Experimental version for testing purpose only!

My private, inofficial Version of:

SUSE Linux Enterprise Server 15 SP7

AutoYaST Guide

AutoYaST Guide

SUSE Linux Enterprise Server 15 SP7

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

File generated at 2025-11-17 15:09

Contents

Preface

Available documentation

Online documentation

Our documentation is available online at https://documentation.suse.com. Browse or download the documentation in various formats.

Latest updates

The latest updates are usually available in the English-language version of this documentation.

SUSE Knowledgebase

If you run into an issue, check out the Technical Information Documents (TIDs) that are available online at https://www.suse.com/support/kb/. Search the SUSE Knowledgebase for known solutions driven by customer need.

Release notes

For release notes, see https://www.suse.com/releasenotes/.

In your system

For offline use, the release notes are also available under `/usr/share/doc/release-notes` on your system. The documentation for individual packages is available at `/usr/share/doc/packages`.

Many commands are also described in their *manual pages*. To view them, run **`man`**, followed by a specific command name. If the **`man`** command is not installed on your system, install it with **`sudo zypper install man`**.

Improving the documentation

Your feedback and contributions to this documentation are welcome. The following channels for giving feedback are available:

Service requests and support

For services and support options available for your product, see https://www.suse.com/support/.

To open a service request, you need a SUSE subscription registered at SUSE Customer Center. Go to https://scc.suse.com/support/requests, log in, and click *Create New*.

Bug reports

Report issues with the documentation at https://bugzilla.suse.com/.

To simplify this process, click the *Report an issue* icon next to a headline in the HTML version of this document. This preselects the right product and category in Bugzilla and adds a link to the current section. You can start typing your bug report right away.

A Bugzilla account is required.

Contributions

To contribute to this documentation, click the *Edit source document* icon next to a headline in the HTML version of this document. This will take you to the source code on GitHub, where you can open a pull request.

A GitHub account is required.

***Edit source document* only available for English**

The *Edit source document* icons are only available for the English version of each document. For all other languages, use the *Report an issue* icons instead.

For more information about the documentation environment used for this documentation, see the repository's README.

Mail

You can also report errors and send feedback concerning the documentation to `doc-team@suse.com`. Include the document title, the product version, and the publication date of the document. Additionally, include the relevant section number and title (or provide the URL) and provide a concise description of the problem.

Documentation conventions

The following notices and typographic conventions are used in this document:

- `/etc/passwd`: Directory names and file names
- *PLACEHOLDER*: Replace *PLACEHOLDER* with the actual value
- `PATH`: An environment variable

- **`ls`**, `--help`: Commands, options, and parameters
- `user`: The name of a user or group
- `package_name`: The name of a software package
- `Alt`, `Alt`–`F1`: A key to press or a key combination. Keys are shown in uppercase as on a keyboard.
- *File*, *File* > *Save As*: menu items, buttons
- x86_64 ▶ This paragraph is only relevant for the AMD64/Intel 64 architectures. The arrows mark the beginning and the end of the text block. ◀

  zseries;power ▶ This paragraph is only relevant for the architectures `IBM Z` and `POWER`. The arrows mark the beginning and the end of the text block. ◀
- *Chapter 1, "Example chapter"*: A cross-reference to another chapter in this guide.
- Commands that must be run with `root` privileges. You can also prefix these commands with the **`sudo`** command to run them as a non-privileged user:

```
#command>sudocommand
```

- Commands that can be run by non-privileged users:

```
>command
```

- Commands can be split into two or multiple lines by a backslash character (\) at the end of a line. The backslash informs the shell that the command invocation will continue after the end of the line:

```
>echo a b \
c d
```

- A code block that shows both the command (preceded by a prompt) and the respective output returned by the shell:

```
>command
output
```

- Notices

Warning notice

Vital information you must be aware of before proceeding. Warns you about security issues, potential loss of data, damage to hardware, or physical hazards.

Important notice

Important information you should be aware of before proceeding.

Note notice

Additional information, for example about differences in software versions.

Tip notice

Helpful information, like a guideline or a piece of practical advice.

- Compact Notices

Note

Additional information, for example about differences in software versions.

Tip

Helpful information, like a guideline or a piece of practical advice.

Support

Find the support statement for SUSE Linux Enterprise Server and general information about technology previews below. For details about the product lifecycle, see https://www.suse.com/lifecycle.

If you are entitled to support, find details on how to collect information for a support ticket at https://documentation.suse.com/sles-15/html/SLES-all/cha-adm-support.html.

Support statement for SUSE Linux Enterprise Server

To receive support, you need an appropriate subscription with SUSE. To view the specific support offers available to you, go to https://www.suse.com/support/ and select your product.

The support levels are defined as follows:

L1

Problem determination, which means technical support designed to provide compatibility information, usage support, ongoing maintenance, information gathering and basic troubleshooting using available documentation.

L2

Problem isolation, which means technical support designed to analyze data, reproduce customer problems, isolate a problem area and provide a resolution for problems not resolved by Level 1 or prepare for Level 3.

L3

Problem resolution, which means technical support designed to resolve problems by engaging engineering to resolve product defects which have been identified by Level 2 Support.

For contracted customers and partners, SUSE Linux Enterprise Server is delivered with L3 support for all packages, except for the following:

- Technology previews.
- Sound, graphics, fonts, and artwork.
- Packages that require an additional customer contract.
- Some packages shipped as part of the module *Workstation Extension* are L2-supported only.
- Packages with names ending in `-devel` (containing header files and similar developer resources) will only be supported together with their main packages.

SUSE will only support the usage of original packages. That is, packages that are unchanged and not recompiled.

Technology previews

Technology previews are packages, stacks, or features delivered by SUSE to provide glimpses into upcoming innovations. Technology previews are included for your convenience to give you a chance to test new technologies within your environment. We would appreciate your feedback. If you test a technology preview, please contact your SUSE representative and let them know about your experience and use cases. Your input is helpful for future development.

Technology previews have the following limitations:

- Technology previews are still in development. Therefore, they may be functionally incomplete, unstable, or otherwise *not* suitable for production use.
- Technology previews are *not* supported.
- Technology previews may only be available for specific hardware architectures.
- Details and functionality of technology previews are subject to change. As a result, upgrading to subsequent releases of a technology preview may be impossible and require a fresh installation.
- SUSE may discover that a preview does not meet customer or market needs, or does not comply with enterprise standards. Technology previews can be removed from a product at any time. SUSE does not commit to providing a supported version of such technologies in the future.

For an overview of technology previews shipped with your product, see the release notes at https://www.suse.com/releasenotes.

Chapter 1. Introduction to AutoYaST

1.1. Motivation

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

AutoYaST can be used where no user intervention is required or where customization is required. Using an AutoYaST profile, 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 profile”. You can create this using existing configuration resources, and easily tailor it 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 ability to configure a computer by using existing modules, and avoid 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 in the profile. The profile is transferred onto the target system where its directives will be parsed and fed into YaST.
- Installation: YaST performs the installation and basic configuration (for example, partitioning, networking, firewall) of the target system using the data from the AutoYaST profile.
- Post-configuration: After the installation and configuration of the basic system, the system can run a second stage to perform any additional configurations that require the target system to be already running, such as post-installation scripts, third party modules, or some YaST modules.

Second stage

A regular installation of SUSE Linux Enterprise Server15 SP7 is performed in a single stage. The auto-installation process, however, is divided into two stages. After the installation and main configuration of the basic system, it is booted into a second stage to perform any post-installation configuration steps.

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 runs only if it is strictly necessary, and the second stage can be turned off completely with the `second_stage` parameter:

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

Part I. Understanding and creating the AutoYaST control file

Chapter 2. The AutoYaST control file

2.1. Introduction

A *control file*, also known as a *profile*, 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.

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 new control file, 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 *the section called “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 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, although you can specify it as you want. Lists must be marked as such using the `config:type="list"` attribute.

Using shorter type annotations

Starting with SUSE Linux Enterprise Server 15 SP3, it is possible to use the attribute `t` instead of `config:type` to specify the element type.

```
<mode t="boolean">true</mode>
```

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. There is one exception. When the map is empty, it needs to be marked as a map so it does not get confused with a simple string value.

Example 2.4. An empty map

```
<general t="map" />
```

For properties, `boolean`, `symbol`, and `integer` can be used, the default being a string.

Except for map and string values, as explained before, 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.

Chapter 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)

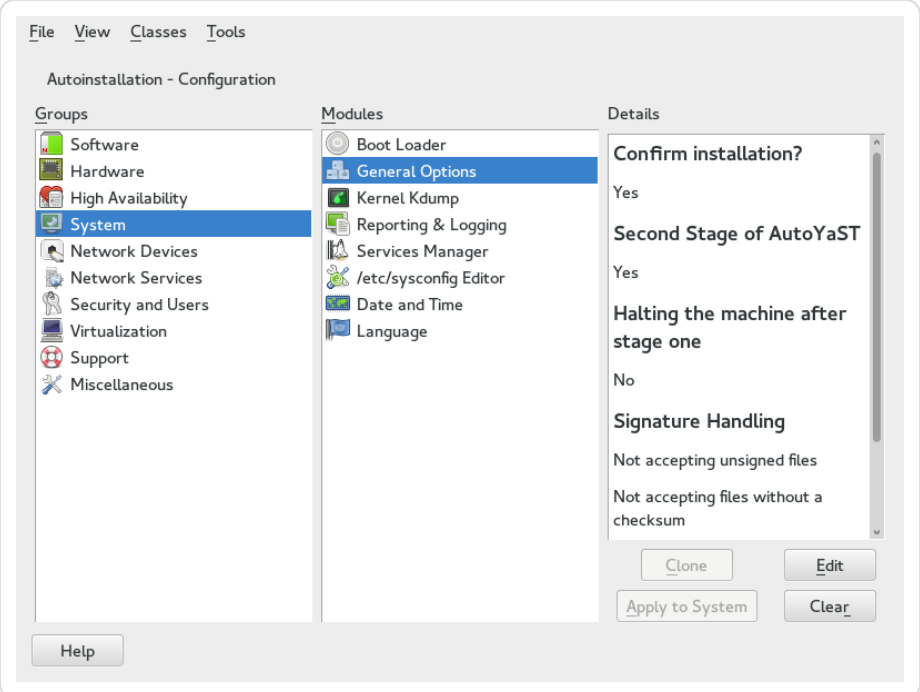
Also verify that both `autoyast2-installation` and `autoyast2` are installed.

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 SUSE Linux Enterprise Server 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 SUSE Linux Enterprise Server system as a template. On an already installed system, launch *YaST* > *Miscellaneous* > *Autoinstallation Configuration*. Then 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 launching *YaST* > *Miscellaneous* > *Autoinstallation Cloning System* or running **`sudo yast clone_system`** from the command line.

Both methods will create the file `/root/autoinst.xml`. The cloned profile can be used to set up an identical clone of the system it was created from. However, you will usually want to adjust the file to allow for installing multiple machines that are very similar, but not identical. This can be done by adjusting the profile with your favorite text/XML editor.

Sensitive data in profiles

Be aware that the profile might contain sensitive information such as password hashes and registration keys.

Carefully review the exported profiles and make sure to keep file permissions restrictive.

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`). Call AutoYaST using the YaST control center or as root with the following command (make sure the `DISPLAY` variable is set correctly to start the graphical user interface instead of the text-based one):

```
/sbin/yast2 autoyast
```

3.3. Creating/editing a control file manually

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

```
xmllint <control file>
```

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

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

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

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

Schema extensions

AutoYaST can be extended by other products and modules, but the schema does not contain the specification for those extensions. As a consequence, when AutoYaST is given a profile that uses one of those extensions, it might report the profile as invalid.

Thus, starting in SUSE Linux Enterprise Server SP3, AutoYaST does not validate top-level unknown sections, and ignores them. For example, in the example below, `<sap-inst>` is not validated. The rest is validated as usual.

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

<sap-inst>
  <!-- this section is not validated -->>
</sap-inst>
```

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 multiple machines, and it should only be seen as an interface between the autoinstallation engine and the Configuration Management System (CMS).

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

3.5. Checking a control file

Depending on the use case, creating an AutoYaST profile can be difficult, especially if you build a dynamic profile using rules/classes, ERB templates or pre-scripts. For more information, see *Part III, "Managing mass installations with dynamic profiles"*.

Starting with SUSE Linux Enterprise Server 15 SP3, AutoYaST validates the profile during the installation, reporting any problem found to the user. Although it is recommended to check whether the profile is correct or not, you can disable this behavior by setting the `YAST_SKIP_XML_VALIDATION` boot parameter to `1`.

Moreover, to simplify the testing and debugging process, AutoYaST offers the `check-profile` command, which takes care of fetching, building and, optionally, importing the profile to detect any potential problem.

Results may vary

Although this command uses the same approach as the installation, the results may vary depending on the differences between the current system and installation media: YaST package versions, architecture, etc.

Use only trusted profiles

You must be careful when running this command because pre-installation scripts and ERB code would run as the `root` user. Use only profiles that you trust.

3.5.1. Basic checks

The simplest way to use this command is just to read and validate the profile:

```
>sudo  yast2 autoyast check-profile filename=autoinst.xml output=result.xml
```

The `result.xml` file contains the result of evaluating the profile. Bear in mind that, even if you do not use any advanced feature, the content of `autoinst.xml` and `result.xml` may differ. The reason is that AutoYaST does some cleaning up when it processes the profile.

`check-profile` can deal with remote files too:

```
>sudo  yast2 autoyast check-profile filename=http://192.168.1.100/autoinst.xml
output=result.xml
```

3.5.2. Running pre-scripts

Optionally, AutoYaST can run the scripts that are included in the profile, reporting any error found during the execution. This is especially relevant if you are using a pre-installation script to modify the profile. To enable this feature, you need to set the `run-scripts` option to `true`.

```
>sudo  yast2 autoyast check-profile filename=http://192.168.1.100/autoinst.xml
output=result.xml run-scripts=true
```

Scripts run as root

You must be careful when enabling the `run-scripts` option, because the scripts will run as root and they may affect the current system.

3.5.3. Importing the profile

It is possible to face some problems when importing a valid profile, even if it is correct. The reason is that AutoYaST does not perform any logic check when fetching, building and validating the profile.

To anticipate such problems, the `check-profile` command imports the profile and reports problems that it has detected. As it may take a while, you can disable this behavior by setting the `import-all` option to `false`.

```
>sudo  yast2 autoyast check-profile filename=http://192.168.1.100/autoinst.xml
output=result.xml import-all=false
```

Importing the profile is a safe operation and does not alter the underlying system in any way.

Part II. AutoYaST configuration examples

Chapter 4. Configuration and installation options

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

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

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

4.1. General options

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

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns"
 xmlns:config="http://www.suse.com/1.0/configns">
 <general>
  <ask-list>❶
   ...
  </ask-list>
  <cio_ignore>❷
   ...
  </cio_ignore>
  <mode>❸
   ...
  </mode>
  <proposals>❹
   ...
  </proposals>
  <self_update>❺
  ...
 </self_update>
  <self_update_url>
   ...
  </self_update_url>
  <semi-automatic config:type="list">❻
  ...
  </semi-automatic>
  <signature-handling>❼
   ...
  </signature-handling>
  <storage>❽
   ...
  </storage>
  <wait>❾
   ...
  </wait>
 </general>
<profile>
```

❶ *the section called “Ask the user for values during installation”*

❷ *the section called “Ignoring unused devices on IBM Z”*

❸ *the section called “The mode section”*

❹ *the section called “Configuring the installation settings screen”*

❺ *Section 4.1.3, &The self-update section&*

❻ *Section 4.1.4, &The semi-automatic section&*

❼ *the section called “The signature handling section”*

❽ *the section called “Partitioning”*

❾ *the section called “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 *the section called "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>
```

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;
      </ntp_sync_time_before_installation>
     </mode>
     ...
    </general>
```

second_stage

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

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

4.1.2. Configuring the installation settings screen

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

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

4.1.3. The self-update section

During the installation, YaST can update itself to solve bugs in the installer that were discovered after the release. Refer to the Deployment Guide for further information about this feature.

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 section called “Custom self-update repositories” in “Deployment Guide”.

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 gives administrators installing the machine the ability to manually configure some aspects of the installation, while also 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.

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. Ignoring unused devices on IBM Z

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

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

4.1.8. 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/configns">
 <general>
  <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>
  <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.

Critical system messages

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

4.3. System registration and extension selection

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

```
<suse_register>
  <do_registration config:type="boolean">true</do_registration>
  <email>tux@example.com</email>
  <reg_code>MY_SECRET_REGCODE</reg_code>
  <install_updates config:type="boolean">true</install_updates>
  <slp_discovery config:type="boolean">false</slp_discovery>
  <--! optionally register some add-ons -->
  <addons config:type="list">
    <addon>
      <name>sle-module-basesystem</name>
      <version>15.7</version>
      <arch>x86_64</arch>
    </addon>
  </addons>
</suse_register>
```

It is recommended to at least register the Basesystem Module to have access to the updates for the base system (the Linux kernel, the system libraries and services).

