240">Optional. Some password settings can be customized: <u>expire</u> (account expiration date in format <u>YYYY-MM-DD</u>), <u>flag</u> (<u>/etc/shadow</u> flag), <u>inact</u> (number of days after password expiration that account is disabled), <u>max</u> (maximum number of days a password is valid), <u>min</u> (grace period in days until which a user can change password after it has expired) and <u>warn</u> (number of days before expiration when the password change reminder starts).</p>
<p data-bbox="181 1294 432 1323"><u>authorized_keys</u></p>	<p data-bbox="598 1294 903 1323">List of authorized keys</p> <pre data-bbox="598 1366 992 1547">&lt;authorized_keys   config:type="list"&gt;   &lt;listentry&gt;ssh-rsa ...&lt;/ listentry&gt; &lt;/authorized_keys&gt;</pre>	<p data-bbox="1015 1294 1410 1464">A list of authorized keys to be written to <u>\$HOME/.ssh/authorized_keys</u>. See <a href="#">example</a> below.</p>

## 4.28.2 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
<u>group</u>	Text <pre>&lt;group&gt;100&lt;/group&gt;</pre>	Optional. Default initial login group.
<u>groups</u>	Text <pre>&lt;groups&gt;users&lt;/groups&gt;</pre>	Optional. List of additional groups.
<u>home</u>	Path <pre>&lt;home&gt;/home&lt;/home&gt;</pre>	Optional. User's home directory prefix.
<u>expire</u>	Date <pre>&lt;expire&gt;2017-12-31&lt;/expire&gt;</pre>	Optional. Default password expiration date in <code>YYYY-MM-DD</code> format.
<u>inactive</u>	Number <pre>&lt;inactive&gt;3&lt;/inactive&gt;</pre>	Optional. Number of days after which an expired account is disabled.
<u>no_groups</u>	Boolean <pre>&lt;no_groups   config:type="boolean"&gt;true&lt;/ no_groups&gt;</pre>	Optional. Do not use secondary groups.
<u>shell</u>	Path <pre>&lt;shell&gt;/usr/bin/fish&lt;/ shell&gt;</pre>	Default login shell. <code>/bin/bash</code> is the default value. If you choose another one, make sure that it is installed

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

### 4.28.3 Groups

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

EXAMPLE 4.58: **GROUP CONFIGURATION**

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

Attribute	Values	Description
<u>groupname</u>	Text <pre>&lt;groupname&gt;users&lt;/groupname&gt;</pre>	Required. It should be a valid group name. Check <u>man 8 groupadd</u> if you are not sure.

Attribute	Values	Description
<u>gid</u>	Number <pre>&lt;gid&gt;100&lt;/gid&gt;</pre>	Optional. Group ID. It must be a unique and non-negative number.
<u>group_password</u>	Text <pre>&lt;group_password&gt;password&lt;/group_password&gt;</pre>	Optional. The group's password can be written in plain text (not recommended) or in encrypted form. Check the <a href="#">encrypted</a> to select the desired behavior.
<u>encrypted</u>	Boolean <pre>&lt;encrypted config:type="boolean"&gt;true&lt;/encrypted&gt;</pre>	Optional. Indicates if the group's password in the profile is encrypted or not.
<u>userlist</u>	Users list <pre>&lt;userlist&gt;bob,alice&lt;/userlist&gt;</pre>	Optional. A list of users who belong to the group. User names must be separated by commas.

#### 4.28.4 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 4.59: 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
<u>password_less_login</u>	Boolean <pre>&lt;password_less_login   config:type="boolean"&gt;true&lt;/ password_less_login&gt;</pre>	Optional. Enables password-less login. It only affects graphical login.
<u>autologin_user</u>	Text <pre>&lt;autologin_user&gt;alice&lt;/ autologin_user&gt;</pre>	Optional. Enables autologin for the given user.

## 4.29 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)

### 4.29.1 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 modify the storage devices in your pre-scripts. For example, you can create new partitions or change the configuration of certain technologies like multipath. AutoYaST always inspects the storage devices again after executing all the pre-install scripts.



### 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>
```

## 4.29.2 Post-partitioning Scripts

Executed after YaST has done the partitioning and written [/etc/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>
```

### 4.29.3 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>
```

### 4.29.4 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>
```

### 4.29.5 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`.

Init scripts elements must be placed as follows:

```
<scripts>
```

```

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

```

Init scripts are different from the rest of script types because they are not executed by YaST, but after YaST has finished. For this reason, their XML representation is different from other script types.

TABLE 4.1: INIT SCRIPT XML REPRESENTATION

Element	Description	Comment
<u>location</u>	Define a location from where the script gets fetched. Locations can be the same as for the profile (HTTP, FTP, NFS, etc.). <pre> &lt;location &gt;http://10.10.0.1/ myInitScript.sh&lt;/location&gt; </pre>	Either <code>&lt;location&gt;</code> or <code>&lt;source&gt;</code> must be defined.
<u>source</u>	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. <pre> &lt;source&gt; &lt;![CDATA[ echo "Testing the init script" &gt; /tmp/init_out.txt ]]&gt; &lt;/source&gt; </pre>	Either <code>&lt;location&gt;</code> or <code>&lt;source&gt;</code> must be defined.
<u>filename</u>	The file name of the script. It will be stored in a temporary directory under <code>/tmp</code>	Optional in case you only have a single init script. The default name ( <code>init-</code>

Element	Description	Comment
	<pre>&lt;filename&gt;myinitScript5.sh&lt;/filename&gt;</pre>	<u>scripts</u> ) is used in this case. If having specified more than one init script, you must set a unique name for each script.
<u>rerun</u>	<p>Normally, a script is only run once, even if you use <u>ayast_setup</u> to run an XML file multiple times. Change this default behavior by setting this boolean to <u>true</u>.</p> <pre>&lt;rerun   config:type="boolean"&gt;true&lt;/rerun&gt;</pre>	Optional. Default is <u>false</u> (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.

## 4.29.6 Script XML Representation

Most of the XML elements described below can be used for all the script types described above, except for *init scripts*, whose definitions can contain only a subset of these elements. See [Section 4.29.5, "Init Scripts"](#) for further information about them.

TABLE 4.2: SCRIPT XML REPRESENTATION

Element	Description	Comment
<u>location</u>	<p>Define a location from where the script gets fetched. Locations can be the same as for the control file (HTTP, FTP, NFS, etc.).</p> <pre>&lt;location &gt;http://10.10.0.1/myPreScript.sh&lt;/location&gt;</pre>	Either <u>location</u> or <u>source</u> must be defined.

Element	Description	Comment
<u>source</u>	<p>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.</p> <pre>&lt;source&gt; &lt;![CDATA[ echo "Testing the pre script" &gt; /tmp/pre- script_out.txt ]]&gt; &lt;/source&gt;</pre>	Either <u>location</u> or <u>source</u> must be defined.
<u>interpreter</u>	<p>Specify the interpreter that must be used for the script. Supported options are <u>shell</u> and <u>perl</u>.</p> <pre>&lt;interpreter&gt;perl&lt;/interpreter&gt;</pre>	Optional; default is <u>shell</u> .
<u>file name</u>	<p>The file name of the script. It will be stored in a temporary directory under <u>/tmp</u>.</p> <pre>&lt;filename&gt;myPreScript5.sh&lt;/filename&gt;</pre>	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.
<u>feedback</u>	<p>If this boolean is <u>true</u>, 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.</p> <pre>&lt;feedback config:type="boolean"&gt;true&lt;/ feedback&gt;</pre>	Optional; default is <u>false</u> .
<u>feedback_type</u>	<p>This can be <u>message</u>, <u>warning</u> or <u>error</u>. Set the timeout for these pop-ups in the <code>&lt;report&gt;</code> section.</p>	Optional; if missing, an always-blocking pop-up is used.

Element	Description	Comment
	<pre>&lt;feedback_type&gt;warning&lt;/feedback_type&gt;</pre>	
<u>debug</u>	<p>If this is <u>true</u>, every single line of a shell script is logged. Perl scripts are run with warnings turned on.</p> <pre>&lt;debug config:type="boolean"&gt;&gt;true&lt;/debug&gt;</pre>	Optional; default is <u>true</u> .
<u>notification</u>	<p>This text will be shown in a pop-up for the time the script is running in the background.</p> <pre>&lt;notification&gt;Please wait while script is running...&lt;/notification&gt;</pre>	Optional; if not configured, no notification pop-up will be shown.
<u>param-list</u>	<p>It is possible to specify parameters given to the script being called. You may have more than one <u>param</u> 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.</p> <pre>&lt;param-list config:type="list"&gt;   &lt;param&gt;par1&lt;/param&gt;   &lt;param&gt;par2 par3&lt;/param&gt;   &lt;param&gt;"par4.1 par4.2"&lt;/param&gt; &lt;/param-list&gt;</pre>	Optional; if not configured, no parameters get passed to script.
<u>rerun</u>	<p>A script is only run once. Even if you use <u>ayast_setup</u> to run an XML file multiple times, the script is only run once. Change this default behavior by setting this boolean to <u>true</u>.</p> <pre>&lt;rerun config:type="boolean"&gt;&gt;true&lt;/rerun&gt;</pre>	Optional; default is <u>false</u> , meaning that scripts only run once.

Element	Description	Comment
<u>chrooted</u>	<p>During installation, the new system is mounted at <u>/mnt</u>. If this parameter is set to <u>false</u>, AutoYaST does not run <b>chroot</b> and does not install the boot loader at this stage. If the parameter is set to <u>true</u>, AutoYaST performs a <b>chroot</b> into <u>/mnt</u> and installs the boot loader. The result is that to change anything in the newly-installed system, you no longer need to use the <u>/mnt</u> prefix.</p> <pre>&lt;chrooted config:type="boolean"&gt;true&lt;/chrooted&gt;</pre>	Optional; default is <u>false</u> . This option is only available for chroot environment scripts.