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

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

Using the installation network settings

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

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

`suse_register` Values

do_registration

Boolean

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

Specify whether the system should be registered or not. If set to `false` all other options are ignored and the system is not registered.

e-mail

E-mail address

```
<email>tux@example.com</email>
```

Optional. The e-mail address matching the registration code.

reg_code

Text

```
<reg_code>SECRET_REGCODE</reg_code>
```

Required. Registration code.

install_updates

Boolean

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

Optional. Determines if updates from the Updates channels should be installed. The default value is to not install them (`false`).

slp_discovery

Boolean

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

Optional. Search for a registration server via SLP. The default value is `false`.

Expects to find a single server. If more than one server is found, the installation will fail. In case there is more than one registration server available, you need to specify one with `reg_server`.

If neither `slp_discovery` nor `reg_server` are set, the system is registered with the SUSE Customer Center.

This setting also affects the self-update feature: If it is disabled, no SLP search will be performed.

reg_server

URL

```
<reg_server>https://smt.example.com</reg_server>
```

Optional. RMT server URL. If neither `slp_discovery` nor `reg_server` are set, the system is registered with the SUSE Customer Center.

The RMT server is queried for a URL of the self-update repository. So if `self_update_url` is not set, the RMT server influences where the self-updates are downloaded from. Check the Deployment Guide to find further information about this feature.

reg_server_cert_fingerprint_type

`SHA1` or `SHA256`

```
<reg_server_cert_fingerprint_type>SHA1</reg_server_cert_fingerprint_type>
```

Optional. Requires a checksum value provided with `reg_server_cert_fingerprint`. Using the fingerprint is recommended, since it ensures the SSL certificate is verified. The matching certificate will be automatically imported when the SSL communication fails because of a verification error.

reg_server_cert_fingerprint

Server Certificate Fingerprint value in hexadecimal notion (case-insensitive).

```
<reg_server_cert_fingerprint>01:AB...:EF</reg_server_cert_fingerprint>
```

Optional. Requires a fingerprint type value provided with `reg_server_cert_fingerprint_type`. Using the fingerprint is recommended, since it ensures the SSL certificate is verified. The matching certificate will be automatically imported when SSL communication fails because of a verification error.

reg_server_cert

URL

```
<reg_server_cert>http://smt.example.com/smt.crt</reg_server_cert>
```

Optional. URL of the SSL certificate on the server. Using this option is not recommended, since the certificate that is downloaded is not verified. Use `reg_server_cert_fingerprint` instead.

addons

Add-ons list

Specify an extension from the registration server that should be added to the installation repositories. See *the section called “Extensions”* for details.

Obtaining a server certificate fingerprint

To obtain a server certificate fingerprint for use with the `reg_server_cert_fingerprint` entry, run the following command on the SMT server (edit the default path to the `smt.crt` file, if needed):

```
openssl x509 -noout -in /srv/www/htdocs/smt.crt -fingerprint -sha256
```

To retrieve a fingerprint from the SMT server, use the following command:

```
curl --insecure -v https://scc.suse.com/smt.crt 2> /dev/null |
openssl \
      x509 -noout -fingerprint -sha256
```

Replace `scc.suse.com` with your server.

Note: This can be used in a trusted network only! In a non-trusted network, for example the Internet, you should get the fingerprint directly from the server by other means. Fingerprints can be fetched via SSH, a saved server configuration and other sources. Alternatively, you can verify that the downloaded certificate is identical on the server.

4.3.1. Extensions

The SUSE Customer Center provides several extensions, such as `sle-module-development-tools` (Development Tools Module) that can be included as additional sources during the installation. Extensions can be added via the `addons` property within the `suse_register` block.

Availability of extensions

The availability of extensions is product and architecture dependent, not all extensions are available on all architectures.

Some extensions, such as `sle-ha`, require a registration code. Depending on your subscription, either use a dedicated registration code for the extension, or restate the code for the base product.

With **`SUSEConnect --list-extensions`** you can list all available extensions in a registered system, and the commands to activate and disable them.

The following example shows which extensions are already activated, and labels the extensions that require their own registration codes:

```
>sudo SUSEConnect --list-extensions
AVAILABLE EXTENSIONS AND MODULES

    Basesystem Module 15 SP 7 x86_64 (Activated)
    Deactivate with: SUSEConnect -d -p sle-module-basesystem/15.7/x86_64

        Containers Module 15 SP 7 x86_64
        Activate with: SUSEConnect -p sle-module-containers/15.7/x86_64

        Desktop Applications Module 15 SP 7 x86_64 (Activated)
        Deactivate with: SUSEConnect -d -p sle-module-desktop-applications/
        15.7/x86_64

            SUSE Linux Enterprise Workstation Extension 15 SP 7 x86_64 (BETA)
            Activate with: SUSEConnect -p sle-we/15.7/x86_64 -r ADDITIONAL
REGCODE
            [...]
```

The `-p` argument (in the above example) displays the *NAME/VERSION/ARCH* values that can be used in the AutoYaST profile.

The following example shows how to configure a list of extensions. These go in the `suse_register` block:

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

  <--! optionally register some add-ons -->
 <addons config:type="list">
   <addon>
    <!-- Development Tools Module -->
    <!-- Depends on: Desktop Applications Module -->
    <name>sle-module-development-tools</name>
    <version>15.3</version>
    <arch>x86_64</arch>
   </addon>

   <addon>
    <!-- SUSE CaaS Platform (BETA) -->
    <!-- Depends on: Containers Module -->
    <name>caasp</name>
    <version>4.0</version>
    <arch>x86_64</arch>
    <reg_code>REG_CODE_REQUIRED</reg_code>
   </addon>

   <addon>
    <!-- SUSE Enterprise Storage -->
    <!-- Depends on: Server Applications Module -->
    <name>ses</name>
    <version>6</version>
    <arch>x86_64</arch>
    <reg_code>REG_CODE_REQUIRED</reg_code>
   </addon>

   <addon>
    <!-- SUSE Linux Enterprise High Availability Extension -->
    <!-- Depends on: Server Applications Module -->
    <name>sle-ha</name>
    <version>15.3</version>
    <arch>x86_64</arch>
    <reg_code>REG_CODE_REQUIRED</reg_code>
   </addon>
 </addons>
</suse_register>
```

You may also see all available modules and extensions at https://scc.suse.com/packages. Select your product and architecture, then click the In Module form to see a list of all extensions.

Extension dependencies

Since SLES 15, AutoYaST automatically reorders the extensions according to their dependencies during registration. This means the order of the extensions in the AutoYaST profile is not important.

Also AutoYaST automatically registers the dependent extensions even though they are missing in the profile. This means you are not required to fill the extensions list completely.

However, if the dependent extension requires a registration key, this must be specified in the profile, including the registration key. Otherwise the registration would fail.

The architecture and version of an extension are not mandatory. The registration workflow will evaluate the right one.

4.4. The GRUB 2 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/`

By default, AutoYaST proposes the same booting mechanism as used by the booting medium. For example, if you boot using EFI, the GRUB 2 for EFI is installed. Therefore, you can omit this section unless you have specific requirements. As the EFI boot requires specific partitioning, we recommend using the automatic partitioning as described in *the section called "Automatic partitioning"*, which will create all needed partitions automatically.

If you need to adapt the default, use the `<bootloader>` part. Its general structure looks like the following snippet:

```
<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>
```

You do not need to fill out all settings. Rather, you only need to define those that you need to change. AutoYaST will then merge the default values with those specified in the profile.

4.4.1. 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.

4.4.2. Globals

This is an important if optional part. Define here where to install GRUB 2 and how the boot process will work. Again, **`yast2-bootloader`** will propose 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.

Hibernation

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.

Hibernation

If there is a need for specific hibernation settings, then `resume` or `noresume` in the `append` configuration can be used.

To disable hibernation regardless of what the installer proposes, specify `noresume` as a kernel parameter in the `append` section.

To specify the hibernation device, use the `resume` key with the device path. The recommended way to get stable results is configuring your own partitioning and having a swap device with a label:

```
        <append>quiet resume=/dev/disk/by-label/my_swap</append>
```

If you do not use `resume` or `noresume`, or if `resume` specifies a device that will not exist on the installed system, then the installer may propose a correct value for `resume`, or it may remove the hibernation parameter completely, depending on installer logic.

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

Boot loader global options

activate

Set the boot flag on the boot partition. The boot partition can be `/` if there is no separate `/boot` partition. If the boot partition is on a logical partition, the boot flag is set to the extended partition.

```
<activate>true</activate>
```

append

Kernel parameters added at the end of boot entries for normal and recovery mode.

```
<append>nomodeset vga=0x317</append>
```

boot_boot

Write GRUB 2 to a separate `/boot` partition. If no separate `/boot` partition exists, GRUB 2 will be written to `/`.

```
<boot_boot>false</boot_boot>
```

boot_custom

Write GRUB 2 to a custom device.

```
<boot_custom>/dev/sda3</boot_custom>
```

boot_extended

Write GRUB 2 to the extended partition (important if you want to use generic boot code and the `/boot` partition is logical). Note: if the boot partition is logical, you should use `boot_mbr` (write GRUB 2 to MBR) rather than `generic_mbr`.

```
<boot_extended>false</boot_extended>
```

boot_mbr

Write GRUB 2 to the MBR of the first disk in the order. (`device.map` includes the order of the disks.)

```
<boot_mbr>false</boot_mbr>
```

boot_root

Write GRUB 2 to `/` partition.

```
<boot_root>false</boot_root>
```

cpu_mitigations

Lets you select a default setting of kernel boot command-line parameters for CPU mitigation (and, at the same time, strike a balance between security and performance).

Possible values are:

auto

Enables all mitigations required for your CPU model, but does not protect against cross-CPU thread attacks. This setting may impact performance to some degree, depending on the workload.

nosmt

Provides the full set of available security mitigations. Enables all mitigations required for your CPU model. In addition, it disables Simultaneous Multithreading (SMT) to avoid side-channel attacks across multiple CPU threads. This setting may further impact performance, depending on the workload.

off

Disables all mitigations. Side-channel attacks against your CPU are possible, depending on the CPU model. This setting has no impact on performance.

manual

Does not set any mitigation level. Specify your CPU mitigations manually by using the kernel command line options.

```
<cpu_mitigations>auto</cpu_mitigations>
```

If not set in AutoYaST, the respective settings can be changed via kernel command line. By default, the (product-specific) settings in the `/control.xml` file on the installation medium are used (if nothing else is specified).

generic_mbr

Write generic boot code to the MBR (will be ignored if `boot_mbr` is set to `true`).

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

gfxmode

Graphical resolution of the GRUB 2 screen (requires <terminal> to be set to `gfxterm`).

Valid entries are `auto`, *HORIZONTAL*x*VERTICAL*, or *HORIZONTAL*x*VERTICAL*x*COLOR DEPTH*. You can see the screen resolutions supported by GRUB 2 on a particular system by using the **vbeinfo** command at the GRUB 2 command line in the running system.

```
<gfxmode>1280x1024x24</gfxmode>
```

os_prober

If set to `true`, automatically searches for operating systems already installed and generates boot entries for them during the installation.

```
<os_prober>false</os_prober>
```

password

If this is defined, it protects the boot loader with a password. The system will not boot until the password is entered.

It has three subelements: `value`, `encrypted`, and `unrestricted`.

`value` holds the password. It can be either plain text, which YaST will encrypt, or a password already encrypted with **grub-mkpasswd-pbkdf2**. Set `encrypted` to `true` when you use an already encrypted password.

When `unrestricted` is set to `false`, users need the password defined by the `value` subelement to boot or edit GRUB 2 menu entries (by pressing `E` on a selected boot menu item). When it is set to `true`, users can boot the system without a password, but need a password to edit GRUB 2 menu entries. If the option is omitted, it defaults to `true`.

For more information on managing boot passwords, see *Protect Boot Loader with Password* in "Administration Guide".

```
<password><value>my_strong_password</value><encrypted>false</
encrypted><unrestricted>false</unrestricted></password>
```

suse_btrfs

Obsolete and no longer used. Booting from Btrfs snapshots is automatically enabled.

serial

Command to execute if the GRUB 2 terminal mode is set to `serial`.

```
<serial>serial --speed=115200 --unit=0 --word=8 --parity=no --stop=1</
serials>
```

secure_boot

If set to `false`, then UEFI secure boot is disabled. Works only for `grub2-efi` boot loader.

```
<secure_boot>false</secure_boot>
```

terminal

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

```
<terminal>serial</terminal>
```

timeout

The timeout in seconds until the default boot entry is booted automatically.

```
<timeout config:type="integer">10</timeout>
```

trusted_boot

If set to `true`, then Trusted GRUB is used. Trusted GRUB supports Trusted Platform Module (TPM). Works only for `grub2` boot loader.

```
<trusted_boot">true</trusted_boot>
```

update_nvram

If set to `true`, then AutoYaST adds an NVRAM entry for the boot loader in the firmware. This is the desirable behavior unless you want to preserve a specific setting or you need to work around firmware issues.

```
<update_nvram>true</update_nvram>
```

vgamode

Adds the kernel parameter `vga=`*`VALUE`* to the boot entries.

```
<vgamode>0x317</vgamode>
```

xen_append

Kernel parameters added at the end of boot entries for Xen guests.

```
<xen_append>nomodeset vga=0x317</xen_append>
```

xen_kernel_append

Kernel parameters added at the end of boot entries for Xen kernels on the VM Host Server.

```
<xen_kernel_append>dom0_mem=768M</xen_kernel_append>
```

4.4.3. Device map

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

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

4.5. The Systemd boot loader

This documentation is for **`yast2-bootloader`** and applies to systemd-boot.

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

```
<bootloader>
  <loader_type>
    systemd-boot
  </loader_type>
  <global>
    <!--
      entries defining the installation settings for systemd-boot and
      the generic boot code
    -->
  </global>
 </bootloader>
```

4.5.1. Loader type

This defines which boot loader (systemd-boot) to use. Not all architectures support both legacy and EFI variants of the boot loader.

```
<loader_type>systemd-boot</loader_type>
```

4.5.2. Globals

This is an important if optional part. Define here where to install systemd-boot and how the boot process will work. **`yast2-bootloader`** proposes a configuration if you do not define one. Unless you have some special requirements, do not specify the boot loader configuration in the XML file.

```
<global>
  <timeout config:type="integer">10</timeout>
  <secure_boot>false</secure_boot>
</global>
```

Boot loader global options

secure_boot

If set to `false`, then UEFI secure boot is disabled.

```
<secure_boot>false</secure_boot>
```

timeout

The timeout in seconds until the default boot entry is booted automatically.

```
<timeout config:type="integer">10</timeout>
```

4.6. 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 wants to set some basic settings. For example, a user wants 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*.

4.6.1. 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 *the section called “Guided partitioning”*.

4.6.2. 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>
```

lvm_vg_reuse

Tells the installer whether an existing LVM should be reused in the proposal. The default is `true`.

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

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>
```

4.6.3. 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 can 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>
  <partitioning>
```

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.

4.6.3.1. 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

device

Optional, the device you want to configure. If left out, AutoYaST tries to guess the device. See *Skipping devices* on how to influence guessing.

If set to `ask`, AutoYaST will ask the user which device to use during installation.

You can use persistent device names via ID, like `/dev/disk/by-id/`*`ata-WDC_WD3200AAKS-75L9`* or *by-path*, like `/dev/disk/by-path/`*`pci-0001:00:03.0-scsi-0:0:0:0`*.

```
<device>/dev/sda</device>
```

In case of volume groups, software RAID or `bcache` devices, the name in the installed system may be different (to avoid clashes with existing devices).

See *the section called "Multipath support"* for further information about dealing with multipath devices.

initialize

Optional, the default is `false`. If set to `true`, the partition table is wiped out before AutoYaST starts the partition calculation.

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

partitions

Optional, a list of `<partition>` entries (see *the section called "Partition configuration"*).

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

If no partitions are specified, AutoYaST will create a reasonable partitioning layout (see *the section called "Filling the gaps"*).

pesize

Optional, for LVM only. The default is 4M for LVM volume groups.

```
<pesize>8M</pesize>
```

use

Recommended, specifies the strategy AutoYaST will use to partition the hard disk. Choose from:

`all`, uses the whole device while calculating the new partitioning.

`linux`, only existing Linux partitions are used.

`free`, only unused space on the device is used, no existing partitions are touched.

1,2,3, a list of comma-separated partition numbers to use.

type

Optional, specifies the type of the `drive`. The default is `CT_DISK` for a normal physical hard disk. The following is a list of all options:

`CT_DISK` for physical hard disks (default).

`CT_LVM` for LVM volume groups.

`CT_MD` for software RAID devices.

`CT_DMMULTIPATH` for Multipath devices (deprecated, implied with CT_DISK).

`CT_BCACHE` for software `bcache` devices.

`CT_BTRFS` for multi-device Btrfs file systems.

`CT_NFS` for NFS.

`CT_TMPFS` for `tmpfs` file systems.

```
<type config:type="symbol">CT_LVM</type>
```

disklabel

Optional. By default YaST decides what makes sense. If a partition table of a different type already exists, it will be re-created with the given type only if it does not include any partition that should be kept or reused. To use the disk without creating any partition, set this element to `none`. The following is a list of all options:

`msdos`

`gpt`

`none`

```
<disklabel>gpt</disklabel>
```

keep_unknown_lv

Optional, the default is `false`.

This value only makes sense for type=CT_LVM drives. If you are reusing a logical volume group and you set this to `true`, all existing logical volumes in that group will not be touched unless they are specified in the <partitioning> section. So you can keep existing logical volumes without specifying them.

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

enable_snapshots

Optional, the default is `true`.

Enables snapshots on Btrfs file systems mounted at `/` (does not apply to other file systems, or Btrfs file systems not mounted at `/`).

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

quotas

Optional, the default is `false`.

Enables support for Btrfs subvolume quotas. Setting this element to `true` will enable support for quotas for the file system. However, you need to set the limits for each subvolume. Check *the section called "Btrfs subvolumes"* for further information.

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

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.

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.

4.6.3.2. Partition configuration

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

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

create

Specify if this partition or logical volume must be created, or if it already exists. If set to `false`, you also need to set one of `partition_nr`, `lv_name`, `label`, or `uuid` to tell AutoYaST which device to use.

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

crypt_method

Optional, the partition will be encrypted using one of these methods:

- `luks1`: regular LUKS1 encryption.
- `luks2`: regular LUKS2 encryption.
- `pervasive_luks2`: pervasive volume encryption.
- `protected_swap`: encryption with volatile protected key.
- `secure_swap`: encryption with volatile secure key.
- `random_swap`: encryption with volatile random key.

```
<crypt_method config:type="symbol">luks1</crypt_method>
```

Encryption method selection was introduced in SUSE Linux Enterprise Server 15 SP2. To mimic the behavior of previous versions, use `luks1`.

See `crypt_key` element to learn how to specify the encryption password if needed.

When using regular LUKS encryption, it is possible to customize several aspects of the encryption using `crypt_pbkdf`, `crypt_cipher` or `crypt_key_size`, depending on the exact variant of LUKS that is used. Keep in mind that the encryption method and its corresponding settings may dramatically affect the amount of RAM needed to complete the installation process. Using regular LUKS2 with default parameters typically means that several gigabytes of RAM are needed in the system in order to encrypt the devices.

crypt_fs

Partition will be encrypted, the default is `false`. This element is deprecated. Use `crypt_method` instead.

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

crypt_key

Required if `crypt_method` has been set to a method that requires a password (that is, `luks1`, `luks2` or `pervasive_luks2`).

```
<crypt_key>xxxxxxxx</crypt_key>
```

crypt_cipher

Cipher used for LUKS encryption. This value is only honored if the value of `crypt_method` is `luks1` or `luks2`. The format and value of the string must be compatible with the `--cipher` argument of the command **`cryptsetup`**. You can find compatible ciphers in `/proc/crypto`.

```
<crypt_cipher>aes-xts-plain64</crypt_cipher>
```

crypt_key_size

Key size, in bits, used for LUKS encryption. This value is only honored if the value of `crypt_method` is `luks1` or `luks2`. The value has to be a multiple of 8. The possible key sizes are limited by the used cipher.

```
<crypt_key_size config:type="integer">256</crypt_key_size>
```

crypt_pbkdf

Password-based key derivation function used for LUKS2 encryption. This is only relevant if the `crypt_method` is `luks2`. The possible values are `pbkdf2`, `argon2i` and `argon2id`. If omitted, the device will be encrypted using the default function of the command **`cryptsetup`**. Note that both variants of Argon2 are designed to intentionally consume a large amount of memory during the encryption process. Using any of those functions or omitting this setting (which will likely result in the usage of Argon2) means that more RAM is needed to complete the installation process compared to a non-encrypted setup.

```
<crypt_pbkdf config:type="symbol">argon2id</crypt_pbkdf>
```

crypt_label

LUKS label for the encrypted device. This is only relevant if the `crypt_method` is `luks2`.

```
<crypt_label>crypt_home</crypt_label>
```

mount

You should have at least a root partition (/) and a swap partition.

```
<mount>/</mount><mount>swap</mount>
```

fstopt

Mount options for this partition; see **`man mount`** for available mount options.

```
<fstopt>ro,noatime,user,data=ordered,acl,user_xattr</fstopt>
```

label

The label of the partition. Useful when formatting the device (especially if the `mountby` parameter is set to `label`) and for identifying a device that already exists (see `create` above). See **`man e2label`** for an example.

```
<label>mydata</label>
```

uuid

The uuid of the partition. Only useful for identifying an existing device (see `create` above). The uuid cannot be enforced for new devices. (See **`man uuidgen`**.)

```
<uuid>1b4e28ba-2fa1-11d2-883f-b9a761bde3fb</uuid>
```

size

The size of the partition, for example 4G, 4500M, etc. The /boot partition and the swap partition can have `auto` as size. Then AutoYaST calculates a reasonable size. One partition can have the value `max` to use all remaining space.

You can also specify the size in percentage. So 10% will use 10% of the size of the hard disk or volume group. You can mix `auto`, `max`, `size`, and percentage as you like.

```
<size>10G</size>
```

Starting with SUSE Linux Enterprise Server 15, all values (including `auto` and `max`) can be used for resizing partitions as well.

format

Specify if AutoYaST should format the partition. If you set `create` to `true`, then you likely want this option set to `true` as well.

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

file system

Optional. The default is `btrfs` for the root partition (`/`) and `xfs` for data partitions. Specify the file system to use on this partition:

- `btrfs`
- `ext2`
- `ext3`
- `ext4`
- `fat`
- `xfs`
- `swap`

```
<filesystem config:type="symbol">ext3</filesystem>
```

mkfs_options

Optional, specify an option string for the **`mkfs`**. Only use this when you know what you are doing. (See the relevant mkfs man page for the file system you want to use.)

```
<mkfs_options>-I 128</mkfs_options>
```

partition_nr

The number of this partition. If you have set `create=false` or if you use LVM, then you can specify the partition via `partition_nr`.

```
<partition_nr config:type="integer">2</partition_nr>
```

partition_id

The `partition_id` sets the id of the partition. If you want different identifiers than 131 for Linux partition or 130 for swap, configure them with `partition_id`. The default is `131` for a Linux partition and `130` for swap.

```
<partition_id config:type="integer">131</partition_id>
```

FAT16 (MS-DOS): `6`
NTFS (MS-DOS): `7`
FAT32 (MS-DOS): `12`
Extended FAT16 (MS-DOS): `15`
DIAG, Diagnostics and firmware (MS-DOS, GPT): `18`
PPC PReP Boot partition (MS-DOS, GPT): `65`
Swap (MS-DOS, GPT, DASD, implicit): `130`
Linux (MS-DOS, GPT, DASD): `131`
Intel Rapid Start Technology (MS-DOS, GPT): `132`
LVM (MS-DOS, GPT, DASD): `142`
EFI System Partition (MS-DOS, GPT): `239`
MD RAID (MS-DOS, GPT, DASD): `253`
BIOS boot (GPT): `257`
Windows basic data (GPT): `258`
EFI (GPT): `259`
Microsoft reserved (GPT): `261`

partition_type

Optional. Allowed value is `primary`. When using an `msdos` partition table, this element sets the type of the partition to `primary`. This value is ignored when using a `gpt` partition table, because such a distinction does not exist in that case.

```
<partition_type>primary</partition_type>
```

mountby

Instead of a partition number, you can tell AutoYaST to mount a partition by `device`, `label`, `uuid`, `path` or `id`, which are the udev path and udev id (see `/dev/disk/...`).

See `label` and `uuid` documentation above. The default depends on YaST and usually is `id`.

```
<mountby config:type="symbol">label</mountby>
```

subvolumes

List of subvolumes to create for a file system of type Btrfs. This key only makes sense for file systems of type Btrfs. (See *the section called “Btrfs subvolumes”* for more information.)

If no `subvolumes` section has been defined for a partition description, AutoYaST will create a predefined set of subvolumes for the given mount point.

```
<subvolumes config:type="list">
  <path>tmp</path>
  <path>opt</path>
  <path>srv</path>
  <path>var</path>
  ...
</subvolumes>
```

create_subvolumes

Determine whether Btrfs subvolumes should be created or not. It is set to `true` by default. When set to `false`, no subvolumes will be created.

subvolumes_prefix

Set the Btrfs subvolumes prefix name. If no prefix is wanted, it must be set to an empty value:

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

It is set to @ by default.

lv_name

If this partition is on a logical volume in a volume group, specify the logical volume name here (see the `type` parameter in the drive configuration).

```
<lv_name>opt_lv</lv_name>
```

stripes

An integer that configures LVM striping. Specify across how many devices you want to stripe (spread data).

```
<stripes config:type="integer">2</stripes>
```

stripesize

Specify the size of each block in KB.

```
<stripesize config:type="integer">4</stripesize>
```

lvm_group

If this is a physical partition used by (part of) a volume group (LVM), you need to specify the name of the volume group here.

```
<lvm_group>system</lvm_group>
```

pool

`pool` must be set to `true` if the LVM logical volume should be an LVM thin pool.

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

used_pool

The name of the LVM thin pool that is used as a data store for this thin logical volume. If this is set to something non-empty, it implies that the volume is a so-called thin logical volume.

```
<used_pool>my_thin_pool</used_pool>
```

raid_name

If this physical volume is part of a RAID array, specify the name of the RAID array.

```
<raid_name>/dev/md/0</raid_name>
```

raid_options

Specify RAID options. Setting the RAID options at the `partition` level is deprecated. See *the section called “Software RAID”*.

bcache_backing_for

If this device is used as a *`bcache` backing device*, specify the name of the `bcache` device. See *the section called “ `bcache` configuration”* for further details.

```
<bcache_backing_for>/dev/bcache0</bcache_backing_for>
```

bcache_caching_for

If this device is used as a *`bcache` caching device*, specify the names of the `bcache` devices. See *the section called “ `bcache` configuration”* for further details.

```
<bcache_caching_for config:type="list"><listentry>/dev/bcache0</
listentry></bcache_caching_for>
```

resize

Starting with SUSE Linux Enterprise Server 15 resizing works with physical disk partitions and with LVM volumes

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

4.6.3.3. Btrfs subvolumes

As mentioned in *the section called "Partition configuration"*, it is possible to define a set of subvolumes for each Btrfs file system. In its simplest form, they are specified using a list of paths:

```
<subvolumes config:type="list">
  <path>usr/local</path>
  <path>tmp</path>
  <path>opt</path>
  <path>srv</path>
  <path>var</path>
</subvolumes>
```

However, it is possible to specify additional settings for each subvolume. For example, we might want to set a quota or to disable the copy-on-write mechanism. For that purpose, it is possible to expand any of the elements of the list as shown in the example below:

```
<subvolumes config:type="list">
  <listentry>usr/local</listentry>
  <listentry>
    <path>tmp</path>
    <referenced_limit>1 GiB</referenced_limit>
  </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>
```

`path`

Mount point for the subvolume.

```
<path>tmp</tmp>
```

Required. AutoYaST will ignore the subvolume if the `path` is not specified.

`copy-on-write`

Whether copy-on-write should be enabled for the subvolume.

```
<copy-on-write config:type="boolean">false</copy-on-write>
```

Optional. The default value is `false`.

`referenced_limit`

Set a quota for the subvolume.

```
<referenced_limit>1 GiB</referenced_limit>
```

Optional. The default value is `unlimited`. Btrfs supports two kinds of limits: `referenced` and `exclusive`. At this point, only the former is supported.

If there is a default subvolume used for the distribution (for example `@` in SUSE Linux Enterprise Server), the name of this default subvolume is automatically prefixed to the names of the defined subvolumes. This behavior can be disabled by setting the `subvolumes_prefix` in the *the section called "Drive configuration"* section.

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

4.6.3.4. Using the whole disk

AutoYaST allows to use a whole disk without creating any partition by setting the `disklabel` to `none` as described in *the section called "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 *the section called "Logical volume manager (LVM)"* and *the section called "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 SUSE Linux Enterprise Server 15.

4.6.3.5. 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>
      <partition>
        <filesystem config:type="symbol">xfs</filesystem>
        <mount>/data2</mount>
        <size>auto</size>
      </partition>
    </partitions>
  </drive>
</partitioning>
```

4.6.4. Advanced partitioning features

4.6.4.1. 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.

4.6.4.2. 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>
```

Check supported file system options

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

4.6.4.3. 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>
        <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>
```

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.

4.6.5. 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>
        <partition>
          <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`.

4.6.6. Software RAID

The support for software RAID devices has been greatly improved in SUSE Linux Enterprise Server 15 SP2.

If needed, see *the section called “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. 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 RAID10 configuration that uses a partition from the first disk and another one from the second disk as RAID members:

Example 4.13. RAID10 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>near_2</parity_algorithm>
      <raid_type>raid10</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. RAID10 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>near_2</parity_algorithm>
    <raid_type>raid10</raid_type>
  </raid_options>
  <use>all</use>
</drive>
```

4.6.6.1. Using the deprecated syntax

If the installer self-update feature is enabled, it is possible to partition a software RAID for SUSE Linux Enterprise Server 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 RAID10 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>near_2</parity_algorithm>
          <raid_type>raid10</raid_type>
        </raid_options>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
</partitioning>
```

4.6.6.2. RAID options

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

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

chunk_size

Can be expressed as a number with the corresponding units (for example, 32M) or just as a number. If the unit is omitted, kilobytes are used as the default unit. Do not specify `chunk_size` for RAID1. Keep in mind that `raid1` is the default type.

```
<chunk_size>4</chunk_size>
```

parity_algorithm

Possible values are:

`left_asymmetric`, `left_symmetric`, `right_asymmetric`, `right_symmetric`, `first`, `last`, `first_6`, `left_asymmetric_6`, `left_symmetric_6`, `right_asymmetric_6`, `right_symmetric_6`, `near_2`, `offset_2`, `far_2`, `near_3`, `offset_3` and `far_3`.

For backwards compatibility with previous versions of AutoYaST, the following aliases are also recognized:

`parity_first`, `parity_last`, `parity_first_6`, `n2`, `o2`, `f2`, `n3`, `o3` and `f3`.

The accepted values for each RAID depend on the RAID level (for example, `raid5`) and the number of devices in the RAID. Given that RAID0 or RAID1 do not provide any parity, do not specify this option for such devices.

```
<parity_algorithm>left_asymmetric</parity_algorithm>
```

raid_type

Possible values are: `raid0`, `raid1`, `raid5`, `raid6` and `raid10`.

```
<raid_type>raid1</raid_type>
```

The default is `raid1`.

device_order

This list contains the order of the physical devices:

```
<device_order config:type="list"><device>/dev/sdb2</device><device>/dev/
sda1</device>...</device_order>
```

This is optional, and the default is alphabetical order.

4.6.7. Multipath support

AutoYaST can handle multipath devices. 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/14945540000000000f86756dce9286158be4c6e3567e75ba5`, `/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
14945540000000000f86756dce9286158be4c6e3567e75ba5 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/14945540000000000f86756dce9286158be4c6e3567e75ba5</
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>
```

4.6.8. `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 SUSE Linux Enterprise Server, 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 bcaches -->
        <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 below.

cache_mode

Cache mode for `bcache`. Possible values are:

- `writethrough`
- `writeback`
- `writearound`
- `none`

```
<cache_mode>writethrough</cache_mode>
```

4.6.9. 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 SUSE Linux Enterprise Server 15 SP7.

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 (for example, 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_leve>raid1</raid_level>
      <metadata_raid_leve>raid1</metadata_raid_level>
    </btrfs_options>
  </drive>
</partitioning>
```

The supported data and metadata RAID levels are: `default`, `single`, `dup`, `raid0`, `raid1`, and `raid10`. By default, file system metadata is mirrored across two devices and data is striped across all of the devices. If only one device is present, metadata 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.

4.6.10. NFS configuration

AutoYaST allows to install SUSE Linux Enterprise Server onto *Network File System* (NFS) shares. To do so, you must 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 *the section called “NFS client and server”*.

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>
```

4.6.11. `tmpfs` configuration

AutoYaST supports the definition of `tmpfs` virtual file systems by setting the `type` element to `CT_TMPFS`. Each `partition` section represents a `tmpfs` file system.