## 4.29.7 Script Example

### EXAMPLE 4.60: 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/
configs">
<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
```

## 4.30 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 4.61: 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.

## 4.31 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>`.

It is also possible to specify a local file using the `relurl://` prefix, for example: `<file_location>relurl://path/to/file.conf</file_location>`.

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

### EXAMPLE 4.62: 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 4.63: 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>

```

## 4.32 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
<u>question</u>	The question you want to ask the user. <pre>&lt;question&gt;Enter the LDAP server&lt;/question&gt;</pre>	The default value is the path to the element (the path often looks strange, so we recommend entering a question).
<u>default</u>	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. <pre>&lt;default&gt;dc=suse,dc=de&lt;/default&gt;</pre>	Optional.
<u>help</u>	An optional help text that is shown on the left side of the question.	Optional.

Element	Description	Comment
	<pre>&lt;help&gt;Enter the LDAP server address.&lt;/help&gt;</pre>	
<u>title</u>	<p>An optional title that is shown above the questions.</p> <pre>&lt;title&gt;LDAP server&lt;/title&gt;</pre>	Optional.
<u>type</u>	<p>The type of the element you want to change. Possible values are <u>symbol</u>, <u>boolean</u>, <u>string</u> and <u>integer</u>. The file system in the partition section is a symbol, while the <u>encrypted</u> 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 <u>config:type="..."</u> attribute. You can also use <u>static_text</u> as type. A <u>static_text</u> is a text that does not require any user input and can show information not included in the help text.</p> <pre>&lt;type&gt;symbol&lt;/type&gt;</pre>	Optional. The default is <u>string</u> . If type is <u>symbol</u> , you must provide the selection element too (see below).
<u>password</u>	<p>If this boolean is set to <u>true</u>, a password dialog pops up instead of a simple text</p>	Optional. The default is <u>false</u> .

Element	Description	Comment
	<p>entry. Setting this to <u>true</u> only makes sense if <u>type</u> is string.</p> <pre data-bbox="600 389 992 521"> &lt;password   config:type="boolean"&gt;true&lt;/ password&gt;</pre>	
<p><u>pathlist</u></p>	<p>A list of <u>path</u> 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 <u>&lt;ldap&gt;&lt;ldap_server&gt;</u> section. So to change that value, you need to set the path to <u>ldap,ldap_server</u>.</p> <pre data-bbox="600 1099 992 1375"> &lt;pathlist   config:type="list"&gt;    &lt;path&gt;networking,dns,hostname&lt;/ path&gt;   &lt;path&gt;...&lt;/path&gt; &lt;/pathlist&gt;</pre> <p>To change the password of the first user in the control file, you need to set the path to <u>users,0,user_password</u>. The <u>0</u> indicates the first user in the <u>&lt;users config:type="list"&gt;</u> list of</p>	<p>This information is optional but you should at least provide <u>path</u> or <u>file</u>.</p>

Element	Description	Comment
	<p>users in the control file. <u>1</u> would be the second one, and so on.</p> <pre data-bbox="603 389 992 1137"> &lt;users config:type="list"&gt;   &lt;user&gt;     &lt;username&gt;root&lt;/ username&gt;     &lt;user_password&gt;password to change&lt;/user_password&gt;     &lt;encrypted config:type="boolean"&gt;&gt;false&lt;/ encrypted&gt;   &lt;/user&gt;   &lt;user&gt;     &lt;username&gt;tux&lt;/ username&gt;     &lt;user_password&gt;password to change&lt;/user_password&gt;     &lt;encrypted config:type="boolean"&gt;&gt;false&lt;/ encrypted&gt;   &lt;/user&gt; &lt;/users&gt; </pre>	
<p><u>file</u></p>	<p>You can store the answer to a question in a file, to use it in one of your scripts later. If you ask during <u>stage=initial</u> 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 <u>chrooted=false</u>. Use the command: <b><u>cp /tmp/my_answer /mnt/tmp/.</u></b> The reason is that <u>/tmp</u> in stage 1 is in the RAM disk</p>	<p>This information is optional, but you should at least provide <u>path</u> or <u>file</u>.</p>



Element	Description	Comment
	<p>and will be lost after the reboot, but the installed system is already mounted at <u>/mnt/</u>.</p> <pre data-bbox="600 439 991 533" style="border: 1px solid gray; padding: 5px;"> &lt;file&gt;/tmp/ answer_hostname&lt;/file&gt;</pre>	
stage	<p>Stage configures the installation stage in which the question pops up. You can set this value to <u>cont</u> or <u>initial</u>. <u>initial</u> means the pop-up comes up very early in the installation, shortly after the pre-script has run. <u>cont</u> means, that the dialog with the question comes after the first reboot when the system boots for the very first time. Questions you answer during the <u>initial</u> stage will write their answer into the control file on the hard disk. You should know that if you enter clear text passwords during <u>initial</u>. Of course it does not make sense to ask for the file system to use during the <u>cont</u> phase. The hard disk is already partitioned at that stage and the question will have no effect.</p>	Optional. The default is <u>initial</u> .

Element	Description	Comment
	<pre data-bbox="603 235 987 282">&lt;stage&gt;cont&lt;/stage&gt;</pre>	
<p data-bbox="186 327 336 353"><u>selection</u></p>	<p data-bbox="598 327 992 685">The selection element contains a list of <u>entry</u> elements. Each entry represents a possible option for the user to choose. The user cannot enter a value in a text box, but they can choose from a list of values.</p> <pre data-bbox="603 725 987 1458"> &lt;selection   config:type="list"&gt;   &lt;entry&gt;     &lt;value&gt;       btrfs     &lt;/value&gt;     &lt;label&gt;       Btrfs File System     &lt;/label&gt;   &lt;/entry&gt;   &lt;entry&gt;     &lt;value&gt;       ext3     &lt;/value&gt;     &lt;label&gt;       Extended3 File       System     &lt;/label&gt;   &lt;/entry&gt; &lt;/selection&gt; </pre>	<p data-bbox="1019 327 1385 544">Optional for <u>type=string</u>, not possible for <u>type=boolean</u> and mandatory for <u>type=symbol</u>.</p>
<p data-bbox="186 1507 288 1534"><u>dialog</u></p>	<p data-bbox="598 1507 992 1823">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.</p>	<p data-bbox="1019 1507 1142 1534">Optional.</p>

Element	Description	Comment
	<pre data-bbox="603 235 987 351">&lt;dialog   config:type="integer"&gt;3&lt;/ dialog&gt;</pre>	
<u>element</u>	<p data-bbox="603 398 987 667">You can have more than one question per dialog. To make that possible you need to specify the <u>element-id</u> with an integer. The questions in a dialog are sorted by ID.</p> <pre data-bbox="603 698 987 815">&lt;element   config:type="integer"&gt;1&lt;/ element&gt;</pre>	Optional (see dialog).
<u>width</u>	<p data-bbox="603 871 987 1184">You can increase the default width of the dialog. If there are multiple width specifications per dialog, the largest one is used. The number is roughly equivalent to the number of characters.</p> <pre data-bbox="603 1216 987 1332">&lt;width   config:type="integer"&gt;50&lt;/ width&gt;</pre>	Optional.
<u>height</u>	<p data-bbox="603 1393 987 1706">You can increase the default height of the dialog. If there are multiple height specifications per dialog, the largest one is used. The number is roughly equivalent to the number of lines.</p>	Optional.

Element	Description	Comment
	<pre data-bbox="603 235 987 347">&lt;height   config:type="integer"&gt;15&lt;/ height&gt;</pre>	
<u>frametitle</u>	<p data-bbox="603 398 987 801">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.</p> <pre data-bbox="603 846 987 929">&lt;frametitle&gt;User data&lt;/ frametitle&gt;</pre>	Optional; default is no frame title.
<u>script</u>	<p data-bbox="603 978 987 1193">You can run scripts after a question has been answered. See the table below for detailed instructions about scripts.</p> <pre data-bbox="603 1238 987 1288">&lt;script&gt;...&lt;/script&gt;</pre>	Optional; default is no script.
<u>ok_label</u>	<p data-bbox="603 1332 987 1503">You can change the label on the <i>Ok</i> button. The last element that specifies the label for a dialog wins.</p> <pre data-bbox="603 1547 987 1585">&lt;ok_label&gt;Finish&lt;/ok_label&gt;</pre>	Optional.
<u>back_label</u>	<p data-bbox="603 1639 987 1809">You can change the label on the <i>Back</i> button. The last element that specifies the label for a dialog wins.</p>	Optional.

Element	Description	Comment
	<pre data-bbox="603 235 987 309">&lt;back_label&gt;change values&lt;/back_label&gt;</pre>	
<p data-bbox="186 360 304 389"><u>timeout</u></p>	<p data-bbox="603 360 987 913">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 <i>Ok</i>.</p> <pre data-bbox="603 952 987 1070">&lt;timeout   config:type="integer"&gt;30&lt;/timeout&gt;</pre>	<p data-bbox="1019 360 1369 533">Optional; a missing value is interpreted as <u>0</u>, which means that there is no timeout.</p>
<p data-bbox="186 1120 512 1149"><u>default_value_script</u></p>	<p data-bbox="603 1120 987 1579">You can run scripts to set the default value for a question (see <a href="#">Section 4.32.1, "Default Value Scripts"</a> for detailed instructions about default value scripts). This feature is useful if you can <u>calculate</u> a default value, especially in combination with the <u>timeout</u> option.</p> <pre data-bbox="603 1617 987 1691">&lt;default_value_script&gt;...&lt;/default_value_script&gt;</pre>	<p data-bbox="1019 1120 1406 1149">Optional; default is no script.</p>

## 4.32.1 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
<u>source</u>	The source code of the script. Whatever you <b>echo</b> 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 <b>echo -n</b> to suppress the <code>\n</code> and that you echo reasonable values and not “okay” for a boolean <pre>&lt;source&gt;...&lt;/source&gt;</pre>	This value is required, otherwise nothing would be executed.
<u>interpreter</u>	The interpreter to use. <pre>&lt;interpreter&gt;perl&lt;/interpreter&gt;</pre>	The default value is <u>shell</u> . You can also set <u>/bin/myinterpreter</u> as value.