Example 4.22. `tmpfs` definition

```
<partitioning config:type="list">
  <drive>
    <type config:type="symbol">CT_TMPFS</type>
    <partitions config:type="list">
      <partition>
        <mount>/srv</mount>
        <fstopt>size=512M</fstopt>
      </partition>
      <partition>
        <mount>/temp</mount>
      </partition>
    </partitions>
  <drive>
<partitioning>
```

`tmpfs` devices are different from regular file systems like Ext4 or Btrfs. Therefore, the only relevant elements are `mount`, which is mandatory, and `fstopt`. The latter is used to set file system attributes like its size limit, mode, and so on. You can find additional information about the known options in the `tmpfs` man page.

4.6.12. IBM Z specific configuration

4.6.12.1. Configuring DASD disks

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

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

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

DASD configuration

device

DASD is the only value allowed.

```
<device>DASD</dev_name>
```

dev_name

Specify the device (`dasd`*N*) you want to configure in this section.

```
<dev_name>/dev/dasda</dev_name>
```

Optional but recommended. If left out, AutoYaST tries to guess the device.

channel

Channel by which the disk is accessed.

```
<channel>0.0.0150</channel>
```

Mandatory.

diag

Enable or disable the use of `DIAG`. Possible values are `true` (enable) or `false` (disable).

```
<diagconfig:type="boolean">true</diag>
```

Optional.

4.6.12.2. Configuring zFCP disks

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

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

Each disk needs to be configured in a separate `listentry` section, which needs to provide the following elements:

controller_id

Channel number

```
<controller_id>0.0.fc00</controller_id>
```

wwpn

Worldwide port number, the target port through which the SCSI device is attached

```
<wwpn>0x500507630300c562</wwpn>
```

fcp_lun

Logical unit number

```
<fcp_lun>0x4010403200000000</fcp_lun>
```

See the IBM documentation for more information, https://www.ibm.com/docs/en/linux-on-systems?topic=wsd-configuring-devices.

4.7. iSCSI initiator overview

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

Example 4.23. 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>
```

iSCSI initiator configuration settings

initiatorname

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

version

Version of the YaST module. Default: 1.0

targets

List of targets. Each entry contains:

authmethod

Authentication method: None/CHAP

portal

Portal address

startup

Value: manual/onboot

target

Target name

iface

Interface name

4.8. Fibre channel over Ethernet configuration (FCoE)

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

Example 4.24. 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>
```

fcoe_cfg

Values: `yes`/`no`

`DEBUG` is used to enable or disable debugging messages from the fcoe service script and fcoemon.

`USE_SYSLOG` messages are sent to the system log if set to yes.

interfaces

List of network cards including the status of VLAN and FCoE configuration.

service_start

Values: `yes`/`no`

Enable or disable the start of the services `fcoe` and `lldpad` boot time.

Starting the `fcoe` service means starting the Fibre Channel over Ethernet service daemon `fcoemon`, which controls the FCoE interfaces and establishes a connection with the `lldpad` daemon.

The `lldpad` service provides the Link Layer Discovery Protocol agent daemon `lldpad`, which informs `fcoemon` about DCB (Data Center Bridging) features and configuration of the interfaces.

4.9. Country settings

Language, time zone, and keyboard settings.

Example 4.25. Language

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

language

Primary language

languages

Secondary languages separated by commas

A list of available languages can be found under `/usr/share/YaST2/data/languages`.

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

Example 4.26. Time zone

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

hwclock

Whether the hardware clock uses local time or UTC.

Values: `localtime`/`UTC`.

timezone

Time zone.

A list of available time zones can be found under `/usr/share/YaST2/data/timezone_raw.ycp`

Example 4.27. Keyboard

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

keymap

Keyboard layout

Keymap-code values or keymap-alias values are valid. A list of available entries can be found in `/usr/share/YaST2/lib/y2keyboard/keyboards.rb`. For example, `english-us, us, english-uk, uk.`

4.10. Software

4.10.1. Product selection

Starting with SUSE Linux Enterprise Server 15, all products are distributed using a single installation medium. Therefore you need to choose which product to install by using the `product` tag.

The available values for the `product` tag are:

SLES

SUSE Linux Enterprise Server

SLE_HPC

SUSE Linux Enterprise High Performance Computing

SLE_RT

SUSE Linux Enterprise Real Time

SLES_SAP

SUSE Linux Enterprise Server for SAP applications

SLED

SUSE Linux Enterprise Desktop

SUSE-manager-server

SUSE Multi-Linux Manager Server

SUSE-manager-retail-branch-server

SUSE Multi-Linux Manager for Retail

SUSE-manager-proxy

SUSE Multi-Linux Manager Proxy

Example 4.28. Explicit product selection

In the following example, SUSE Linux Enterprise Desktop is the chosen product:

```
<software>
  <products config:type="list">
    <product>SLED</product>
  </products>
</software>
```

In special cases, the medium might contain only one product. If so, you do not need to select a product explicitly as described above. AutoYaST will select the only available product automatically.

Using AutoYaST files from previous versions

If you are using or migrating an AutoYaST configuration file from an older version of SUSE Linux Enterprise Server, be aware that there are some special considerations. For details, refer to *the section called "Product selection"*.

4.10.2. Package selection with patterns and packages sections

Patterns or packages are configured like this:

Example 4.29. 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>
```

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.

It is possible to specify package and pattern names using regular expressions. In that case, AutoYaST will select all packages or patterns that match the expression. Beware that such expressions must be enclosed within slashes. In *Example 4.30, "Packages selection using a regular expression"*, all packages whose name starts with `nginx` will be selected (for example, `nginx` and `nginx-macros`).

Example 4.30. Packages selection using a regular expression

```
<software>
  <packages config:type="list">
    <package>/nginx.*/</package>
  </packages>
</software>
```

4.10.3. 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 with the SUSE packages and must install the kernel files to the same locations.

Unlike in earlier 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 4.31. Creating a package database with the additional package inst-source-utils.rpm

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

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

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

The advantage of this method is that you can keep an up-to-date repository with a 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 4.32. `add_on_products.xml`

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

```
<?xml version="1.0"?>
<add_on_products xmlns="http://www.suse.com/1.0/yast2ns"
   xmlns:config="http://www.suse.com/1.0/configns">
   <product_items config:type="list">
      <product_item>
         <name>SUSE Linux Enterprise Software Development Kit</name>
         <url>relurl:////sdk?alias=SLE_SDK</url>
         <path>/</path>
         <-- Users are asked whether to add such a product -->
         <ask_user config:type="boolean">false</ask_user>
         <-- Defines the default state of pre-selected state in case of ask_user
used. -->
         <selected config:type="boolean">true</selected>
      </product_item>
   </product_items>
</add_on_products>
```

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

Repositories served by a registration server

If you want to use add-ons from a registration server (SMT, RMT, or SCC), define them in the `suse_register` section. See *the section called “Extensions”*.

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

Example 4.33. 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:

media_url

Product URL. Can have the prefix `cd:///`, `http://`, `ftp://`, etc. This entry is mandatory.

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 `product_dir` value (see below).

product

Internal product name if the add-on is a product. The command **`zypper products`** shows the names of installed products.

alias

Repository alias name. Defined by the user.

product_dir

Optional subpath. This should only be used for multi-product media such as the SUSE Linux Enterprise Packages DVD.

priority

Sets the repository libzypp priority. Priority of 1 is the highest. The higher the number, the lower the priority. Default is 99.

ask_on_error

AutoYaST can ask the user to make add-on products, modules or extensions available instead of reporting a time-out error when no repository can be found at the given location. Set `ask_on_error` to `true` (the default is `false`).

confirm_license

The user needs to confirm the license. Default is `false`.

name

Repository name. The command **`zypper lr`** shows the names of added repositories.

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

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.

accept_unsigned_file

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

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

accept_file_without_checksum

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

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

accept_verification_failed

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

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

accept_unknown_gpg_key

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

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

accept_non_trusted_gpg_key

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

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

import_gpg_key

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

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

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

accept_unsigned_file

If set to `true`, AutoYaST will accept unsigned files such as the content file for this add-on product.

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

accept_file_without_checksum

If set to `true`, AutoYaST will accept files without a checksum in the content file for this add-on.

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

accept_verification_failed

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

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

accept_unknown_gpg_key

If `all` is set to `true`, AutoYaST will accept new GPG keys on the installation source.

```
<accept_unknown_gpg_key> <all config:type="boolean">true</all> </
accept_unknown_gpg_key>
```

Alternatively, you can define single keys:

```
<accept_unknown_gpg_key> <all config:type="boolean">false</all> <keys
config:type="list"> <keyid>3B3011B76B9D6523</keyid> lt;/keys> </
accept_unknown_gpg_key>
```

accept_non_trusted_gpg_key

This means that the key is known, but it is not trusted by you. You can trust all keys by adding:

```
<accept_non_trusted_gpg_key> <all config:type="boolean">true</all> </
accept_non_trusted_gpg_key>
```

Alternatively, you can trust specific keys:

```
<accept_non_trusted_gpg_key> <all config:type="boolean">false</all> <keys
config:type="list"> <keyid>3B3011B76B9D6523</keyid> </keys> </
accept_non_trusted_gpg_key>
```

import_gpg_key

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

```
<import_gpg_key> <all config:type="boolean">true</all> </import_gpg_key>
```

This can be done for specific keys only:

```
<import_gpg_key> <all config:type="boolean">false</all> <keys
config:type="list"> <keyid>3B3011B76B9D6523</keyid> </keys> </
import_gpg_key>
```

4.10.4. 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 4.34. Kernel selection in the control file

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

4.10.5. 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 SUSE Linux Enterprise Server. The following example shows how this can be done:

Example 4.35. 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>
```

Package removal failure

Note that it is not possible to remove a package that is part of a pattern (see *the section called "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.
```

4.10.6. Installing recommended packages and patterns

AutoYaST enables you to control which *recommended* packages and patterns are installed. There are three options:

- Install all recommended packages and patterns
- Install only required packages and patterns
- Install recommended packages, ignore recommended patterns

Set the `install_recommended` flag to `true` in the configuration file to install all recommended packages and patterns.

If you want a minimal installation, and to install only *required* packages and patterns, set the flag to `false`.

Omit the flag from the configuration file to install only recommended packages, and ignore all recommended patterns. Note that this flag only affects a fresh installation and will be ignored during an upgrade.

The `install_recommended` flag affects only the installation process

Keep in mind that the flag influences only the package resolver during the installation process and does not change any settings in `/etc/zypp/zypp.conf`. Therefore, the package resolving in the running system is not affected by this AutoYaST setting.

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

4.10.7. 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>
```

4.10.8. 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>
```

4.10.9. 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>
```

4.11. 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 4.36. 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>
```

stop_on_solver_conflict

Halt installation if there are package dependency issues.

modified

Create backups of modified files.

sysconfig

Create backup of `/etc/sysconfig` directory.

remove_old

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 `Upgrade` 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 `Upgrade` 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.

4.12. 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 4.37. 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>
```

4.13. Network configuration

4.13.1. Configuration Workflow

Network configuration is mainly used to connect a single workstation to an Ethernet-based LAN. It is commonly configured before AutoYaST starts, to fetch the profile from a network location. This network configuration is usually done through **`linuxrc`**

The `linuxrc` program

For a detailed description of how **`linuxrc`** works and its keywords, see *Appendix C, Advanced **`linuxrc`** options*.

By default, YaST copies the network settings that were used during the installation into the final, installed system. This configuration is merged with the one defined in the AutoYaST profile.

AutoYaST settings have higher priority than any existing configuration files. YaST will write `ifcfg-*` files based on the entries in the profile without removing old ones. If the DNS and routing section is empty or missing, YaST will keep any pre-existing values. Otherwise, it applies the settings from the profile file.

Use AutoYaST network settings during installation

Since SUSE Linux Enterprise Server 15.3, writing the configuration based on the profile happens at the end of the first stage.

However, if network settings are needed during the installation, you can force AutoYaST to write and apply them before registration takes place by setting the `setup_before_proposal` option to `true`. Asking AutoYaST to set up the network in the early stages is useful when installation on a network is needed, but the configuration is too complex to define it using linuxrc (see *the section called "Auto-installing a single system"*).

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

If the configuration is written at the end of installation, it will not be applied until the system is rebooted.

Network settings and service activation are defined under the `profilenetworking` global resource.

4.13.2. The Network Resource

Example 4.38. Network configuration

```
<networking>
  <dns>
    <dhcp_hostname config:type="boolean">true</dhcp_hostname>
    <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>
  <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>192.168.100.0/24</destination>
        <device>eth1</device>
        <extrapara>scope link src 192.168.100.100 table one</extrapara>
        <gateway>-</gateway>
      </route>
      <route>
        <destination>default</destination>
        <device>eth1</device>
        <gateway>192.168.100.1</gateway>
      </route>
      <route>
        <destination>default</destination>
        <device>lo</device>
        <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 z Systems-specific device settings.
- `net-udev` enumerates the udev rules used to set persistent names.

Additionally, there are a few elements that allow modification of how the network configuration is applied:

backend

Selects the network back-end to be used. Supported values are `wicked`, `network_manager` or `none`, the latter of which will disable the network service.

```
<backend>network_manager</backend>
```

keep_install_network

As described in *the section called "Configuration Workflow"*, by default, AutoYaST merges the network configuration from the running system with the one defined in the profile. If you want to use only the configuration from the profile, set this element to `false`. The value is `true` by default.

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

managed

Determines whether to use NetworkManager instead of Wicked.

Deprecated. Use `backend` instead.

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

start_immediately

Forces AutoYaST to restart the network just after writing the configuration.

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

setup_before_proposal

Use the network configuration defined in the profile during the installation process. Otherwise, AutoYaST relies on the configuration set by **`linuxrc`**.

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

strict_IP_check_timeout

After setting up the network, AutoYaST checks whether the assigned IP address is duplicated. In that case, it shows a warning whose timeout in seconds is controlled by this element. If it is set to `0`, the installation is stopped.

```
<strict_IP_check_timeout config:type="integer">5</strict_IP_check_timeout>
```

virt_bridge_proposal

AutoYaST configures a bridge when a virtualization package is selected to be installed (for example, Xen, QEMU or KVM). You can disable this behavior by setting this element to `false`.

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

IPv6 address support

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

4.13.3. 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.

`bootproto`

Boot protocol used by the interface. Possible values:

- `static` for statically assigned addresses. It is required to specify the IP using the `ipaddr` element.
- `dhcp4`, `dhcp6` or `dhcp` for setting the IP address with DHCP (IPv4, IPv6 or any).
- `dhcp+autoip` to get the IPv4 configuration from *Zeroconf* and get IPv6 from DHCP.
- `autoip` to get the IPv4 configuration from *Zeroconf*.
- `ibft` to get the IP address using the iBFT protocol.
- `none` to skip setting an address. This value is used for bridges and bonding ports.

Required.

`broadcast`

Broadcast IP address.

Used only with `static` boot protocol.

`device`

Device name.

Deprecated. Use `name` instead.

`name`

Device name, for example: `eth0`.

Required.

`lladdr`

Link layer address (MAC address).

Optional.

`ipaddr`

IP address assigned to the interface.

Used only with `static` boot protocol. It can include a network prefix, for example: `192.168.1.1/24`.

`remote_ipaddr`

Remote IP address for point-to-point connections.

Used only with `static` boot protocol.

`netmask`

Network mask, for example: `255.255.255.0`.

Deprecated. Use `prefixlen` instead or include the network prefix in the `ipaddr` element.

`network`

Network IP address.

Deprecated. Use `ipaddr` with `prefixlen` instead.

`prefixlen`

Network prefix, for example: `24`.

Used only with `static` boot protocol.

startmode

When to bring up an interface. Possible values are:

- `hotplug` when the device is plugged in. Useful for USB network cards, for example.
- `auto` when the system boots. `onboot` is a deprecated alias.
- `ifplugd` when the device is managed by the `ifplugd` daemon.
- `manual` when the device is supposed to be started manually.
- `nfsroot` when the device is needed to mount the root file system, for example, when `/` is on an NFS volume.
- `off` to never start the device.

ifplugd_priority

Priority for `ifplugd` daemon. It determines in which order the devices are activated.

Used only with `ifplugd` start mode.

usercontrol

Parameter is no longer used.

Deprecated.

bonding_slaveX

Name of the bonding device.

Required for bonding devices. X is replaced by a number starting from 0, for example `bonding_slave0`. Each port needs to have a unique number.

bonding_module_opts

Options for bonding device.

Used only with `bond` device.

mtu

Maximum transmission unit for the interface.

Optional.

ethtool_options

Ethtool options during device activation.

Optional.

`zone`

Firewall zone name which the interface is assigned to.

Optional.

`vlan_id`

Identifier used for this VLAN.

Used only with a `vlan` device.

`etherdevice`

Device to which VLAN is attached.

Used only with a `vlan` device and required for it.

`bridge`

`yes` if interface is a bridge.

Deprecated. It is inferred from other attributes.

`bridge_ports`

Space-separated list of bridge ports, for example, `eth0 eth1`.

Used only with a `bridge` device and required for it.

`bridge_stp`

Spanning tree protocol. Possible values are `on` (when enabled) and `off` (when disabled).

Used only with a `bridge` device.

`bridge_forward_delay`

Forward delay for bridge, for example: `15`.

Used only with `bridge` devices. Valid values are between `4` and `30`.

`aliases`

Additional IP addresses. See *the section called “Assigning multiple IP addresses”*.

Example 4.39. Bonding interface configuration

```
<networking>
  <setup_before_proposal config:type="boolean">false</setup_before_proposal>
  <keep_install_network config:type="boolean">false</keep_install_network>
  <interfaces config:type="list">
    <interface>
      <bonding_master>yes</bonding_master>
      <bonding_module_opts>mode=active-backup miimon=100</bonding_module_opts>
      <bonding_slave0>eth1</bonding_slave0>
      <bonding_slave1>eth2</bonding_slave1>
      <bootproto>static</bootproto>
      <name>bond0</name>
      <ipaddr>192.168.1.61</ipaddr>
      <prefixlen>24</prefixlen>
      <startmode>auto</startmode>
    </interface>
    <interface>
      <bootproto>none</bootproto>
      <name>eth1</name>
      <startmode>auto</startmode>
    </interface>
    <interface>
      <bootproto>none</bootproto>
      <name>eth2</name>
      <startmode>auto</startmode>
    </interface>
  </interfaces>
  <net-udev config:type="list">
    <rule>
      <name>eth1</name>
      <rule>ATTR{address}</rule>
      <value>dc:e4:cc:27:94:c7</value>
    </rule>
    <rule>
      <name>eth2</name>
      <rule>ATTR{address}</rule>
      <value>dc:e4:cc:27:94:c8</value>
    </rule>
  </net-udev>
</networking>
```

Example 4.40. Bridge interface configuration

```
<interfaces config:type="list">
  <interface>
    <name>br0</name>
    <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>
    <prefixlen>24</prefixlen>
    <startmode>auto</startmode>
  </interface>
  <interface>
    <name>eth0</name>
    <bootproto>none</bootproto>
    <startmode>hotplug</startmode>
  </interface>
  <interface>
    <name>eth1</name>
    <bootproto>none</bootproto>
    <startmode>hotplug</startmode>
  </interface>
</interfaces>
```

4.13.4. Assigning multiple IP addresses

AutoYaST makes it possible to assign multiple IP addresses to the same interface. They are specified using an `aliases` element that contains an `aliasX` entry for each address.

Each entry supports the following elements:

IPADDR

Additional IP address. It can include a network prefix, for example: `192.168.1.1/24`.

PREFIXLEN

Network prefix, for example: `24`.

NETMASK

Netmask of the address.

Deprecated. Use `PREFIXLEN` instead or include the network prefix in the `IPADDR` element.

LABEL

Label of the address.

Case-sensitive elements

Keep in mind that for historical reasons, the `IPADDR`, `PREFIXLEN`, `LABEL` and `NETMASK` elements within the `aliases` section are case-sensitive.

Example 4.41. Multiple IP Addresses

```
<interfaces config:type="list">
  <interface>
    <name>br0</name>
    <bootproto>static</bootproto>
    <ipaddr>192.168.1.100</ipaddr>
    <prefixlen>24</prefixlen>
    <startmode>auto</startmode>
    <aliases>
      <alias0>
        <IPADDR>192.168.1.101</IPADDR>
        <PREFIXLEN>24</PREFIXLEN>
        <LABEL>http</LABEL>
      </alias0>
      <alias1>
        <IPADDR>192.168.2.100</IPADDR>
        <PREFIXLEN>24</PREFIXLEN>
        <LABEL>extra</LABEL>
      </alias1>
    </aliases>
  </interface>
</interfaces>
```

4.13.5. 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.

name

Network interface name, for example `eth3`. (Required.)

rule

`ATTR{address}` for a MAC-based rule, `KERNELS` for a bus-ID-based rule. (Required.)

value

For example: `f0:de:f1:6b:da:69` for a MAC rule, `0000:00:1c.1 or 0.0.0700` for a bus ID rule. (Required.)

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 4.42. 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>
```

4.13.6. Domain name system

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

hostname

Host name, excluding the domain name part. For example: *foo* instead of *foo.bar*. The Linux kernel allows you to use the fully qualified domain name (FQDN) in place of the host name, and so does YaST. However, this is not the correct usage in the dns section of YaST. The resolver should determine the FQDN. (See "THE FQDN" section of **`man 1 hostname`** for information on how FQDNs are resolved.)

If a host name is not specified and is not assigned by a DHCP server (see `dhcp_hostname`), AutoYaST will generate `install` as the host name.

nameservers

List of name servers. Example:

```
<nameservers config:type="list">
  <nameserver>192.168.1.116</nameserver>
  <nameserver>192.168.1.117</nameserver>
</nameservers>
```

searchlist

Search list for host name lookup.

```
<searchlist config:type="list">
  <search>example.com</search>
</searchlist>
```

Optional.

dhcp_hostname

Specifies whether the host name must be taken from DHCP or not.

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

4.13.7. Routing

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

ipv4_forward

Optional: Whether IP forwarding must be enabled for IPv4.

ipv6_forward

Optional: Whether IP forwarding must be enabled for IPv6.

routes

Optional: List of routes.

The following settings describe how routes are defined.

destination

Required: Route destination. An address prefix can be specified, for example: `192.168.122.0/24`.

The heading `default` can be used to indicate that the route is the default gateway in the same address family (IPv4 or IPv6) as the gateway.

device

Required: Interface associated to the route.

gateway

Optional: Gateway's IP address.

netmask

(Deprecated.) Destination's netmask.

Specifying the prefix as part of the `destination` value is preferred.

extrapara

Optional: Further route options like `metric`, `mtu` or `table`.

Example 4.43. Network routing configuration

```
<routing>
  <ipv4_forward config:type="boolean">true</ipv4_forward>
  <ipv6_forward config:type="boolean">true</ipv6_forward>
  <routes config:type="list">
    <route>
      <destination>192.168.100.0/24</destination>
      <device>eth1</device>
      <extrapara>scope link src 192.168.100.100 table one</extrapara>
    </route>
    <route>
      <destination>default</destination>
      <device>eth1</device>
      <gateway>192.168.100.1</gateway>
    </route>
    <route>
      <destination>default</destination>
      <device>lo</device>
      <gateway>192.168.5.1</gateway>
    </route>
  </routes>
</routing>
```

4.13.8. s390 options

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

type

`qeth`, `ctc` or `iucv`.

chanids

channel IDs, separated by a colon (preferred) or a space

```
<chanids>0.0.0700:0.0.0701:0.0.0702</chanids>
```

layer2

```
<layer2 config:type="boolean">true</layer2>
```

boolean; default: `false`

portname

QETH port name (deprecated since SLE 12 SP2)

protocol

Optional: CTC / LCS protocol, a small number (as a string)

```
<protocol>1</protocol>
```

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.

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.

4.14. 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 4.44. 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>
```

Note

The proxy settings will be written during the installation when the network configuration is forced to be written before the proposal, or when the proxy settings are given through **`linuxrc`**.

4.15. 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 4.45. Network configuration: NIS

```
<nis>
 <nis_broadcast config:type="boolean">true</nis_broadcast>
 <nis_broken_server config:type="boolean">true</nis_broken_server>
 <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>
```

4.16. NIS server

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

Example 4.46. NIS server configuration

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

Attribute, Values, Description

`domain`

NIS domain name.

maps_to_serve

List of maps which are available for the server.

Values: auto.master, ethers, group, hosts, netgrp, networks, passwd, protocols, rpc, services, shadow

merge_passwd

Select if your passwd file should be merged with the shadow file (only possible if the shadow file exists).

Value: true/false

mingid

Minimum GID to include in the user maps.

minuid

Minimum UID to include in the user maps.

nopush

Do not push the changes to worker servers. (Useful if there are none).

Value: true/false

pwd_chfn

YPPWD_CHFN - allow changing the full name

Value: true/false

pwd_chsh

YPPWD_CHSH - allow changing the login shell

Value: true/false

pwd_srcdir

YPPWD_SRCDIR - source directory for passwd data

Default: `/etc`

securenets

List of allowed hosts to query the NIS server

A host address will be allowed if network is equal to the bitwise AND of the host's address and the netmask.

The entry with netmask 255.0.0.0 and network 127.0.0.0 must exist to allow connections from the local host.

Entering netmask 0.0.0.0 and network 0.0.0.0 gives access to all hosts.

`server_type`

Select whether to configure the NIS server as a master or a worker or not to configure a NIS server.

Values: master, slave, none

`slaves`

List of host names to configure as NIS server workers.

`start_ypbind`

This host is also a NIS client (only when client is configured locally).

Value: true/false

`start_yppasswdd`

Also start the password daemon.

Value: true/false

`start_ypxfrd`

Also start the map transfer daemon. Fast Map distribution; it will speed up the transfer of maps to the workers.

Value: true/false

4.17. 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 4.47. /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>
```

4.18. Windows domain membership

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

Example 4.48. Samba client configuration

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

Attribute, Values, Description

`disable_dhcp_hostname`

Do not allow DHCP to change the host name.

Value: true/false

`global/security`

Kind of authentication regime (domain technology or Active Directory server (ADS)).

Value: ADS/domain

`global/usershare_allow_guests`

Sharing guest access is allowed.

Value: No/Yes

global/usershare_max_shares

Max. number of shares from `smb.conf`.

0 means that shares are not enabled.

global/workgroup

Workgroup or domain name.

winbind

Using winbind.

Value: true/false

4.19. Samba server

Configuration of a simple Samba server.

Example 4.49. Samba server configuration

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

Attribute, Values, Description

accounts

List of Samba accounts.

backend

List of available back-ends.

Value: `true`/`false`.

config

Setting additional user-defined parameters in `/etc/samba/smb.conf`.

The example shows parameters in the `global` section of `/etc/samba/smb.conf`.

service

Samba service starts during boot.

Value: `Enabled`/`Disabled`.

trustdom/

Trusted Domains.

A map of two maps (keys: `establish, revoke`). Each map contains entries in the format `key: domainnamevalue: password`.

version

Samba version.

Default: 2.11.

4.20. 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*.

Using ldaps://

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

4.21. 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 SUSE Linux Enterprise Server15 SP7 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 SUSE Linux Enterprise Server onto NFS shares, refer to *the section called “NFS configuration”*.

Example 4.50. 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 4.51. 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>
```

4.22. NTP client

NTP client profile incompatible

Starting with SUSE Linux Enterprise Server 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 SUSE Linux Enterprise Server versions to be compatible with version 15 and newer.

Following is an example of the NTP client configuration:

Example 4.52. 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.

❸ `iburst` speeds up the initial time synchronization for the specific time source after `chronyd` is started.

❹ 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.

❺ 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:

```
<ntp-server>
  <address>2001:418:3ff::1:53</address>
</ntp-server>

<ntp-server>
  <address>2.pool.ntp.org</address>
</ntp-server>
```

4.23. 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 4.53. 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>
```

4.24. 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 4.54. 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>
            <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>
          <KEY>ServerAdmin</KEY>
          <VALUE>second_server@suse.de</VALUE>
        </listentry>
        <listentry>
          <KEY>_SECTION</KEY>
          <SECTIONNAME>Directory</SECTIONNAME>
          <SECTIONPARAM>/srv/www/htdocs</SECTIONPARAM>
          <VALUE config:type="list">
            <listentry>
              <KEY>AllowOverride</KEY>
              <VALUE>None</VALUE>
            </listentry>
            <listentry>
              <KEY>Require</KEY>
              <VALUE>all granted</VALUE>
            </listentry>
          </VALUE>
        </listentry>
      </VALUE>
    </hosts_entry>
  </hosts>
  <modules config:type="list">
    <module_entry>
      <change>enable</change>
      <name>socache_shmcb</name>
      <userdefined config:type="boolean">true</userdefined>
    </module_entry>
    <module_entry>
      <change>enable</change>
      <name>reqtimeout</name>
      <userdefined config:type="boolean">true</userdefined>
    </module_entry>
    <module_entry>
      <change>enable</change>
      <name>authn_core</name>
      <userdefined config:type="boolean">true</userdefined>
    </module_entry>
    <module_entry>
      <change>enable</change>
      <name>authz_core</name>
      <userdefined config:type="boolean">true</userdefined>
    </module_entry>
  </modules>
  <service config:type="boolean">true</service>
  <version>2.9</version>
</http-server>
```

List Name, List Elements, Description

Listen

List of host `Listen` settings

PORT

port address

ADDRESS

Network address. All addresses will be taken if this entry is empty.

hosts

List of Hosts configuration

Key

Host name; <KEY>`main`</KEY> defines the main hosts, for example <KEY>`192.168.43.2/secondserver.suse.de`</KEY>

VALUE

List of different values describing the host.

modules

Module list. Only user-defined modules need to be described.

name

Module name

userdefined

For historical reasons, it is always set to `true`.

change

For historical reasons, it is always set to `enable`.

Element, Description, Comment

version

Version of used Apache server

Only for information. Default 2.9

service

Enable Apache service

Optional. Default: false

Firewall

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

4.25. Squid server

Squid is a caching and forwarding Web proxy.

Example 4.55. Squid server configuration

```
<squid>
  <acls config:type="list">
    <listentry>
      <name>QUERY</name>
      <options config:type="list">
        <option>cgi-bin \?</option>
      </options>
      <type>urlpath_regex</type>
    </listentry>
    <listentry>
      <name>apache</name>
      <options config:type="list">
        <option>Server</option>
        <option>^Apache</option>
      </options>
      <type>rep_header</type>
    </listentry>
    <listentry>
      <name>all</name>
      <options config:type="list">
        <option>0.0.0.0/0.0.0.0</option>
      </options>
      <type>src</type>
    </listentry>
    <listentry>
      <name>manager</name>
      <options config:type="list">
        <option>cache_object</option>
      </options>
      <type>proto</type>
    </listentry>
    <listentry>
      <name>localhost</name>
      <options config:type="list">
        <option>127.0.0.1/255.255.255.255</option>
      </options>
      <type>src</type>
    </listentry>
    <listentry>
      <name>to_localhost</name>
      <options config:type="list">
        <option>127.0.0.0/8</option>
      </options>
      <type>dst</type>
    </listentry>
    <listentry>
      <name>SSL_ports</name>
      <options config:type="list">
        <option>443</option>
      </options>
      <type>port</type>
    </listentry>
    <listentry>
      <name>Safe_ports</name>
      <options config:type="list">
        <option>80</option>
      </options>
      <type>port</type>
    </listentry>
    <listentry>
      <name>Safe_ports</name>
      <options config:type="list">
        <option>21</option>
      </options>
      <type>port</type>
    </listentry>
    <listentry>
```

```
      <name>Safe_ports</name>
      <options config:type="list">
        <option>443</option>
      </options>
      <type>port</type>
    </listentry>
    <listentry>
      <name>Safe_ports</name>
      <options config:type="list">
        <option>70</option>
      </options>
      <type>port</type>
    </listentry>
    <listentry>
      <name>Safe_ports</name>
      <options config:type="list">
        <option>210</option>
      </options>
      <type>port</type>
    </listentry>
    <listentry>
      <name>Safe_ports</name>
      <options config:type="list">
        <option>1025-65535</option>
      </options>
      <type>port</type>
    </listentry>
    <listentry>
      <name>Safe_ports</name>
      <options config:type="list">
        <option>280</option>
      </options>
      <type>port</type>
    </listentry>
    <listentry>
      <name>Safe_ports</name>
      <options config:type="list">
        <option>488</option>
      </options>
      <type>port</type>
    </listentry>
    <listentry>
      <name>Safe_ports</name>
      <options config:type="list">
        <option>591</option>
      </options>
      <type>port</type>
    </listentry>
    <listentry>
      <name>Safe_ports</name>
      <options config:type="list">
        <option>777</option>
      </options>
      <type>port</type>
    </listentry>
    <listentry>
      <name>CONNECT</name>
      <options config:type="list">
        <option>CONNECT</option>
      </options>
      <type>method</type>
    </listentry>
  </acls>
  <http_accesses config:type="list">
    <listentry>
      <acl config:type="list">
        <listentry>manager</listentry>
        <listentry>localhost</listentry>
      </acl>
```