## 4.32.2 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
<u>file name</u>	The file name of the script. <pre>&lt;filename&gt;my_ask_script.sh&lt;/filename&gt;</pre>	The default is ask_script.sh
<u>source</u>	The source code of the script. Together with <u>rerun_on_error</u> activated, you check the value that was entered for sanity. Your script can create a file <u>/tmp/next_dialog</u> 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). <pre>&lt;source&gt;...&lt;/source&gt;</pre>	This value is required, otherwise nothing would be executed.

Element	Description	Comment
<u>environment</u>	<p>A boolean that passes the value of the answer to the question as an environment variable to the script. The variable is named <u>VAL</u>.</p> <pre data-bbox="600 506 991 633" style="border: 1px solid gray; padding: 5px;"> &lt;environment   config:type="boolean"&gt;true&lt;/environment&gt;</pre>	Optional. Default is <u>false</u> .
<u>feedback</u>	<p>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.</p> <pre data-bbox="600 1025 991 1153" style="border: 1px solid gray; padding: 5px;"> &lt;feedback   config:type="boolean"&gt;true&lt;/feedback&gt;</pre>	Optional, default is <u>false</u> .
<u>debug</u>	<p>A boolean that turns on debugging for the script execution.</p> <pre data-bbox="600 1361 991 1489" style="border: 1px solid gray; padding: 5px;"> &lt;debug   config:type="boolean"&gt;true&lt;/debug&gt;</pre>	Optional, default is <u>true</u> . This value needs <u>feedback</u> to be turned on, too.
<u>rerun_on_error</u>	<p>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</p>	Optional, default is <u>false</u> . This value should be used together with the feedback option.



Element	Description	Comment
	<p>in the script and display an error with the <u>feedback</u> option.</p> <pre>&lt;rerun_on_error   config:type="boolean"&gt;true&lt;/rerun_on_error&gt;</pre>	

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">ext4</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>
</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>

```

## 4.33 Kernel Dumps

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 4.64: 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>
```

### 4.33.1 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 4.65: 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
<u>add_crash_kernel</u>	Set to <code>true</code> if memory should be reserved and Kdump enabled. <pre>&lt;add_crash_kernel   config:type="boolean"&gt;true&lt;/ add_crash_kernel&gt;</pre>	required
<u>crash_kernel</u>	Use the syntax of the <code>crashkernel</code> command line as discussed above. <pre>&lt;crash_kernel&gt;256M:64M&lt;/ crash_kernel&gt;</pre> A list of values is also supported. <pre>&lt;crash_kernel   config:type="list"&gt;   &lt;listentry&gt;72M,low&lt;/ listentry&gt;   &lt;listentry&gt;256M,high&lt;/ listentry&gt; &lt;/crash_kernel&gt;</pre>	required

### 4.33.2 Dump Saving

This section describes where and how crash dumps will be stored.

### 4.33.2.1 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/SMB 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).

### 4.33.2.2 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.

### 4.33.2.3 Summary

TABLE 4.7: DUMP TARGET SETTINGS: XML REPRESENTATION

Element	Description	Comment
<u>KDUMP_SAVEDIR</u>	<p>A URL that specifies the target to which the dump and related files will be saved.</p> <pre>&lt;KDUMP_SAVEDIR&gt;file:///var/crash/&lt;/KDUMP_SAVEDIR&gt;</pre>	required
<u>KDUMP_COPY_KERNEL</u>	<p>Set to <code>true</code>, if not only the dump should be saved to <u>KDUMP_SAVEDIR</u> but also the kernel and its debugging information (if installed).</p> <pre>&lt;KDUMP_COPY_KERNEL&gt;false&lt;/KDUMP_COPY_KERNEL&gt;</pre>	optional
<u>KDUMP_FREE_DISK_SIZE</u>	<p>Disk space in megabytes that must remain free after saving the dump. If not enough space is available, the dump will not be saved.</p> <pre>&lt;KDUMP_FREE_DISK_SIZE&gt;64&lt;/KDUMP_FREE_DISK_SIZE&gt;</pre>	optional
<u>KDUMP_KEEP_OLD_DUMPS</u>	<p>The number of dumps that are kept (not deleted) if <u>KDUMP_SAVEDIR</u> points to a local directory. Specify 0 if you do not want any dumps to be automatically deleted,</p>	optional



Element	Description	Comment
	specify -1 if all dumps except the current one should be deleted. <pre>&lt;KDUMP_KEEP_OLD_DUMPS&gt;4&lt;/KDUMP_KEEP_OLD_DUMPS&gt;</pre>	

### 4.33.3 E-Mail Notification

Configure e-mail notification 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
<code>KDUMP_NOTIFICATION_TO</code>	Exactly one e-mail address to which the e-mail should be sent. Additional recipients can be specified in <code>KDUMP_NOTIFICATION_CC</code> . <pre>&lt;KDUMP_NOTIFICATION_TO&gt;tux@example.com&lt;/KDUMP_NOTIFICATION_TO&gt;</pre>	optional (notification disabled if empty)
<code>KDUMP_NOTIFICATION_CC</code>	Zero, one or more recipients that are in the cc line of the notification e-mail. <pre>&lt;KDUMP_NOTIFICATION_CC</pre>	optional

Element	Description	Comment
	<pre>&gt;wilber@example.com geeko@example.com&lt;/ KDUMP_NOTIFICATION_CC&gt;</pre>	
<u>KDUMP_SMTP_SERVER</u>	<p>Host name of the SMTP server used for mail delivery. SMTP authentication is supported (see <u>KDUMP_SMTP_USER</u> and <u>KDUMP_SMTP_PASSWORD</u>) but TLS/SSL are not.</p> <pre>&lt;KDUMP_SMTP_SERVER&gt;email.suse.de&lt;/ KDUMP_SMTP_SERVER&gt;</pre>	optional (notification disabled if empty)
<u>KDUMP_SMTP_USER</u>	<p>User name used together with <u>KDUMP_SMTP_PASSWORD</u> for SMTP authentication.</p> <pre>&lt;KDUMP_SMTP_USER&gt;bwalle&lt;/ KDUMP_SMTP_USER&gt;</pre>	optional
<u>KDUMP_SMTP_PASSWORD</u>	<p>Password used together with <u>KDUMP_SMTP_USER</u> for SMTP authentication.</p> <pre>&lt;KDUMP_SMTP_PASSWORD&gt;geheim&lt;/ KDUMP_SMTP_PASSWORD&gt;</pre>	optional

#### 4.33.4 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 vmlinux 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
<code>KDUMP_KERNELVER</code>	Version string for the kernel used for Kdump. Leave it empty to use the auto-detection mechanism (strongly recommended).  <pre>&lt;KDUMP_KERNELVER &gt;2.6.27-default&lt;/ KDUMP_KERNELVER&gt;</pre>	optional (auto-detection if empty)
<code>KDUMP_COMMANDLINE_APPEND</code>	Additional command line parameters for the Kdump kernel.  <pre>&lt;KDUMP_COMMANDLINE_APPEND &gt;console=ttyS0,57600&lt;/ KDUMP_COMMANDLINE_APPEND&gt;</pre>	optional
<code>KDUMP_Command Line</code>	Overwrite the automatically generated Kdump command line. Use with care. Usually, <code>KDUMP_COMMANDLINE_APPEND</code> should suffice.  <pre>&lt;KDUMP_COMMANDLINE_APPEND &gt;root=/dev/sda5 maxcpus=1 irqpoll&lt;/ KDUMP_COMMANDLINE&gt;</pre>	optional

## 4.33.5 Expert Settings

TABLE 4.10: EXPERT SETTINGS: XML REPRESENTATIONS

Element	Description	Comment
<u>KDUMP_IMMEDIATE_REBOOT</u>	<p><u>true</u> if the system should be rebooted automatically after the dump has been saved, <u>false</u> otherwise. The default is to reboot the system automatically.</p> <pre>&lt;KDUMP_IMMEDIATE_REBOOT&gt;true&lt;/KDUMP_IMMEDIATE_REBOOT&gt;</pre>	optional
<u>KDUMP_VERBOSE</u>	<p>Bitmask that specifies how verbose the Kdump process should be. Read <code>kdump(5)</code> for details.</p> <pre>&lt;KDUMP_VERBOSE&gt;3&lt;/KDUMP_VERBOSE&gt;</pre>	optional
<u>KEXEC_OPTIONS</u>	<p>Additional options that are passed to <code>kexec</code> when loading the Kdump kernel. Normally empty.</p> <pre>&lt;KEXEC_OPTIONS&gt;--noio&lt;/KEXEC_OPTIONS&gt;</pre>	optional

## 4.34 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
<u>chroot</u>	0 / 1	The DNS server must be jailed in a chroot.
<u>start_service</u>	0 / 1	Bind is enabled (executed on system start).
<u>use_ldap</u>	0 / 1	Store the settings in LDAP instead of native configuration files.

EXAMPLE 4.66: 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
<u>logging</u>	Options of the DNS server logging.
<u>options</u>	Bind options like the files and directories to use, the list of forwarders and other configuration settings.
<u>zones</u>	List of DNS zones known by the server, including all the settings, records and SOA records.