```
        <allow config:type="boolean">true</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>manager</listentry>
        </acl>
        <allow config:type="boolean">false</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>!Safe_ports</listentry>
        </acl>
        <allow config:type="boolean">false</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>CONNECT</listentry>
          <listentry>!SSL_ports</listentry>
        </acl>
        <allow config:type="boolean">false</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>localhost</listentry>
        </acl>
        <allow config:type="boolean">true</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>all</listentry>
        </acl>
        <allow config:type="boolean">false</allow>
      </listentry>
    </http_accesses>
    <http_ports config:type="list">
      <listentry>
        <host/>
        <port>3128</port>
        <transparent config:type="boolean">false</transparent>
      </listentry>
    </http_ports>
    <refresh_patterns config:type="list">
      <listentry>
        <case_sensitive config:type="boolean">true</case_sensitive>
        <max>10080</max>
        <min>1440</min>
        <percent>20</percent>
        <regexp>^ftp:</regexp>
      </listentry>
      <listentry>
        <case_sensitive config:type="boolean">true</case_sensitive>
        <max>1440</max>
        <min>1440</min>
        <percent>0</percent>
        <regexp>^gopher:</regexp>
      </listentry>
      <listentry>
        <case_sensitive config:type="boolean">true</case_sensitive>
        <max>4320</max>
        <min>0</min>
        <percent>20</percent>
        <regexp>.</regexp>
      </listentry>
    </refresh_patterns>
    <service_enabled_on_startup config:type="boolean">true</
service_enabled_on_startup>
    <settings>
      <access_log config:type="list">
        <listentry>/var/log/squid/access.log</listentry>
```

```
      </access_log>
      <cache_dir config:type="list">
        <listentry>ufs</listentry>
        <listentry>/var/cache/squid</listentry>
        <listentry>100</listentry>
        <listentry>16</listentry>
        <listentry>256</listentry>
      </cache_dir>
      <cache_log config:type="list">
        <listentry>/var/log/squid/cache.log</listentry>
      </cache_log>
      <cache_mem config:type="list">
        <listentry>8</listentry>
        <listentry>MB</listentry>
      </cache_mem>
      <cache_mgr config:type="list">
        <listentry>webmaster</listentry>
      </cache_mgr>
      <cache_replacement_policy config:type="list">
        <listentry>lru</listentry>
      </cache_replacement_policy>
      <cache_store_log config:type="list">
        <listentry>/var/log/squid/store.log</listentry>
      </cache_store_log>
      <cache_swap_high config:type="list">
        <listentry>95</listentry>
      </cache_swap_high>
      <cache_swap_low config:type="list">
        <listentry>90</listentry>
      </cache_swap_low>
      <client_lifetime config:type="list">
        <listentry>1</listentry>
        <listentry>days</listentry>
      </client_lifetime>
      <connect_timeout config:type="list">
        <listentry>2</listentry>
        <listentry>minutes</listentry>
      </connect_timeout>
      <emulate_httpd_log config:type="list">
        <listentry>off</listentry>
      </emulate_httpd_log>
      <error_directory config:type="list">
        <listentry/>
      </error_directory>
      <ftp_passive config:type="list">
        <listentry>on</listentry>
      </ftp_passive>
      <maximum_object_size config:type="list">
        <listentry>4096</listentry>
        <listentry>KB</listentry>
      </maximum_object_size>
      <memory_replacement_policy config:type="list">
        <listentry>lru</listentry>
      </memory_replacement_policy>
      <minimum_object_size config:type="list">
        <listentry>0</listentry>
        <listentry>KB</listentry>
      </minimum_object_size>
    </settings>
  </squid>
```

Attribute, Values, Description

`acls`

List of Access Control Settings (ACLs).

Each list entry contains the name, type, and additional options. Use the YaST Squid configuration module to get an overview of possible entries.

`http_accesses`

In the Access Control table, access can be denied or allowed to ACL Groups.

If there are more ACL Groups in one definition, access will be allowed or denied to members who belong to all ACL Groups at the same time.

The Access Control table is checked in the order listed here. The first matching entry is used.

`http_ports`

Define all ports where Squid will listen for clients' HTTP requests.

`Host` can contain a host name or IP address or remain empty.

`transparent` disables PMTU discovery when transparent.

`refresh_patterns`

Refresh patterns define how Squid treats the objects in the cache.

The refresh patterns are checked in the order listed here. The first matching entry is used.

`Min` determines how long (in minutes) an object should be considered fresh if no explicit expiry time is given. `Max` is the upper limit of how long objects without an explicit expiry time will be considered fresh. `Percent` is the percentage of the object's age (time since last modification). An object without an explicit expiry time will be considered fresh.

`settings`

Map of all available general parameters with default values.

Use the YaST Squid configuration module to get an overview about possible entries.

`service_enabled_on_startup`

Squid service start when booting.

Value: true/false

4.26. FTP server

Configure your FTP Internet server settings.

Example 4.56. 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

`AnonAuthen`

Enable/disable anonymous and local users.

Authenticated Users Only: 1; Anonymous Only: 0; Both: 2

`AnonCreatDirs`

Anonymous users can create directories.

Values: YES/NO

`AnonReadOnly`

Anonymous users can upload.

Values: YES/NO

`AnonMaxRate`

The maximum data transfer rate permitted for anonymous clients.

KB/s

AntiWarez

Disallow downloading of files that were uploaded but not validated by a local admin.

Values: YES/NO

Banner

Specify the name of a file containing the text to display when someone connects to the server.

CertFile

DSA certificate to use for SSL-encrypted connections

This option specifies the location of the DSA certificate to use for SSL-encrypted connections.

ChrootEnable

When enabled, local users will by default be placed in a `chroot` jail in their home directory after login.

Warning: This option has security implications. Values: YES/NO

EnableUpload

If enabled, FTP users can upload.

To allow anonymous users to upload, enable `AnonReadOnly`. Values: YES/NO

FTPUser

Defines the anonymous FTP user.

FtpDirAnon

FTP directory for anonymous users.

Specify a directory which is used for anonymous FTP users.

FtpDirLocal

FTP directory for authenticated users.

Specify a directory which is used for FTP authenticated users.

LocalMaxRate

The maximum data transfer rate permitted for local authenticated users.

KB/s

MaxClientsNumber

The maximum number of clients allowed to connect.

MaxClientsPerIP

Defines the maximum number of clients for one IP.

This limits the number of clients allowed to connect from a single source Internet address.

MaxIdleTime

The maximum time (timeout) a remote client may wait between FTP commands.

Minutes

PasMaxPort

Maximum value for a port range for passive connection replies.

`PassiveMode` needs to be set to YES.

PasMinPort

Minimum value for a port range for passive connection replies.

`PassiveMode` needs to be set to YES.

PassiveMode

Enable Passive Mode

Value: YES/NO

SSL

Security Settings

Disable TLS/SSL: 0; Accept TLS/SSL: 1; Refuse Connections Without TLS/SSL: 2

SSLEnable

If enabled, SSL connections are allowed.

Value: YES/NO

SSLv2

If enabled, SSL version 2 connections are allowed.

Value: YES/NO

SSLv3

If enabled, SSL version 3 connections are allowed.

Value: YES/NO

StartDaemon

How the FTP daemon will be started.

Manually: 0; when booting: 1; via systemd socket: 2

TLS

If enabled, TLS connections are allowed.

Value: YES/NO

Umask

File creation mask, in the format (umask for files):(umask for directories).

For example `177:077` if you feel paranoid.

UmaskAnon

The value to which the umask for file creation is set for anonymous users.

To specify octal values, remember the "0" prefix, otherwise the value will be treated as a base-10 integer.

UmaskLocal

Umask for authenticated users.

To specify octal values, remember the "0" prefix, otherwise the value will be treated as a base-10 integer.

VerboseLogging

When enabled, all FTP requests and responses are logged.

Value: YES/NO

`VirtualUser`

By using virtual users, FTP accounts can be administrated without affecting system accounts.

Value: YES/NO

Firewall

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

4.27. 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 4.57. TFTP server configuration:

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

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).

4.28. 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 4.58. Enabling firstboot workflow

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

4.29. Security settings

Using the features of this module, you can 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 4.59. 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>
  <selinux_mode>permissive</selinux_mode>
  <lsm_select>selinux</lsm_select>
</security>
```

4.29.1. Password settings options

Use the `<pass_*` resources to change various password settings, such as minimum password length, password expiration, and more.

Use the `<encryption>` resource to activate one of the encryption methods currently supported. If not set, `sha512` is configured.

You can use one of the following encryption methods:

- `md5` — allows longer passwords with 128-bit hash value
- `sha256` or `sha512` — widely used secure hash algorithm

- `des` — we do not recommend using this encryption method because of insufficient security

4.29.2. Boot settings

Use the security resource, to change various boot settings.

How to interpret `Ctrl`– `Alt`– `Delete` ?

When someone at the console has pressed the `Ctrl`–`Alt`–`Delete` 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.

4.29.3. Login settings

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

4.29.4. New user settings (`useradd` settings)

Set the minimum and maximum possible user and group IDs.

4.29.5. Linux Security Module (LSM) settings

In SUSE Linux Enterprise 15 SP4 and up, the installation control file has a new option, `<lsm_select>` for configuring which major Linux Security Module (LSM) will be activated by default after installation: AppArmor, SELinux, or none.

selinux_mode

Optional. Configure the SELinux mode. Values: `permissive`, `enforcing` and `disabled`.

lsm_select

Optional. Major Linux Security Module to be selected during installation. Values: `selinux`, `apparmor`, or `none`.

4.29.6. Using OpenSCAP security policies

YaST allows for system hardening using OpenSCAP security policies. Checking and applying a security policy happens in two phases:

- At installation, YaST checks a subset of the security policy rules, especially those that are hard to fix after the installation, such as encrypting the file system. If the system described in

the profile does not comply with any of these rules, AutoYaST will report the problems and abort the installation.

- Additionally, AutoYaST installs and configures the **ssg-apply** tool. During first boot, **ssg-apply** can be run to scan the system and, optionally, remediate system to meet the selected policy.

Availability in SUSE Linux Enterprise 15 SP4

This feature is available for SUSE Linux Enterprise 15 SP4 GM via self-update or using the QU2 media. Make sure to enable updates during installation with `<install_updates t="boolean">true</install_updates>` in the `<suse_register>` section (see *the section called "System registration and extension selection"*).

If you install without an internet connection, add the `Basesystem` module from the QU2 medium to the `<add_on_products>` section:

```
<listentry t="map">
  <media_url>relurl://</media_url>
  <product>sle-module-basesystem</product>
  <product_dir>/Module-Basesystem</product_dir>
</listentry>
```

For more information, refer to *the section called "Installing additional/customized packages or products"*.

The `security_policy` section selects a security policy and configures **ssg-apply**.

policy

Selects the security policy to check or apply. Currently, only the Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) is supported. Use the name `stig` to refer to this policy. This element is mandatory.

action

Specify what **ssg-apply** should do during first boot.

- `scan`: scan the system during first boot. This is the default behavior.
- `remediate`: scan and remediate the system to comply with the selected policy.
- `none`: configure but do not run **ssg-apply** during first boot. This option is useful if you want to modify the policy before hardening your system.

Example 4.60. Select the Defense Information Systems Agency Security Technical Implementation Guide

The following excerpt instructs AutoYaST to check the DISA STIG policy and remediate the system during the first boot.

```
<security>
  <security_policy>
    <policy>stig</policy>
    <action>remediate</action>
  </security_policy>
</security>
```

4.30. Linux audit framework (LAF)

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

Example 4.61. LAF configuration

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

Attribute, Values, Description

`auditd/flush`

Describes how to write the data to disk.

If set to `INCREMENTAL` the Frequency parameter tells how many records to write before issuing an explicit flush to disk. `NONE` means: no special effort is made to flush data, `DATA`: keep data portion synchronized, `SYNC`: keep data and metadata fully synchronized.

`auditd/freq`

This parameter tells how many records to write before issuing an explicit flush to disk.

The parameter `flush` needs to be set to `INCREMENTAL`.

`auditd/log_file`

The full path name to the log file.

auditd/log_fomat

How much information needs to be logged.

Set RAW to log all data (store in a format exactly as the kernel sends it) or NOLOG to discard all audit information instead of writing it to disk (does not affect data sent to the dispatcher).

auditd/max_log_file

How much information needs to be logged.

Unit: Megabytes

auditd/num_logs

Number of log files.

max_log_file_action needs to be set to ROTATE

auditd/max_log_file_action

What happens if the log capacity has been reached.

If the action is set to ROTATE the Number of Log Files specifies the number of files to keep. Set to SYSLOG, the audit daemon will write a warning to the system log. With SUSPEND the daemon stops writing records to disk. IGNORE means do nothing, KEEP_LOGS is similar to ROTATE, but log files are not overwritten.

auditd/name_format

Computer Name Format describes how to write the computer name to the log file.

If USER is set, the user-defined name is used. NONE means no computer name is inserted. HOSTNAME uses the name returned by the 'gethostname' syscall. FQD uses the fully qualified domain name.

rules

Rules for auditctl

You can edit the rules manually, which we only recommend for advanced users. For more information about all options, see **man auditctl**.

4.31. Users and groups

4.31.1. 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 4.62. 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 4.63. 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> ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDKLt1vnW2vTJpBp3VK91rFsBvpY97NljsVLdgUrlPbZ/
L51FerQQ+djQ/ivDASQjO+567nMGqfYGFA/De1EGMMEoeShza67qjNi14L1HBGgVojaNajMR/
NI2d1kDyvsgRy7D7FT5UGGUNT0dlcSD3b85zwgHeYLidgcGIoKeRi7HpVDOOTyhwUv4sq3ubrPCWARgP
eOLdVFa9clC8PTZdxSeKp4jpNjIHEyREPin2Un1luCIPWrOYyym7aRJEPopCEqBA9HvfwpbuwBI5F0uI
WZgSQLfpwW86599fBo/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>
```

`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.

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.

Do not create a superuser account with a name other than `root`

While it is technically possible to create an account with the user ID (`uid`) `0` and a name other than `root`, certain applications, scripts or third-party products may rely on the existence of a user called `root`. While such a configuration always targets individual environments, necessary adjustments could be overwritten by vendor updates, so this becomes an ongoing task rather than a one-time setting. This is especially true in very complex setups involving third-party applications, where it needs to be verified with every vendor involved whether a rename of the `root` account is supported.

As the implications for renaming the `root` account cannot be foreseen, SUSE does not support renaming the `root` account.

Usually, the idea behind renaming the `root` account is to hide it or make it unpredictable. However, `/etc/passwd` requires `644` permissions for regular users, so any user of the system can retrieve the login name for the user ID 0. For better ways to secure the `root` account, refer to the section called “Restricting `root` logins” in “Security and Hardening Guide” and the section called “Restricting SSH logins” in “Security and Hardening Guide”.

Specifying a user ID (`uid`)

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

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

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

Attribute, Values, Description

`username`

Text

```
<username>lukesw</username>
```

Required. It should be a valid user name. Check `man 8 useradd` if you are not sure.

`fullname`

Text

```
<fullname>Tux Torvalds</fullname>
```

Optional. User's full name.

`forename`

Text

```
<forname>Tux</forename>
```

Optional. User's forename.

`surname`

Text

```
<surname>Skywalker</surname>
```

Optional. User's surname.

uid

Number

```
<uid>1001</uid>
```

Optional. User ID. It should be a unique and must be a non-negative number. If not specified, AutoYaST will automatically choose a user ID. Also refer to *Specifying a user ID (`uid`)* for additional information.

gid

Number

```
<gid>100</gid>
```

Optional. Initial group ID. It must be a unique and non-negative number. Moreover it must refer to an existing group.

home

Path

```
<home>/home/luke</home>
```

Optional. Absolute path to the user's home directory. By default, `/home/username` will be used (for example, `alice`'s home directory will be `/home/alice`).

home_btrfs_subvolume

Boolean

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

Optional. Generates the home directory in a Btrfs subvolume. Disabled by default.

shell

Path

```
<shell>/usr/bin/zsh</shell>
```

Optional. `/bin/bash` is the default value. If you choose another one, make sure that it is installed (adding the corresponding package to the `software` section).

user_password

Text

```
<user_password>some-password</user_password>
```

Optional. A user's password can be written in plain text (not recommended) or in encrypted form. To create an encrypted password, use **mkpasswd**. Enter the password as written in `/etc/shadow` (second column). To enable or disable the use of encrypted passwords in the profile, see the `encrypted` parameter. If you enter an exclamation mark (!) with encrypted passwords enabled, the value is copied to the password field of `/etc/shadow`. Therefore, you get an account with a locked password that cannot log in on console.

encrypted

Boolean

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

Optional. Considered `false` if not present. Indicates if the user's password in the profile is encrypted or not. AutoYaST supports standard encryption algorithms (see **man 3 crypt)**.

password_settings

Password settings

```
<password_settings>
  <expire/>
  <max>60</max>
  <warn>7</warn>
</password_settings>
```

Optional. Some password settings can be customized: `expire` (account expiration date in format `YYYY-MM-DD`), `flag` (`/etc/shadow` flag), `inact` (number of days after password expiration that account is disabled), `max` (maximum number of days a password is valid), `min` (grace period in days until which a user can change password after it has expired) and `warn` (number of days before expiration when the password change reminder starts).

authorized_keys

List of authorized keys

```
<authorized_keys config:type="list">
  <listentry>ssh-rsa ...</listentry>
</authorized_keys>
```

A list of authorized keys to be written to `$HOME/.ssh/authorized_keys`. See example below.

4.31.2. 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` or any other appropriate file to be read for `useradd`.

Attribute, Values, Description

`group`

Text