EXAMPLE 4.67: 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>
```

## 4.35 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
<u>chroot</u>	0 / 1	A value of 1 means that the DHCP server must be jailed in a chroot.
<u>start_service</u>	0 / 1	Set this to 1 to enable the DHCP server (that is, run it on system startup).
<u>use_ldap</u>	0 / 1	If set to 1, the settings will be stored in LDAP instead of native configuration files.
<u>other_options</u>	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 <code>dhcpcd</code> manual page. If left blank, default values will be used.

Element	Value	Description
<u>allowed_interfaces</u>	List	List of network cards in which the DHCP server will be operating. See the example below for the exact format.
<u>settings</u>	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 <u>children</u> , <u>parent_id</u> and <u>parent_type</u> properties are used to represent that nesting. See the example below for the exact format.

#### EXAMPLE 4.68: 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>
      </directives>
    </settings_entry>
  </settings>
</dhcp-server>

```



```

</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>

```

## 4.36 Firewall Configuration

SuSEfirewall2 has been replaced by `firewalld` starting with openSUSE Leap 15.0. Profiles using SuSEfirewall2 properties will be translated to `firewalld` profiles. However, not all profile properties can be converted. For details about `firewalld`, refer to *Book "Security and Hardening Guide", Chapter 25 "Masquerading and Firewalls", Section 25.4 "firewalld"*.



### Important: Limited Backward Compatibility with SuSEFirewall2 Based Profiles

The use of SuSEFirewall2 based profiles will be only partially supported as many options are not valid in `firewalld` and some missing configuration could affect your network security.

## 4.36.1 General Firewall Configuration

In `firewalld` the general configuration only exposes a few properties and most of the configuration is done by zones.

Attribute	Value	Description
<code>start_firewall</code>	Boolean	Whether <code>firewalld</code> should be started right after applying the configuration.
<code>enable_firewall</code>	Boolean	Whether <code>firewalld</code> should be started on every system startup.
<code>default_zone</code>	Zone name	The default zone is used for everything that is not explicitly assigned.
<code>log_denied_packets</code>	Type of dropped packages to be logged	Enable logging of dropped packages for the type selected. Values: <code>off</code> , <code>unicast</code> , <code>multicast</code> , <code>broadcast</code> , <code>all</code> .
<code>name</code>	Identifier of zone	Used to identify a zone. If the zone is not known yet, a new zone will be created.
<code>short</code>	Short summary of zone	Briefly summarizes the purpose of the zone. Ignored for already existing zones. If not specified, the name is used.
<code>description</code>	Description of zone	Describes the purpose of the zone. Ignored for already existing zones. If not specified, the name is used.

Attribute	Value	Description
<u>target</u>	Default action	Defines the default action in the zone if no rule matches. Possible values are <u>ACCEPT</u> , <u>%%REJECT%%</u> , <u>DROP</u> and <u>default</u> . If not specified, <u>default</u> is used. For details about values, see <a href="https://firewalld.org/documentation/zone/options.html">https://firewalld.org/documentation/zone/options.html</a> .

## 4.36.2 Firewall Zones Configuration

The configuration of `firewalld` is based on the existence of several zones which define the trust level for a connection, interface or source address. The behavior of each zone can be tweaked in several ways although not all the properties are exposed yet.

Attributes	Value	Description
<u>interfaces</u>	List of interface names	List of interface names assigned to this zone. Interfaces or sources can only be part of one zone.
<u>services</u>	List of services	List of services accessible in this zone.
<u>ports</u>	List of ports	List of single ports or ranges to be opened in the assigned zone.
<u>protocols</u>	List of protocols	List of protocols to be opened or be accessible in the assigned zone.

Attributes	Value	Description
<u>masquerade</u>	Enable masquerade	It will enable or disable network address translation (NAT) in the assigned zone.

### 4.36.3 A Full Example

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

EXAMPLE 4.69: EXAMPLE FIREWALL SECTION

```
<firewall>
  <enable_firewall config:type="true">true</enable_firewall>
  <log_denied_packets>all</log_denied_packets>
  <default_zone>external</default_zone>
  <zones config:type="list">
    <zone>
      <name>public</name>
      <interfaces config:type="list">
        <interface>eth0</interface>
      </interfaces>
      <services config:type="list">
        <service>ssh</service>
        <service>dhcp</service>
        <service>dhcpv6</service>
        <service>samba</service>
        <service>vnc-server</service>
      </services>
      <ports config:type="list">
        <port>21/udp</port>
        <port>22/udp</port>
        <port>80/tcp</port>
        <port>443/tcp</port>
        <port>8080/tcp</port>
      </ports>
    </zone>
    <zone>
      <name>dmz</name>
      <interfaces config:type="list">
        <interface>eth1</interface>
      </interfaces>
    </zone>
  </zones>
</firewall>
```

```
</zone>
</zones>
</firewall>
```

## 4.37 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.

### 4.37.1 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 4.70, "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 4.70: 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 openSUSE Leap 15.2, AutoYaST only supports modifying cupsd.conf since the default settings in cups-files.conf are sufficient for usual printing setups.

## 4.37.2 Sound devices

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

#### EXAMPLE 4.71: 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>

```

## 4.38 Importing SSH Keys and Configuration

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

### EXAMPLE 4.72: 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
<u>import</u>	true / false	SSH keys will be imported. If set to <u>false</u> , nothing will be imported.
<u>copy_config</u>	true / false	Additionally, SSH server configuration will be imported. This setting will not have effect if <u>import</u> is set to <u>false</u> .



Attributes	Value	Description
<u>device</u>	Partition	Partition to import keys and configuration from. If it is not set, the partition which contains the most recently accessed key is used.

## 4.39 Configuration Management

AutoYaST allows delegating part of the configuration to a *configuration management tool* like Salt. AutoYaST takes care of the basic system installation (partitioning, network setup, etc.) and the remaining configuration tasks can be delegated.



### Note: Only Salt is Officially Supported

Although Puppet is mentioned in this document, only Salt is officially supported. Nevertheless, feel free to report any problem you might find with Puppet.

AutoYaST supports two different approaches:

- Using a configuration management server. In this case, AutoYaST sets up a configuration management tool. It connects to a master server to get the instructions to configure the system.
- Getting the configuration from elsewhere (for example, an HTTP server or a flash disk like a USB stick) and running the configuration management tool in stand-alone mode.

### 4.39.1 Connecting to a Configuration Management Server

This approach is especially useful when a configuration management server (a *master* in Salt and Puppet jargon) is already in place. In this case, the hardest part might be to set up a proper authentication mechanism.

Both Salt and Puppet support the following authentication methods:

- Manual authentication on the fly. When AutoYaST starts the client, a new authentication request is generated. The administrator can manually accept this request on the server. AutoYaST will retry the connection. If the key was accepted meanwhile, AutoYaST continues the installation.
- Using a preseed key. Refer to the documentation of your configuration management system of choice to find out how to generate them. Use the `keys_url` option to tell AutoYaST where to look for them.

With the configuration example below, AutoYaST will launch the client to generate the authentication request. It will try to connect up to three times, waiting 15 seconds between each try.

EXAMPLE 4.73: CLIENT/SERVER WITH MANUAL AUTHENTICATION

```
<configuration_management>
  <type>salt</type>
  <master>my-salt-server.example.net</master>
  <auth_attempts config:type="integer">3</auth_attempts>
  <auth_time_out config:type="integer">15</auth_time_out>
</configuration_management>
```

However, with the following example, AutoYaST will retrieve the keys from a flash disk (for example, a USB stick) and will use them to connect to the master server.

EXAMPLE 4.74: CLIENT/SERVER WITH PRESEED KEYS

```
<configuration_management>
  <type>salt</type>
  <master>my-salt-server.example.net</master>
  <keys_url>usb:/</keys_url>
</configuration_management>
```

The table below summarizes the supported options for these scenarios.

Attributes	Value	Description
<code>type</code>	String	Configuration management name. Currently only <code>salt</code> is officially supported.

Attributes	Value	Description
<u>master</u>	String	Host name or IP address of the configuration management server.
<u>auth_attempts</u>	Integer	Maximum attempts to connect to the server. The default is three attempts.
<u>auth_time_out</u>	Integer	Time (in seconds) between attempts to connect to the server. The default is 15 seconds.
<u>keys_url</u>	URL of used key	Path to an HTTP server, hard disk, flash drive or similar with the files <u>default.key</u> and <u>default.pub</u> . This key must be known to the configuration management master.
<u>enable_services</u>	True/False	Enables the configuration management services on the client side after the installation. The default is <u>true</u> .

## 4.39.2 Running in Stand-alone Mode

For simple scenarios, deploying a configuration management server is unnecessary. Instead, use Salt or Puppet in *stand-alone* (or *masterless*) mode.

As there is no server, AutoYaST needs to know where to get the configuration from. Put the configuration into a TAR archive and store it anywhere (for example, on a flash drive, an HTTP/HTTPS server, an NFS/SMB share).

The TAR archive must have the same layout that is expected under `/srv` in a Salt server. This means that you need to place your Salt states in a `salt` directory and your formulas in a separate `formulas` directory.

Additionally, you can have a `pillar` directory containing the pillar data. Alternatively, you can provide that data in a separate TAR archive by using the `pillar_url` option.

EXAMPLE 4.75: **STANDALONE MODE**

```
<configuration_management>
  <type>salt</type>
  <states_url>my-salt-server.example.net</states_url>
  <pillar_url>my-salt-server.example.net</pillar_url>
</configuration_management>
```

Attributes	Value	Description
<code>type</code>	String	Configuration management name. Currently only <code>salt</code> is officially supported.
<code>states_url</code>	URL	Location of the Salt states TAR archive. It may include formulas and pillars. Files must be located in a <code>salt</code> directory.
<code>pillar_url</code>	URL	Location of the TAR archive that contains the pillars.
<code>modules_url</code>	URL	Location of Puppet modules.

### 4.39.3 SUSE Manager Salt Formulas Support

AutoYaST offers support for SUSE Manager Salt Formulas when running in stand-alone mode. In case a formula is found in the states TAR archive, AutoYaST displays a screen which allows the user to select and configure the formulas to apply.

Bear in mind that this feature defeats the AutoYaST purpose of performing an unattended installation, as AutoYaST will wait for the user's input.

# III Managing Mass Installations with Rules and Classes

5 Rules and Classes **198**

## 5 Rules and Classes

Rules and classes allow customizing installations for sets of machines in different ways:

- Rules allow configuring a system depending on its attributes.
- Classes represent configurations for groups of target systems. Classes can be assigned to systems.



### Note: Use `autoyast` Boot Option Only

Rules and classes are only supported by the boot parameter `autoyast=URL`.

`autoyast2=URL` is not supported, because this option downloads a single AutoYaST control file only.

### 5.1 Rule-based Automatic Installation

Rules offer the possibility to configure a system depending on system attributes by merging multiple control files during installation. The rule-based installation is controlled by a rules file. For example, this could be useful to install 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).

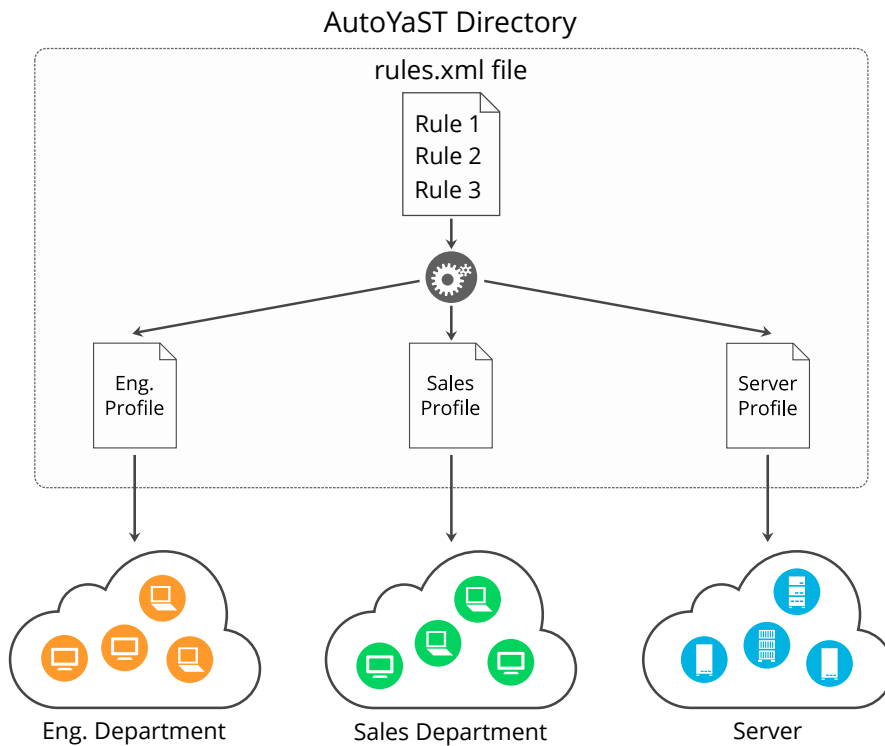


FIGURE 5.1: RULES

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/configs">
  <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/configs">
  <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
<code>hostaddress</code>	IP address of the host	This attribute must always match exactly.
<code>host name</code>	The name of the host	This attribute must always match exactly.
<code>domain</code>	Domain name of host	This attribute must always match exactly.
<code>installed_product</code>	The name of the product to be installed.	This attribute must always match exactly.
<code>installed_product_version</code>	The version of the product to be installed.	This attribute must always match exactly.
<code>network</code>	network address of host	This attribute must always match exactly.
<code>mac</code>	MAC address of host	This attribute must always match exactly (the MAC addresses should have the form <code>0080c8f6484c</code> ).

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

Attribute	Values	Description
<u>custom1-5</u>	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
<u>dialog_nr</u>	<p>All rules with the same <u>dialog_nr</u> are presented in the same pop-up dialog. The same <u>dialog_nr</u> can appear in multiple rules.</p> <pre>&lt;dialog_nr   config:type="integer"&gt;3&lt;/   dialog_nr&gt;</pre>	<p>This element is optional and the default for a missing <u>dialog_nr</u> is always <u>0</u>. To use one pop-up for all rules, you do not need to specify the <u>dialog_nr</u>.</p>
<u>element</u>	<p>Specify a unique ID. Even if you have more than one dialog, you must not use the same id twice. Using id <u>1</u> on dialog 1 and id <u>1</u> on dialog 2 is not supported. (This behavior is contrary</p>	<p>Optional. If left out, AutoYaST adds its own ids internally. Then you cannot specify conflicting rules (see below).</p>

Attribute	Values	Description
	<p>to the <u>ask</u> dialog, where you can have the same ID for multiple dialogs.)</p> <pre data-bbox="600 389 991 521">&lt;element   config:type="integer"&gt;3&lt;/ element&gt;</pre>	
<u>title</u>	<p>Caption of the pop-up dialog</p> <pre data-bbox="600 629 991 723">&lt;title&gt;Desktop Selection&lt;/ title&gt;</pre>	Optional
<u>question</u>	<p>Question shown in the pop-up behind the check box.</p> <pre data-bbox="600 875 991 969">&lt;question&gt;GNOME Desktop&lt;/ question&gt;</pre>	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.
<u>timeout</u>	<p>Timeout in seconds after which the dialog will automatically “press” the okay button. Useful for a non-blocking installation in combination with rules dialogs.</p> <pre data-bbox="600 1384 991 1514">&lt;timeout   config:type="integer"&gt;30&lt;/ timeout&gt;</pre>	Optional. A missing timeout will stop the installation process until the dialog is confirmed by the user.
<u>conflicts</u>	<p>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</p>	<u>optional</u>

Attribute	Values	Description
	<p>deselected and disabled in the pop-up. Take care that you do not create deadlocks.</p> <pre data-bbox="603 392 994 761"> &lt;conflicts   config:type="list"&gt;   &lt;element     config:type="integer"&gt;1&lt;/   element&gt;   &lt;element     config:type="integer"&gt;5&lt;/   element&gt;   ... &lt;/conflicts&gt; </pre>	

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

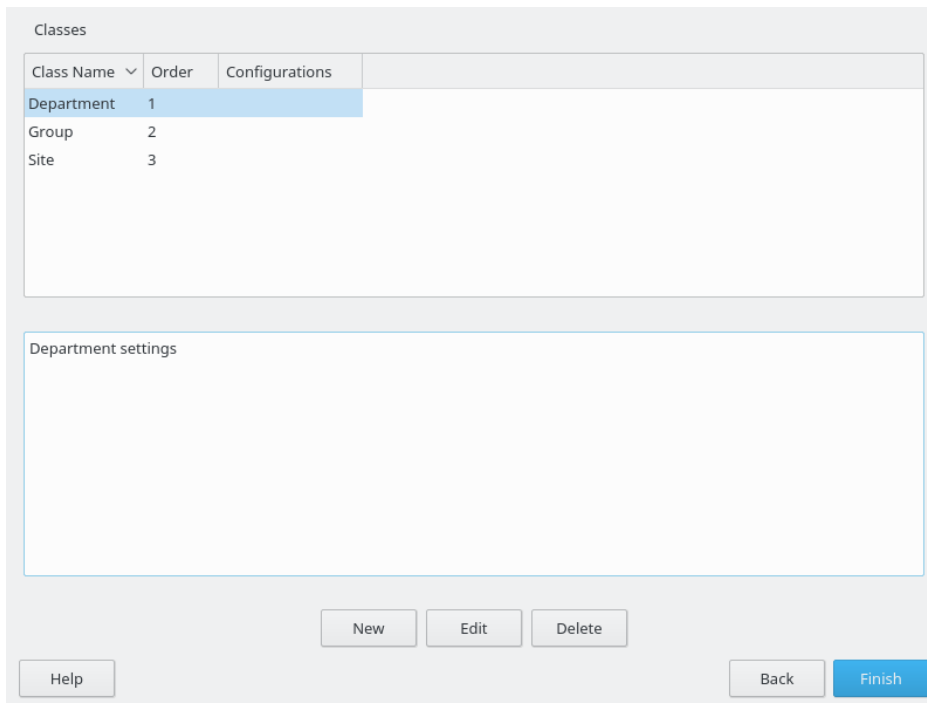


FIGURE 5.2: DEFINING CLASSES

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>
    </partitions>
  </drive>
</partitioning>
```

```

    </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 three 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>
```

# IV Understanding the Auto-Installation Process

6 The Auto-Installation Process 215

## 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 (for example, a USB stick) 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 example, a USB stick)

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: Copying the Installation Media Image to a Removable Flash Disk

Use the following command to copy the contents of the installation image to a removable flash disk.

```
tux > sudo dd if=IMAGE of=FLASH_DISK bs=4M && sync
```

`IMAGE` needs to be replaced with the path to the `SLE-15-SP2-Online-ARCH-GM-media1.iso` or `SLE-15-SP2-Full-ARCH-GM-media1.iso` image file. `FLASH_DISK` needs to be replaced with the flash device. To identify the device, insert it and run:

```
root # grep -Ff <(hwinfo --disk --short) <(hwinfo --usb --short)
disk:
/dev/sdc          General USB Flash Disk
```

Make sure the size of the device is sufficient for the desired image. You can check the size of the device with:

```
root # fdisk -l /dev/sdc | grep -e "^/dev"
/dev/sdc1 *      2048 31490047 31488000 15G 83 Linux
```



In this example, the device has a capacity of 15 GB. The command to use for the SLE-15-SP2-Full-ARCH-GM-media1.iso would be:

```
dd if=SLE-15-SP2-Full-ARCH-GM-media1.iso of=/dev/sdc1 bs=4M && sync
```

The device must not be mounted when running the **dd** command. Note that all data on the partition will be erased!

## 6.2.2 Booting from the SUSE Linux Enterprise Installation Media

You can use the SUSE Linux Enterprise installation media (SLE-15-SP2-Online-ARCH-GM-media1.iso or SLE-15-SP2-Full-ARCH-GM-media1.iso) 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 installation media 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.