```
<group>100</group>
```

Optional. Default initial login group.

`home`

Path

```
<home>/home</home>
```

Optional. User's home directory prefix.

`expire`

Date

```
<expire>2017-12-31</expire>
```

Optional. Default password expiration date in `YYYY-MM-DD` format.

`inactive`

Number

```
<inactive>3</inactive>
```

Optional. Number of days after which an expired account is disabled.

`shell`

Path

```
<shell>/usr/bin/fish</shell>
```

Default login shell. `/bin/bash` is the default value. If you choose another one, make sure that it is installed (adding the corresponding package to the `software` section).

`umask`

File creation mode mask

```
<umask>022</umask>
```

Set the file creation mode mask for the home directory. By default `useradd` will use `022`. Check `man 8 useradd` and `man 1 umask` for further information.

4.31.3. Groups

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

Example 4.64. Group configuration

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

Attribute, Values, Description

`groupname`

Text

```
<groupname>users</groupname>
```

Required. It should be a valid group name. Check `man 8 groupadd` if you are not sure.

`gid`

Number

```
<gid>100</gid>
```

Optional. Group ID. It must be a unique and non-negative number.

`userlist`

Users list

```
<userlist>bob,alice</userlist>
```

Optional. A list of users who belong to the group. User names must be separated by commas.

4.31.4. 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 4.65. Enabling autologin and password-less login

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

Attribute, Values, Description

`password_less_login`

Boolean

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

Optional. Enables password-less login. It only affects graphical login.

`autologin_user`

Text

```
<autologin_user>alice</autologin_user>
```

Optional. Enables autologin for the given user.

4.32. 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:

Except for `init scripts`, all scripts must be included 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` (when the installed system is first booted, when all services are running) are not executed by YaST and therefore have a special Status. See *the section called "Init scripts"* for further information.

4.32.1. Pre-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-scripts.

Pre-scripts with confirmation

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

Pre-scripts and Zypper

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

The `pre-script` elements must be placed as follows:

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

4.32.2. Postpartitioning scripts

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

The `postpartitioning-script` elements must be placed as follows:

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

4.32.3. 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).

The `chroot-scripts` elements must be placed as follows:

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

4.32.4. Post-scripts

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

The `post-script` elements must be placed as follows:

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

4.32.5. Init scripts

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

The `init-script` elements must be placed as follows:

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

Init scripts are different from the other 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.

Init script XML representation

`location`

> Define a location from where the script gets fetched. Locations can be the same as for the profile (HTTP, FTP, NFS, etc.).
>
> ```
> <location>http://10.10.0.1/myInitScript.sh</location>
> ```
>
> Either <location> or <source> must be defined.

`source`

The script itself (source code), encapsulated in a CDATA tag. If you do not want to put the whole shell script into the XML profile, use the location parameter.

```
<source>
<![CDATA[echo "Testing the init script" >/tmp/init_out.txt]]></source>
```

Either <location> or <source> must be defined.

`filename`

The file name of the script. It will be stored in a temporary directory under `/tmp`

```
<filename>mynitScript5.sh</filename>
```

Optional in case you only have a single init script. The default name (`init-scripts`) is used in this case. If having specified more than one init script, you must set a unique name for each script.

`rerun`

Normally, a script is only run once, even if you use `ayast_setup` to run an XML file multiple times. Change this default behavior by setting this boolean to `true`.

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

Optional. Default is `false` (scripts only run once).

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

4.32.6. 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 *the section called “Init scripts”* for further information about them.

Deprecated elements

`debug` is a deprecated element and can be removed in future releases. To adapt, use an interpreter-specific debugging parameter in `interpreter`. For example, instead of <interpreter>shell</interpreter> use <interpreter>/bin/sh -x</interpreter> for the same result as having enabled the `debug` flag.

Script XML representation

`location`

Define a location from where the script gets fetched. Locations can be the same as for the control file (HTTP, FTP, NFS, etc.), for example:

```
<location>http://10.10.0.1/myPreScript.sh</location>
```

The location can also be defined as a relative URL, where the path is relative to the directory with the control file. If the relative URL is used, the `location` attribute appears as follows:

```
<location>relurl://script.sh</location>
```

Alternatively, you can use the `repo` URI scheme. The script location is relative to the installation source, and the definition appears as follows:

```
<location>repo:/script.sh</location>
```

Either `location` or `source` must be defined.

`source`

The script itself (source code), encapsulated in a CDATA tag. If you do not want to put the whole shell script into the XML control file, refer to the location parameter.

```
<source>
<![CDATA[
echo "Testing the pre script" > /tmp/pre-script_out.txt
]]>
</source>
```

Either `location` or `source` must be defined.

`interpreter`

Specify the interpreter that must be used for the script. Any interpreter available in the given environment can be specified. It is possible to provide a full path to the interpreter, including parameters. There are also deprecated keywords interpreter "shell", "perl" and "python" that are supported by the `debug` flag.

```
<interpreter>/bin/bash -x</interpreter>
```

Optional; default is `shell`.

`file name`

The file name of the script. It will be stored in a temporary directory under `/tmp`.

```
<filename>myPreScript5.sh</filename>
```

Optional; default is the type of the script (pre-scripts in this case). If you have more than one script, you should define different names for each script. If `filename` is not defined and `location` is defined, the file name from the location path will be used.

`feedback`

If this boolean is `true`, output and error messages of the script (STDOUT and STDERR) will be shown in a pop-up. The user needs to confirm them via the OK button.

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

Optional; default is `false`.

`feedback_type`

This can be `message`, `warning` or `error`. Set the timeout for these pop-ups in the <report> section.

```
<feedback_type>warning</feedback_type>
```

Optional; if missing, an always-blocking pop-up is used.

`debug`

If this is `true`, every single line of a shell script is logged. Perl scripts are run with warnings turned on. This only works for the deprecated keyword `interpreter`. For other languages, give the path to the interpreter as a parameter in the `interpreter` value, for example "<interpreter>ruby -w</interpreter>".

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

Optional; default is `true`.

`notification`

This text will be shown in a pop-up for the time the script is running in the background.

```
<notification>Please wait while script is running...</notification>
```

Optional; if not configured, no notification pop-up will be shown.

`param-list`

It is possible to specify parameters given to the script being called. You may have more than one `param` entry. They are concatenated by a single space character on the script command line. If any shell quoting should be necessary (for example to protect embedded spaces) you need to include this.

```
<param-list config:type="list">
  <param>par1</param>
  <param>par2 par3</param>
  <param>"par4.1 par4.2"</param>
</param-list>
```

Optional; if not configured, no parameters get passed to script.

rerun

A script is only run once. Even if you use `ayast_setup` to run an XML file multiple times, the script is only run once. Change this default behavior by setting this boolean to `true`.

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

Optional; default is `false`, meaning that scripts only run once.

chrooted

During installation, the new system is mounted at `/mnt`. If this parameter is set to `false`, AutoYaST does not run **`chroot`** and does not install the boot loader at this stage. If the parameter is set to `true`, AutoYaST performs a **`chroot`** into `/mnt` and installs the boot loader. The result is that to change anything in the newly-installed system, you no longer need to use the `/mnt` prefix.

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

Optional; default is `false`. This option is only available for chroot environment scripts.

4.32.7. Script example

Example 4.66. Script configuration

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://
www.suse.com/1.0/configns">
<scripts>
  <chroot-scripts config:type="list">
    <script>
      <chrooted config:type="boolean">true</chrooted>
      <filename>chroot-post.sh</filename>
      <interpreter>shell</interpreter>
      <source><![CDATA[
echo "Testing chroot (chrooted) scripts"
ls
]]>
      </source>
    </script>
    <script>
      <filename>chroot-pre.sh</filename>
        <interpreter>/bin/bash -x</interpreter>
        <source><![CDATA[
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[
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[
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[
echo "Running pre-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 scripts, containing a combination of both the standard output and the standard error output.

If the script ends with a non-zero exit code, then a warning will be shown with the content of the logs, unless the `feedback` option was provided.

4.33. 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 4.67. 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.

4.34. 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 4.68. 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 4.69. 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>
```

4.35. 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>
```

Ask the user for values: XML representation

`question`

The question you want to ask the user.

```
<question>Enter the LDAP server</question>
```

The default value is the path to the element (the path often looks strange, so we recommend entering a question).

default

Set a preselection for the user. A text entry will be filled out with this value. A check box will be true or false and a selection will have the given value preselected.

```
<default>dc=suse,dc=de</default>
```

Optional.

help

An optional help text that is shown on the left side of the question.

```
<help>Enter the LDAP server address.</help>
```

Optional.

title

An optional title that is shown above the questions.

```
<title>LDAP server</title>
```

Optional.

type

The type of the element you want to change. Possible values are `symbol`, `boolean`, `string` and `integer`. The file system in the partition section is a symbol, while the `encrypted` element in the user configuration is a boolean. You can see the type of that element if you look in your control file at the `config:type="...."` attribute. You can also use `static_text` as type. A `static_text` is a text that does not require any user input and can show information not included in the help text.

```
<type>symbol</type>
```

Optional. The default is `string`. If type is `symbol`, you must provide the selection element too (see below).

password

If this boolean is set to `true`, a password dialog pops up instead of a simple text entry. Setting this to `true` only makes sense if `type` is string.

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

Optional. The default is `false`.

pathlist

A list of `path` elements. A path is a comma-separated list of elements that describes the path to the element you want to change. For example, the network configuration element can be found in the control file in the `<networking>` section. So, to change that value, you need to set the path to `networking`.

```
<pathlist config:type="list">
  <path>networking,dns,hostname</path>
  <path>...</path>
</pathlist>
```

To change the password of the first user in the control file, you need to set the path to `users,0,user_password`. `0` indicates the first item in the configuration section. For example, in the <users config:type="list"> list of users mentioned below, it relates to `root`. `1` would be the second item, and so on.

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

To set a password for `root` if the <user> section is similar to the one above, use the <pathlist> as follows:

```
<pathlist config:type="list">
          <path>users,0,user_password</path>
        </pathlist>
```

This information is optional, but you should at least provide `path` or `file`.

file

You can store the answer to a question in a file, to use it in one of your scripts later. If you ask during `stage=initial` and you want to use the answer in stage 2, then you need to copy the answer-file in a chroot script that is running as `chrooted=false`. Use the command: **`cp /tmp/my_answer /mnt/tmp/`**. The reason is that `/tmp` in stage 1 is in the RAM disk and will be lost after the reboot, but the installed system is already mounted at `/mnt/`.

```
<file>/tmp/answer_hostname</file>
```

This information is optional, but you should at least provide `path` or `file`.

stage

Stage configures the installation stage in which the question pops up. You can set this value to `cont` or `initial`. `initial` means the pop-up comes up very early in the installation, shortly after the pre-script has run. `cont` means, that the dialog with the question comes after the first reboot when the system boots for the very first time. Questions you answer during the `initial` stage will write their answer into the control file on the hard disk. You should know that if you enter clear text passwords during `initial`. Of course it does not make sense to ask for the file system to use during the `cont` phase. The hard disk is already partitioned at that stage and the question will have no effect.

```
<stage>cont</stage>
```

Optional. The default is `initial`.

selection

The selection element contains a list of `entry` elements. Each entry represents a possible option for the user to choose. The user cannot enter a value in a text box, but they can choose from a list of values.

```
<selection config:type="list">
  <entry>
    <value>
        btrfs
    </value>
    <label>
        Btrfs File System
    </label>
  </entry>
  <entry>
    <value>
        ext3
    </value>
    <label>
        Extended3 File System
    </label>
  </entry>
</selection>
```

Optional for `type=string`, not possible for `type=boolean` and mandatory for `type=symbol`.

dialog

You can ask more than one question per dialog. To do so, specify the dialog-id with an integer. All questions with the same dialog-id belong to the same dialog. The dialogs are sorted by the id too.

```
<dialog config:type="integer">3</dialog>
```

Optional.

element

You can have more than one question per dialog. To make that possible you need to specify the `element-id` with an integer. The questions in a dialog are sorted by ID.

```
<element config:type="integer">1</element>
```

Optional (see dialog).

width

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.

```
<width config:type="integer">50</width>
```

Optional.

height

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.

```
<height config:type="integer">15</height>
```

Optional.

frametitle

You can have more than one question per dialog. Each question on a dialog has a frame that can have a frame title, a small caption for each question. You can put multiple elements into one frame. They need to have the same frame title.

```
<frametitle>User data</frametitle>
```

Optional; default is no frame title.

script

You can run scripts after a question has been answered. (See *the section called “Default value scripts”* for detailed instructions about scripts.)

```
<script>...</script>
```

Optional; default is no script.

ok_label

You can change the label on the *Ok* button. The last element that specifies the label for a dialog wins.

```
<ok_label>Finish</ok_label>
```

Optional.

`back_label`

You can change the label on the *Back* button. The last element that specifies the label for a dialog wins.

```
<back_label>change values</back_label>
```

Optional.

`timeout`

You can specify an integer here that is used as timeout in seconds. If the user does not answer the question before the timeout, the default value is taken as answer. When the user touches or changes any widget in the dialog, the timeout is turned off and the dialog needs to be confirmed via *Ok*.

```
<timeout config:type="integer">30</timeout>
```

Optional; a missing value is interpreted as `0`, which means that there is no timeout.

`default_value_script`

You can run scripts to set the default value for a question (see *the section called "Default value scripts"* for detailed instructions about default value scripts). This feature is useful if you can `calculate` a default value, especially in combination with the `timeout` option.

```
<default_value_script>...</default_value_script>
```

Optional; default is no script.

4.35.1. 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 scripts are defined by placing the elements described in *the section called "Script XML representation"* within the following XML structure:

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

Whatever you **`echo`** to STDOUT will be used as default value for the ask-dialog. If your script has an exit code other than 0, the normal `default` element is used. Take care you use **`echo -n`** to suppress the `\n` and that you echo reasonable values and not “okay” for a boolean (use “true” instead).

4.35.2. 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>
```

In addition to the elements listed in *the section called “Script XML representation”*, scripts in `<ask>` elements support these options:

Scripts: XML representation

`filename`

The file name of the script.

```
<filename>my_ask_script.sh</filename>
```

The default is ask_script.sh

`environment`

A boolean that passes the value of the answer to the question as an environment variable to the script. The variable is named `VAL`.

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

Optional. Default is `false`.

`feedback`

A boolean that turns on feedback for the script execution. STDOUT will be displayed in a pop-up window that must be confirmed after the script execution.

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

Optional, default is `false`.

`rerun_on_error`

Keep the dialog open until the script has an exit code of 0 (zero). You can use this feature to validate the user's input. The script should print a meaningful error message and return a code different from zero. Bear in mind that you should also set the `feedback` option to `true` so the user can read the error message from the script. Optional, default is `false`.

Your script can create a file `/tmp/next_dialog` containing the ID of the following dialog to display. A value of -1 terminates the sequence.

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>tion>
  </ask-list>
</general>
```

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

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

4.36. Kernel dumps

Availability

This feature is not available on AArch64, or on systems with less than 1 GB of RAM.

With Kdump the system can create crash dump 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 4.70. 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_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>
```

Kdump is enabled by default. The following configuration shows how to disable it.

Example 4.71. Disabled Kdump configuration

```
<kdump>
  <add_crash_kernel config:type="boolean">false</add_crash_kernel>
</kdump>
```

4.36.1. 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 the section called “Manual Kdump configuration” in “System Analysis and Tuning Guide” 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 4.72. 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 list shows the settings necessary to reserve memory:

Kdump memory reservation settings:XML representation

`add_crash_kernel`

Set to `true` if memory should be reserved and Kdump enabled.

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

required

`crash_kernel`

Use the syntax of the crashkernel command line as discussed above.

```
<crash_kernel>256M:64M</crash_kernel>
```

A list of values is also supported.

```
<crash_kernel config:type="list">
  <listentry>72M,low</listentry>
  <listentry>256M,high</listentry>
</crash_kernel>
```

required

4.36.2. Dump saving

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

4.36.2.1. Target

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

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

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

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

4.36.2.2. 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.

4.36.2.3. Summary

Dump target settings: XML representation

`KDUMP_SAVEDIR`

A URL that specifies the target to which the dump and related files will be saved.

```
<KDUMP_SAVEDIR>file:///var/crash/</KDUMP_SAVEDIR>
```

required

`KDUMP_FREE_DISK_SIZE`

Disk space in megabytes that must remain free after saving the dump. If not enough space is available, the dump will not be saved.

```
<KDUMP_FREE_DISK_SIZE>64</KDUMP_FREE_DISK_SIZE>
```

optional

`KDUMP_KEEP_OLD_DUMPS`

The number of dumps that are kept (not deleted) if `KDUMP_SAVEDIR` points to a local directory. Specify 0 if you do not want any dumps to be automatically deleted, specify -1 if all dumps except the current one should be deleted.

```
<KDUMP_KEEP_OLD_DUMPS>4</KDUMP_KEEP_OLD_DUMPS>
```

optional

4.36.3. 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.