If you install via PXE, the installation will run in 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. The `linuxrc` program 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 `linuxrc` configuration file supports multiple keywords. For a detailed description of how `linuxrc` works and other keywords, see [Appendix C, Advanced linuxrc Options](#). Some of the more common ones are:

TABLE 6.1: KEYWORDS FOR `linuxrc`

Keyword	Value
<code>autoupgrade</code>	Initiate an automatic upgrade using AutoYaST; see <a href="#">Section 4.9, “Upgrade”</a> .

Keyword	Value
<u>autoyast</u>	Location of the control file for automatic installation; see <i>AutoYaST Control File Locations</i> for details.
<u>ifcfg</u>	Configure and start the network. Required if the AutoYaST is to be fetched from a remote location. See <i>Section C.3, "Advanced Network Setup"</i> for details.
<u>insmod</u>	Kernel modules to load
<u>install</u>	Location of the installation directory, for example <u>install=nfs://192.168.2.1/CDs/</u> .
<u>instmode</u>	Installation mode, for example <u>nfs</u> , <u>http</u> etc. (not needed if <u>install</u> is set).
<u>rootpassword</u>	Password for root user if not specified in AutoYaST profile
<u>server</u>	Server (NFS) to contact for source directory
<u>serverdir</u>	Directory on NFS Server
<u>y2confirm</u>	Even with <code>&lt; confirm &gt; no &lt;/confirm &gt;</code> 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.

You can pass keywords to `linuxrc` using the kernel command line. This can be done in several ways. You can specify `linuxrc` keywords along with other kernel parameters interactively at boot time, in the usual way. You can also insert kernel parameters into custom network-bootable disk images. It is also possible to configure a DHCP server to pass kernel parameters in combination with Etherboot or PXE.



### Note: Using `autoyast2` Boot Option instead of `autoyast`

The `autoyast2` option is similar to the `autoyast` option, but `linuxrc` parses the provided value and, for example, tries to configure a network when needed. This option is not described in this documentation. For information about differences between the AutoYaST and `linuxrc` URI syntax, see the `linuxrc` appendix: [Appendix C, Advanced `linuxrc` Options](#). AutoYaST's rules and classes are *not* supported.

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 file system. 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, see [Procedure 4.1, "Starting AutoYaST in Offline Upgrade Mode"](#).

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 (for example, a USB stick)
2. The root directory of the installation medium
3. 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 file system, 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 (for example, 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 appropriate installation media of openSUSE Leap (SLE-15-SP2-Online-ARCH-GM-media1.iso or SLE-15-SP2-Full-ARCH-GM-media1.iso). 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 required parameters from an expect script or equivalent.

The following list of scenarios explains how the control file can be supplied:

#### Using the openSUSE Leap installation media

When using the original installation media (SLE-15-SP2-Online-ARCH-GM-media1.iso or SLE-15-SP2-Full-ARCH-GM-media1.iso is needed), the control file needs to be accessible via flash disk (for example, a USB stick) or network:

**Flash Disk (for example, a USB stick).** 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://... Network access needs to be defined using the boot options in linuxrc. This can be done via DHCP: netsetup=dhcp autoyast=http://163.122.3.5/autoyast.xml

#### Using a Custom installation media

In this case, you can include the control file directly on the installation media. 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.

#### 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 (for example, a USB stick). 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.

With all AutoYaST invocation options 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.

The path of this directory needs to end with a `/`.

The files in the directory must not have any extension, for example `.xml`. So the file name needs to be the IP or MAC address only.

```
tux > ls -r control-files
C00002 0080C8F6484C default
```

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

```
tux > /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 openSUSE Leap DVD-ROMs and a flash disk (for example, a USB stick). 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 or as boot options, you may integrate the keywords into the AutoYaST control file. Add an `info_file` section as shown in the example below. This section contains keyword—value pairs, separated by colons, one pair per line.

EXAMPLE 6.2: `linuxrc` OPTIONS IN THE AUTOYAST CONTROL FILE

```
....
<install>
....
  <init>
    <info_file>

install: nfs://192.168.1.1/CDs/full-x86_64
dud: https://example.com/driver_updates/filename.dud
upgrade: 1
textmode: 1
  </info_file>
</init>
.....
</install>
....
```

Note that the `autoyast2` keyword must point to the same file. If it is on a flash disk (for example, a USB stick), 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.

# V Uses for AutoYaST on Installed Systems

- 7 Running AutoYaST in an Installed System [228](#)

## 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 4.8.7, "Installing Packages in Stage 2"), a user (Section 4.28.1, "Users") section and an NTP-client (Section 4.19, "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/>
      </password_settings>
    </user>
  </users>
</profile>
```

```
<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:

```
tux > sudo yast2 ayast_setup setup filename=/tmp/install_foo.xml dopackages="yes"
```

For more information, run yast2 ayast\_setup longhelp

# VI Appendices

- A Handling Rules [231](#)
- B AutoYaST FAQ—Frequently Asked Questions [232](#)
- C Advanced **linuxrc** Options [236](#)
- D Differences Between AutoYaST Profiles in SLE 42.3 and 15 [241](#)

## A Handling Rules

The following figure illustrates how rules are handled and the processes of retrieval and merge.

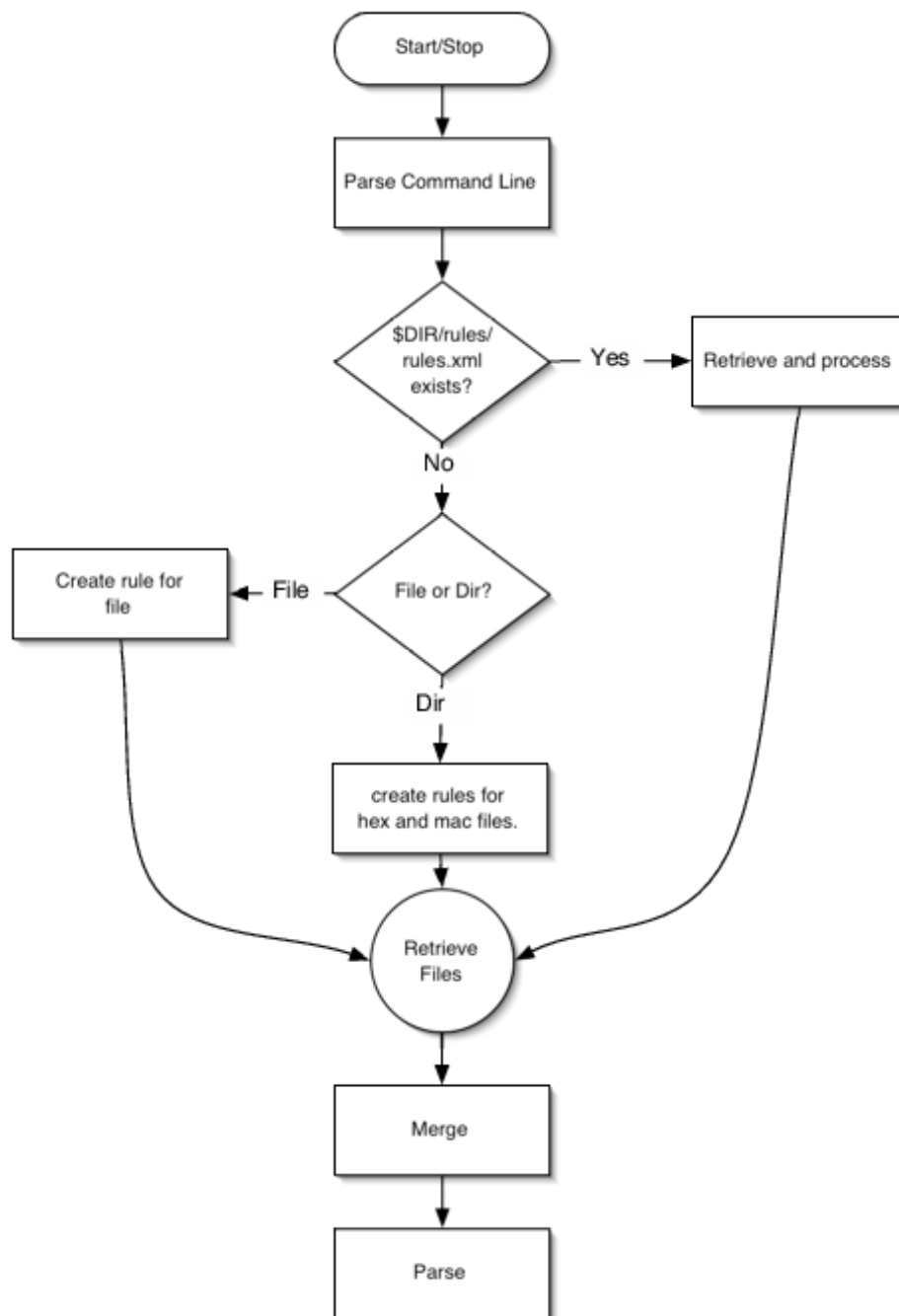


FIGURE A.1: RULES RETRIEVAL PROCESS

## B AutoYaST FAQ—Frequently Asked Questions

### Q: 1. How do I invoke an AutoYaST installation?

On all openSUSE Leap 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.

### Q: 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.

### Q: 3. How do I create an AutoYaST profile?

The easiest way to create an AutoYaST profile is to use an existing openSUSE Leap 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.

### Q: 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.

### Q: 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/configs">
  <users config:type="list">
    <user>
      <encrypted config:type="boolean">>false</encrypted>
      <user_password>linux</user_password>
      <username>root</username>
    </user>
  </users>
</profile>
```

**Q: 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>
```

**Q: 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`.

**Q: 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:

```
tux > /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.

**Q: 9.** *Can 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>
```

**Q: 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, to *merge* different profiles, sections need to be in alphabetical order.

**Q: 11.** linuxrc blocks the installation with File not signed. I need to manually interact.

linuxrc found an unsigned file, such as 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).

**Q: 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 the remote host. See [Section C.3, "Advanced Network Setup"](#) for more information.


**Q: 13.** *Is installation onto an NFS root (/) possible?*

Yes, but it is more complex than other methods. The environment (DHCP, TFTP, etc.) must be set up very carefully. The AutoYaST profile must 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/configs">
  <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>
```

**Q: 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 small program that runs after the kernel has loaded, but before AutoYaST or other stages. It prepares the system for installation. It allows the user to load modules, start an installed system or a rescue system, and to guide the operation of YaST.



### Note: AutoYaST and **linuxrc** Settings Are Not Identical

Some **linuxrc** settings coincidentally have the same names as settings used by AutoYaST in its `autoyast.xml` file. This does *not* mean that they take the same parameters or function in the same way. For example, AutoYaST takes a `self_update` setting. If this value is set to `1`, another setting, `self_update_url` will be read and followed. Although **linuxrc** also has a `self_update` setting, **linuxrc**'s setting takes values of either `0` or a URL.

Do not pass AutoYaST parameters to **linuxrc**, as this will almost certainly not give the desired results.

If **linuxrc** is installed on a machine, information about it can be found in the directory `/usr/share/doc/packages/linuxrc/`. Alternatively, its documentation can be found online at: <https://en.opensuse.org/SDB:Linuxrc>.



### 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.

To keep the **linuxrc** binary file as small as possible, all its libraries and other supplemental files are linked directly into the main program binary file. This means that there is no need for any shared libraries in the initial RAM disk, `initrd`.

## 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 (for example, a USB stick) 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 called `/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. The matching of `key` is on a case-insensitive basis.

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 will be 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: ADVANCED `linuxrc` KEYWORDS

Keyword: Example Value	Description
<code>addswap: 0 3 /dev/sda5</code>	If 0, never ask for swap; if the argument is a positive number <code>n</code> , activate the swap partition; if the argument is a partition name, activate this swap partition.
<code>autoyast: ftp://AUTOYASTFILE</code>	Location of the auto installation file; activates auto installation mode. See <a href="#">AutoYaST Control File Locations</a> for details.
<code>bootptimeout: 10</code>	10 seconds timeout for BOOTP requests.
<code>bootpwait: 5</code>	Sleep 5 seconds between network activation and starting bootp.
<code>display: color mono alt</code>	Set the menu color scheme.

Keyword: Example Value	Description
<u>exec: COMMAND</u>	Run <i>command</i> .
<u>forceinsmod: 0 1</u>	Use the <u>-f</u> option (force) when running <u>insmod</u> commands.
<u>forcerootime: 0 1</u>	Load the installation system into RAM disk.
<u>ifcfg:</u> <u>NETWORK_CONFIGURATION</u>	Set up and start the network. See <i>Section C.3, "Advanced Network Setup"</i> for more information.
<u>insmod: MODULE</u>	Load <u>MODULE</u> .
<u>install: URL</u>	Install from the repository specified with <u>URL</u> . For the syntax of <u>URL</u> refer to <a href="https://en.opensuse.org/SDB:Linuxrc#url_descr">https://en.opensuse.org/SDB:Linuxrc#url_descr</a> .
<u>keytable: de-lat1-nd</u>	Virtual console keyboard map to load.
<u>language: de_DE</u>	Language preselected for the installation.
<u>loghost: 10.10.0.22</u>	Enable remote logging via syslog.
<u>memloadimage: 50000</u>	Load installation system into RAM disk if free memory is above 50000 KB.
<u>memlimit: 10000</u>	Ask for swap if free memory drops below 10000 KB.
<u>memYaST: 20000</u>	Run YaST in text mode if free memory is below 20000 KB.
<u>memYaSTText: 10000</u>	Ask for swap before starting YaST if free memory is below 10000 KB.
<u>proxy: 10.10.0.1</u>	Proxy (either FTP or HTTP).
<u>rescue: 1 nfs://</u> <u>server/dir</u>	Load the rescue system; the URL variant specifies the location of the rescue image explicitly.
<u>rescueimage: /suse/</u> <u>images/rescue</u>	Location of the rescue system image.

Keyword: Example Value	Description
<u>rootimage</u> : /suse/images/root	Location of the installation system image.
<u>textmode</u> : 1	Start YaST in text mode.
<u>usbwait</u> : 4	Wait four seconds after loading the USB modules.
<u>y2confirm</u>	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). Because autoyast is not a linuxrc parameter (this parameter is ignored by linuxrc and is 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 openSUSE Leap 13.2, 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,NAME_SERVER_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](https://en.opensuse.org/SDB:Linuxrc#Network_Configuration).



## D Differences Between AutoYaST Profiles in SLE 42.3 and 15

Significant changes in openSUSE Leap 15, like the new modules concept or replacing SuSEfirewall2 with `firewalld`, required changes in AutoYaST. If you want to reuse existing openSUSE Leap 42.3 profiles with openSUSE Leap 15, you need to adjust them as documented here.

### D.1 Partitioning

The partitioning back-end previously used by YaST, `libstorage`, has been replaced by `libstorage-ng` which is designed to allow new capabilities that were not possible before. Despite the back-end change, the XML syntax for profiles has *not* changed. However, openSUSE Leap 15 comes with some general changes, which are explained below.

#### D.1.1 GPT Becomes the Default Partition Type on AMD64/Intel 64

On AMD64/Intel 64 systems, GPT is now the preferred partition type. However, if you would like to retain the old behavior, you can explicitly indicate this in the profile by setting the `disklabel` element to `msdos`.

#### D.1.2 Setting Partition Numbers

AutoYaST will no longer support forcing partition numbers, as it might not work in some situations. Moreover, GPT is now the preferred partition table type, so partition numbers are less relevant.

However, the `partition_nr` tag is still available to specify a partition to be reused. Refer to [Section 4.4.3.2, "Partition Configuration"](#) for more information.

### D.1.3 Forcing Primary Partitions

It is still possible to force a partition as `primary` (only on MS-DOS partition tables) by setting the `primary_type` to `primary`. However, any other value, like `logical`, will be ignored by AutoYaST, which will automatically determine the partition type.

### D.1.4 Btrfs: Default Subvolume Name

The new storage layer allows the user to set different default subvolumes (or none) for every Btrfs file system. As shown in the example below, a prefix name can be specified for each partition using the `subvolumes_prefix` tag:

#### EXAMPLE D.1: SPECIFYING THE BTRFS DEFAULT SUBVOLUME NAME

```
<partition>
  <mount>/</mount>
  <filesystem config:type="symbol">btrfs</filesystem>
  <size>max</size>
  <subvolumes_prefix>@</subvolumes_prefix>
</partition>
```

To omit the subvolume prefix, set the `subvolumes_prefix` tag:

#### EXAMPLE D.2: DISABLING BTRFS SUBVOLUMES

```
<partition>
  <mount>/</mount>
  <filesystem config:type="symbol">btrfs</filesystem>
  <size>max</size>
  <subvolumes_prefix>@</subvolumes_prefix>
</partition>
```

As a consequence of the new behavior, the old `btrfs_set_default_subvolume_name` tag is not needed and, therefore, it is not supported anymore.

### D.1.5 Btrfs: Disabling Subvolumes

Btrfs subvolumes can be disabled by setting `create_subvolumes` to `false`. To skip the default `@` subvolume, specify `subvolumes_prefix`.

```
<partition>
```

```
<create_subvolumes config:type="boolean">false</create_subvolumes>
<subvolumes_prefix><![CDATA[]]></subvolumes_prefix>
</partition>]]>
```

## D.1.6 Reading an Existing `/etc/fstab` Is No Longer Supported

On openSUSE Leap 15 the ability to read an existing `/etc/fstab` from a previous installation when trying to determine the partitioning layout, is no longer supported.

## D.1.7 Setting for Aligning Partitions Has Been Dropped

As cylinders have become obsolete, the `partition_alignment >` tag makes no sense and it is no longer available. AutoYaST will always try to align partitions in an optimal way.

## D.1.8 Using the `type` to Define an Volume Group

The `is_lvm_vg` element has been dropped in favor of setting the `type` to the `CT_LVM` value. Refer to the [Section 4.4.5, “Logical Volume Manager \(LVM\)”](#) for further details.

# D.2 Firewall Configuration

In openSUSE Leap 15, SuSEfirewall2 has been replaced by `firewalld` as the default firewall. The configuration of these two firewalls differs significantly, and therefore the respective AutoYaST profile syntax has changed.

Old profiles will continue working, but the supported configuration will be very limited. It is recommended to update profiles for Leap 15 as outlined below. If keeping Leap 42.3 profiles, we recommend to check the final configuration to avoid unexpected behavior or network security threats.

TABLE D.1: AUTOYAST FIREWALL CONFIGURATION IN LEAP 15: BACKWARD COMPATIBILITY

Supported (but deprecated)	Unsupported
<code>FW_CONFIGURATIONS_{DMZ, EXT, INT}</code>	<code>FW_ALLOW_FW_BROADCAST_{DMZ, EXT, INT}</code>

Supported (but deprecated)	Unsupported
<u>FW_DEV_{DMZ, EXT, INT}</u>	<u>FW_IGNORE_FW_BROADCAST_{DMZ, EXT, INT}</u>
<u>FW_LOG_DROP_ALL</u>	<u>FW_IPSECT_TRUST</u>
<u>FW_LOG_DROP_CRIT</u>	<u>FW_LOAD_MODULES</u>
<u>FW_MASQUERADE</u>	<u>FW_LOG_ACCEPT_ALL</u>
<u>FW_SERVICES_{DMZ, INT, EXT}_{TCP, UDP, IP}</u>	<u>FW_LOG_ACCEPT_CRIT</u>
	<u>FW_PROTECT_FROM_INT</u>
	<u>FW_ROUTE</u>
	<u>FW_SERVICES_{DMZ, EXT, INT}_RPC</u>
	<u>FW_SERVICES_ACCEPT_RELATED_{DMZ, EXT, INT}</u>

Configuration options from SuSEfirewall2 that are no longer available either have no equivalent mapping in `firewalld` or will be supported in future releases of openSUSE Leap. Some `firewalld` features are not yet supported by YaST and AutoYaST—you can use them with post installation scripts in your AutoYaST profile. See [Section 4.29, “Custom User Scripts”](#) for more information.



## Note: Enabling and Starting the Firewall

Enabling and starting the `systemd` service for `firewalld` is done with the same syntax as in Leap 42.3. This is the only part of the firewall configuration syntax in AutoYaST that has not changed:

```
<firewall>
  <enable_firewall config:type="boolean">true</enable_firewall>
  <start_firewall config:type="boolean">true</start_firewall>
  ...
</firewall>
```

The following examples show how to convert deprecated (but still supported) profiles to the Leap 15 syntax:

## D.2.1 Assigning Interfaces to Zones

Both SuSEfirewall2 and `firewalld` are zone-based, but have a different set of predefined rules and a different level of trust for network connections.

TABLE D.2: MAPPING OF SUSEFIREWALL2 AND `firewalld` ZONES

<code>firewalld</code> (Leap 15)	SuSEfirewall2 (Leap 42.3)
dmz	DMZ
external	EXT with <code>FW_MASQUERADE</code> set to <code>yes</code>
public	EXT with <code>FW_MASQUERADE</code> set to <code>no</code>
internal	INT with <code>FW_PROTECT_FROM_INT</code> set to <code>yes</code>
trusted	INT with <code>FW_PROTECT_FROM_INT</code> set to <code>no</code>
block	N/A
drop	N/A
home	N/A
work	N/A

In SuSEfirewall2 the default zone is the external one (EXT) but it also allows the use of the special keyword `any` to assign all the interfaces that are not listed anywhere to a specified zone.

### D.2.1.1 Default Configuration

The following two examples show the default configuration that is applied for the interfaces `eth0`, `eth1`, `wlan0` and `wlan1`.

EXAMPLE D.3: ASSIGNING ZONES: DEFAULT CONFIGURATION (DEPRECATED SYNTAX)

```
<firewall>
```

```

<FW_DEV_DMZ>any eth0</FW_DEV_DMZ>
<FW_DEV_EXT>eth1 wlan0</FW_DEV_EXT>
<FW_DEV_INT>wlan1</FW_DEV_INT>
</firewall>

```

#### EXAMPLE D.4: ASSIGNING ZONES: DEFAULT CONFIGURATION (LEAP 15 SYNTAX)

```

<firewall>
<default_zone>dmz</default_zone>
<zones config:type="list">
  <zone>
    <name>dmz</name>
    <interfaces config:type="list">
      <interface>eth0</interface>
    </interfaces>
  </zone>
  <zone>
    <name>public</name>
    <interfaces config:type="list">
      <interface>eth1</interface>
    </interfaces>
  </zone>
  <zone>
    <name>trusted</name>
    <interfaces config:type="list">
      <interface>wlan1</interface>
    </interfaces>
  </zone>
</zones>
</firewall>

```

### D.2.1.2 Masquerading and Protecting Internal Zones

The following two examples show how to configure the interfaces `eth0`, `eth1`, `wlan0` and `wlan1` with masquerading and protected internal zones.

#### EXAMPLE D.5: MASQUERADING AND PROTECTING INTERNAL ZONES (DEPRECATED SYNTAX)

```

<firewall>
<FW_DEV_DMZ>any eth0</FW_DEV_DMZ>
<FW_DEV_EXT>eth1 wlan0</FW_DEV_EXT>
<FW_DEV_INT>wlan1</FW_DEV_INT>
<FW_MASQUERADE>yes</FW_MASQUERADE>
<FW_PROTECT_FROM_INT>yes</FW_PROTECT_FROM_INT>
</firewall>

```

#### EXAMPLE D.6: MASQUERADING AND PROTECTING INTERNAL ZONES (LEAP 15 SYNTAX)

```
<firewall>
<default_zone>dmz</default_zone>
<zones config:type="list">
  <zone>
    <name>dmz</name>
    <interfaces config:type="list">
      <interface>eth0</interface>
    </interfaces>
  </zone>
  <zone>
    <name>external</name>
    <interfaces config:type="list">
      <interface>eth1</interface>
    </interfaces>
  </zone>
  <zone>
    <name>internal</name>
    <interfaces config:type="list">
      <interface>wlan1</interface>
    </interfaces>
  </zone>
</zones>
</firewall>
```

### D.2.2 Opening Ports

In SuSEfirewall2 the `FW_SERVICES_{DMZ,EXT,INT}_{TCP,UDP,IP,RPC}` tags were used to open ports in different zones.

For `TCP` or `UDP`, SuSEfirewall2 supported a port number or range, or a service name from `/etc/services` with a single tag for the respective zone and service. For IP services a port number or range, or a protocol name from `/etc/protocols` could be specified with `FW_SERVICES_ZONE_IP`.

For `firewalld` each port, port range, and service requires a separate entry in the `port` section for the respective zone. IP services need separate entries in the `protocol` section.

RPC services, which were supported by SuSEfirewall2, are no longer supported with `firewalld`.

#### EXAMPLE D.7: OPENING PORTS (DEPRECATED SYNTAX)

```
<firewall>
<FW_SERVICES_DMZ_TCP>ftp ssh 80 5900:5999</FW_SERVICES_DMZ_TCP>
```

```
<FW_SERVICES_EXT_UDP>1723 ipsec-nat-t</FW_SERVICES_EXT_UDP>
<FW_SERVICES_EXT_IP>esp icmp gre</FW_SERVICES_EXT_IP>
<FW_MASQUERADE>yes</FW_MASQUERADE>
</firewall>
```

#### EXAMPLE D.8: OPENING PORTS (LEAP 15 SYNTAX)

```
<firewall>
<zones config:type="list">
  <zone>
    <name>dmz</name>
    <ports config:type="list">
      <port>ftp/tcp</port>
      <port>ssh/tcp</port>
      <port>80/tcp</port>
      <port>5900-5999/tcp</port>
    </ports>
  </zone>
  <zone>
    <name>external</name>
    <ports config:type="list">
      <port>1723/udp</port>
      <port>ipsec-nat-t/udp</port>
    </ports>
    <protocols config:type="list">
      <protocol>esp</protocol>
      <protocol>icmp</protocol>
      <protocol>gre</protocol>
    </protocols>
  </zone>
</zones>
</firewall>
```

### D.2.3 Opening firewalld Services

For opening a combination of ports and/or protocols, SuSEfirewall2 provides the `FW_CONFIGURATIONS_{EXT, DMZ, INT}` tags which are equivalent to services in `firewalld`.

#### EXAMPLE D.9: OPENING SERVICES (DEPRECATED SYNTAX)

```
<firewall>
<FW_CONFIGURATIONS_EXT>dhcp dhcpv6 samba vnc-server</FW_CONFIGURATIONS_EXT>
<FW_CONFIGURATIONS_DMZ>ssh</FW_CONFIGURATIONS_DMZ>
</firewall>
```



#### EXAMPLE D.10: OPENING SERVICES (LEAP 15 SYNTAX)

```
<firewall>
  <zones config:type="list">
    <zone>
      <name>dmz</name>
      <services config:type="list">
        <service>ssh</service>
      </services>
    </zone>
    <zone>
      <name>public</name>
      <services config:type="list">
        <service>dhcp</service>
        <service>dhcpv6</service>
        <service>samba</service>
        <service>vnc-server</service>
      </services>
    </zone>
  </zones>
</firewall>
```

The services definition can be added via packages in both cases:

- SuSEfirewall2 Service Definitions: [https://en.opensuse.org/SuSEfirewall2/Service\\_Definitions\\_Added\\_via\\_Packages](https://en.opensuse.org/SuSEfirewall2/Service_Definitions_Added_via_Packages) ↗
- `firewalld` RPM Packaging [https://en.opensuse.org/firewalld/RPM\\_Packaging](https://en.opensuse.org/firewalld/RPM_Packaging) ↗  
`firewalld` already provides support for the majority of important services in `/usr/lib/firewalld/services`. Check this directory for an existing configuration before defining a new one.

### D.2.4 For More Information

- Official `firewalld` Documentation (<http://www.firewalld.org/documentation/>) ↗

## D.3 NTP Configuration

The time server synchronization daemon `ntpd` has been replaced with the more modern daemon `chrony`. Therefore the configuration syntax for the time-keeping daemon in AutoYaST has changed. AutoYaST profiles from Leap 42.3 that contain a section with `ntp:client` need to be updated.

Instead of containing low level configuration options, NTP is now configured by a set of high level options that are applied on top of the default settings:

EXAMPLE D.11: NTP CONFIGURATION (LEAP 15 SYNTAX)

```
<ntp-client>
  <ntp_policy>auto</ntp_policy>
  <ntp_servers config:type="list">
    <ntp_server>
      <iburst config:type="boolean">false</iburst>
      <address>cz.pool.ntp.org</address>
      <offline config:type="boolean">true</offline>
    </ntp_server>
  </ntp_servers>
  <ntp_sync>systemd</ntp_sync>
</ntp-client>
```

## D.4 AutoYaST Packages Are Needed for the Second Stage

A regular installation is performed in a single stage, while an installation performed via AutoYaST usually needs two stages. In order to perform the second stage of the installation AutoYaST requires a few additional packages, for example [autoyast2-installation](#) and [autoyast2](#). If these are missing, a warning will be shown.

## D.5 The CA Management Module Has Been Dropped

The module for CA Management (`yast2-ca-management >`) has been removed from openSUSE Leap 15, and for the time being there is no replacement available. In case you are reusing a Leap 42.3 profile, make sure it does not contain a `ca_mgm` section.

## D.6 Upgrade

### D.6.1 Software

Leap 42.3 has two modes of evaluating which packages need to be upgraded. In openSUSE Leap 15.2, upgrades are always determined by the dependency solver, equivalent to using **zypper dup**.

This makes the option only\_installed\_packages in the software section obsolete.