E-mail notification settings: XML representation

KDUMP_NOTIFICATION_TO

Exactly one e-mail address to which the e-mail should be sent. Additional recipients can be specified in KDUMP_NOTIFICATION_CC.

```
<KDUMP_NOTIFICATION_TO
>tux@example.com</KDUMP_NOTIFICATION_TO>
```

optional (notification disabled if empty)

KDUMP_NOTIFICATION_CC

Zero, one or more recipients that are in the cc line of the notification e-mail.

```
<KDUMP_NOTIFICATION_CC
>wilber@example.com suzanne@example.com</KDUMP_NOTIFICATION_CC>
```

optional

KDUMP_SMTP_SERVER

Host name of the SMTP server used for mail delivery. SMTP authentication is supported (see KDUMP_SMTP_USER and KDUMP_SMTP_PASSWORD) but TLS/SSL are not.

```
<KDUMP_SMTP_SERVER>email.suse.de</KDUMP_SMTP_SERVER>
```

optional (notification disabled if empty)

KDUMP_SMTP_USER

User name used together with KDUMP_SMTP_PASSWORD for SMTP authentication.

```
<KDUMP_SMTP_USER>bwalle</KDUMP_SMTP_USER>
```

optional

KDUMP_SMTP_PASSWORD

Password used together with KDUMP_SMTP_USER for SMTP authentication.

```
<KDUMP_SMTP_PASSWORD>geheim</KDUMP_SMTP_PASSWORD>
```

optional

4.36.4. Kdump kernel settings

As already mentioned, a special kernel is booted to save the dump. If you do not want to use the auto-detection mechanism to find out which kernel is used (see the kdump(5) manual page that describes the algorithm which is used to find the kernel), you can specify the version of a custom

kernel in KDUMP_KERNELVER. If you set it to `foo`, then the kernel located in `/boot/vmlinuz-foo` or `/boot/vmlinux-foo` (in that order on platforms that have a `vmlinuz` file) will be used.

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

Kernel settings: XML representation

KDUMP_KERNELVER

Version string for the kernel used for Kdump. Leave it empty to use the auto-detection mechanism (strongly recommended).

```
<KDUMP_KERNELVER>6.4.0-default</KDUMP_KERNELVER>
```

optional (auto-detection if empty)

KDUMP_COMMANDLINE_APPEND

Additional command line parameters for the Kdump kernel.

```
<KDUMP_COMMANDLINE_APPEND>console=ttyS0,57600</KDUMP_COMMANDLINE_APPEND>
```

optional

KDUMP_Command Line

Overwrite the automatically generated Kdump command line. Use with care. Usually, KDUMP_COMMANDLINE_APPEND should suffice.

```
<KDUMP_COMMANDLINE_APPEND>root=/dev/sda5 nr_cpus=1 irqpoll</
KDUMP_COMMANDLINE>
```

optional

4.36.5. Expert settings

Expert settings: XML representations

KDUMP_IMMEDIATE_REBOOT

`true` if the system should be rebooted automatically after the dump has been saved, `false` otherwise. The default is to reboot the system automatically.

```
<KDUMP_IMMEDIATE_REBOOT>true</KDUMP_IMMEDIATE_REBOOT>
```

optional

KDUMP_VERBOSE

Bitmask that specifies how verbose the Kdump process should be. Read kdump(5) for details.

```
<KDUMP_VERBOSE>3</KDUMP_VERBOSE>
```

optional

KEXEC_OPTIONS

Additional options that are passed to kexec when loading the Kdump kernel. Normally empty.

```
<KEXEC_OPTIONS>--noio</KEXEC_OPTIONS>
```

optional

4.37. DNS server

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

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

Example 4.73. Basic DNS server settings

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

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

List	Description
`logging`	Options of the DNS server logging.
`options`	Bind options like the files and directories to use, the list of forwarders and other configuration settings.

List	Description
zones	List of DNS zones known by the server, including all the settings, records and SOA records.

Example 4.74. 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>
```

4.38. DHCP server

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

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

Example 4.75. Example dhcp-server section

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

  <settings config:type="list">
    <settings_entry>
      <children config:type="list"/>
      <directives config:type="list">
        <listentry>
          <key>fixed-address</key>
          <type>directive</type>
          <value>192.168.0.10</value>
        </listentry>
        <listentry>
          <key>hardware</key>
          <type>directive</type>
          <value>ethernet d4:00:00:bf:00:00</value>
        </listentry>
      </directives>
      <id>static10</id>
      <options config:type="list"/>
      <parent_id>192.168.0.0 netmask 255.255.255.0</parent_id>
      <parent_type>subnet</parent_type>
      <type>host</type>
    </settings_entry>
    <settings_entry>
      <children config:type="list">
        <child>
          <id>static10</id>
          <type>host</type>
        </child>
      </children>
      <directives config:type="list">
        <listentry>
          <key>range</key>
          <type>directive</type>
          <value>dynamic-bootp 192.168.0.100 192.168.0.150</value>
        </listentry>
        <listentry>
          <key>default-lease-time</key>
          <type>directive</type>
          <value>14400</value>
        </listentry>
        <listentry>
          <key>max-lease-time</key>
          <type>directive</type>
          <value>86400</value>
        </listentry>
      </directives>
      <id>192.168.0.0 netmask 255.255.255.0</id>
      <options config:type="list"/>
      <parent_id/>
      <parent_type/>
      <type>subnet</type>
    </settings_entry>
    <settings_entry>
      <children config:type="list">
        <child>
          <id>192.168.0.0 netmask 255.255.255.0</id>
          <type>subnet</type>
        </child>
      </children>
```

```
      <directives config:type="list">
        <listentry>
          <key>ddns-update-style</key>
          <type>directive</type>
          <value>none</value>
        </listentry>
        <listentry>
          <key>default-lease-time</key>
          <type>directive</type>
          <value>14400</value>
        </listentry>
      </directives>
      <id/>
      <options config:type="list"/>
      <parent_id/>
      <parent_type/>
      <type/>
    </settings_entry>
  </settings>
</dhcp-server>
```

4.39. Firewall configuration

SuSEfirewall2 has been replaced by `firewalld` starting with SUSE Linux Enterprise Server 15 GA. Profiles using SuSEfirewall2 properties will be translated to `firewalld` profiles. However, not all profile properties can be converted. For details about `firewalld`, refer to the section called “ `firewalld` ” in “Security and Hardening Guide”.

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.

4.39.1. 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
`start_firewall`	Boolean	Whether `firewalld` should be started right after applying the configuration.
`enable_firewall`	Boolean	Whether `firewalld` should be started on every system start-up.
`default_zone`	Zone name	The default zone is used for everything that is not explicitly assigned.

Attribute	Value	Description
`log_denied_packets`	Type of dropped packets to be logged	Enable logging of dropped packets for the type selected. Values: `off`, `unicast`, `multicast`, `broadcast`, `all`.
`name`	Identifier of zone	Used to identify a zone. If the zone is not known yet, a new zone will be created.
`short`	Short summary of zone	Briefly summarizes the purpose of the zone. Ignored for already existing zones. If not specified, the name is used.
`description`	Description of zone	Describes the purpose of the zone. Ignored for already existing zones. If not specified, the name is used.
`target`	Default action	Defines the default action in the zone if no rule matches. Possible values are `ACCEPT`, `%%REJECT%%`, `DROP` and `default`. If not specified, `default` is used. For details about values, see https://firewalld.org/documentation/zone/options.html.

4.39.2. 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
`interfaces`	List of interface names	List of interface names assigned to this zone. Interfaces or sources can only be part of one zone.
`services`	List of services	List of services accessible in this zone.
`ports`	List of ports	List of single ports or ranges to be opened in the assigned zone.
`protocols`	List of protocols	List of protocols to be opened or be accessible in the assigned zone.
`masquerade`	Enable masquerade	It will enable or disable network address translation *(NAT)* in the assigned zone.

4.39.3. Installation stages when the `firewalld` profile is applied

Starting with SUSE Linux Enterprise Server 15 SP3, the `firewalld` profile is usually applied at the end of the first stage of the installation. (To learn about the installation stages, see *the section called "Overview and concept"*.) However, there are circumstances where the profile is applied in the second stage. The following list specifies the conditions under which the `firewalld` profile is applied in the first or second stage:

- You are running AutoYaST with a `firewalld` section, and not installing SUSE Linux Enterprise Server over SSH or VNC. The firewall is configured in the first stage.
- You are running AutoYaST with a `firewalld` section, installing SUSE Linux Enterprise Server over SSH or VNC, and no second stage is required. The firewall is configured in the first stage.
- You are running AutoYaST with a `firewalld` section, installing SUSE Linux Enterprise Server over SSH or VNC, and the second stage is required. The firewall is configured in the second stage.
- You are running AutoYaST without a `firewalld` section. The firewall is configured in the first stage according to the default product proposals.
- You are running AutoYaST with or without a firewall section, together with custom script which requires network access. The firewall is configured in the first stage either according to the profile or the product proposals, and the firewall configuration must be adapted so that the custom script has network access as needed.

4.39.4. A full example

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

Example 4.76. Example firewall section

```
<firewall>
  <enable_firewall config:type="boolean">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>
```

4.40. 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.

4.40.1. 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 4.77, "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 4.77. 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>
```

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

With release 1.6 the CUPS configuration file has been split into two files: `cupsd.conf` and `cups-files.conf`. As of SUSE Linux Enterprise Server15 SP7, AutoYaST only supports modifying `cupsd.conf` since the default settings in `cups-files.conf` are sufficient for usual printing setups.

4.40.2. Sound devices

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

Example 4.78. 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>
```

4.41. 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 4.79. Importing SSH keys and configuration from /dev/sda2

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

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

4.42. 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.

Only Salt is supported

Although Puppet is mentioned in this document, only Salt is supported. Nevertheless, feel free to report any problems 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.

4.42.1. 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 4.80. 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 4.81. 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
`type`	String	Configuration management name. Currently only `salt` is supported.
`master`	String	Host name or IP address of the configuration management server.
`auth_attempts`	Integer	Maximum attempts to connect to the server. The default is three attempts.
`auth_time_out`	Integer	Time (in seconds) between attempts to connect to the server. The default is 15 seconds.
`keys_url`	URL of used key	Path to an HTTP server, hard disk, flash disk or similar with the files `default.key` and `default.pub`. This key must be known to the configuration management master.
`enable_services`	True/ False	Enables the configuration management services on the client side after the installation. The default is `true`.

4.42.2. 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 disk, 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 4.82. Stand-alone 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
`type`	String	Configuration management name. Currently only `salt` is supported.
`states_url`	URL	Location of the Salt states TAR archive. It may include formulas and pillars. Files must be located in a `salt` directory.
`pillar_url`	URL	Location of the TAR archive that contains the pillars.
`modules_url`	URL	Location of Puppet modules.

4.42.3. SUSE Multi-Linux Manager Salt formulas support

AutoYaST offers support for SUSE Multi-Linux 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.

Part III. Managing mass installations with dynamic profiles

Chapter 5. Supported approaches to dynamic profiles

When dealing with the installation of multiple systems, it might be useful to use a single profile (or a reduced set of them) that adapts automatically to each system. In this regard, AutoYaST offers three different mechanisms to modify the profile at installation time.

Rules and classes

Rules and classes offer the possibility to configure a system by merging multiple control files during installation. You can read more about this feature in the *Chapter 6, Rules and classes* section.

ERB templates

AutoYaST supports Embedded Ruby (ERB) templates syntax to modify the profile's content during installation. The *Chapter 7, ERB templates* section describes how to use them.

Pre-installation scripts

You can use a pre-installation script to modify or even create a brand new profile during installation. *the section called “Pre-scripts”* describes how to benefit from them.

Asking the user during installation

As an alternative, AutoYaST can ask the user for values to use in the profile at runtime. The installation is not fully unattended in that case, but it can be rather useful to set user names, passwords, IP addresses and so on. You can find more information about this feature in the *the section called “Ask the user for values during installation”* section.

Chapter 6. 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.

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.

6.1. 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:

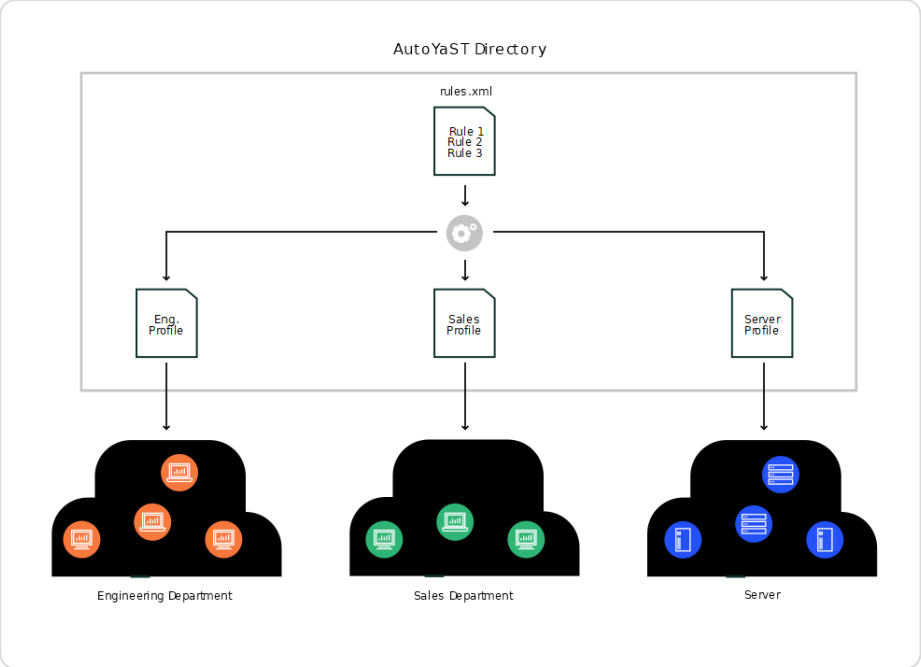
```
autoyast=http://10.10.0.1/profile/myprofile.xml
autoyast=http://10.10.0.1/profile/rules/rules.xml
```

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 6.1. 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.

6.1.1. Rules file explained

Example 6.1. Simple rules file

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

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

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

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

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

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

Example 6.2. Simple rules file

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

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

6.1.2. 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`.

6.1.3. 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`.

6.1.4. 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>
```

6.1.5. 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.

6.1.6. 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 **`/sbin/yast2 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 6.1. System attributes

Attribute	Values	Description
`hostaddress`	IP address of the host	This attribute must always match exactly.
`hostname`	The name of the host	This attribute must always match exactly.
`domain`	Domain name of host	This attribute must always match exactly.

Attribute	Values	Description
`installed_product`	The name of the product to be installed.	This attribute must always match exactly.
`installed_product_version`	The version of the product to be installed.	This attribute must always match exactly.
`network`	network address of host	This attribute must always match exactly.
`mac`	MAC address of host	This attribute must always match exactly (the MAC addresses should have the form `0080c8f6484c`).
`linux`	Number of installed Linux partitions on the system	This attribute can be 0 or more.
`others`	Number of installed non-Linux partitions on the system	This attribute can be 0 or more.
`xserver`	X Server needed for graphic adapter	This attribute must always match exactly.
`memsize`	Memory available on host in megabytes	All match types are available.
`totaldisk`	Total disk space available on host in megabytes	All match types are available.
`hostid`	Hex representation of the IP address	Exact match required
`arch`	Architecture of host	Exact match required
`karch`	Kernel Architecture of host (for example SMP kernel, Xen kernel)	Exact match required

Attribute	Values	Description
`disksize`	Drive device and size in megabytes	All match types are available.
`product`	The hardware product name as specified in SMBIOS	Exact match required
`product_vendor`	The hardware vendor as specified in SMBIOS	Exact match required
`board`	The system board name as specified in SMBIOS	Exact match required
`board_vendor`	The system board vendor as specified in SMBIOS	Exact match required
`custom1-5`	Custom rules using shell scripts	All match types are available.

6.1.7. Rules with dialogs

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

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

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

Attribute, Values, Description

`dialog_nr`

All rules with the same `dialog_nr` are presented in the same pop-up dialog. The same `dialog_nr` can appear in multiple rules.

```
<dialog_nr config:type="integer">3</dialog_nr>
```

This element is optional and the default for a missing dialog_nr is always `0`. To use one pop-up for all rules, you do not need to specify the `dialog_nr`.

`element`

Specify a unique ID. Even if you have more than one dialog, you must not use the same id twice. Using id `1` on dialog 1 and id `1` on dialog 2 is not supported. (This behavior is contrary to the `ask` dialog, where you can have the same ID for multiple dialogs.)

```
<element config:type="integer">3</element>
```

Optional. If omitted, AutoYaST adds its own IDs internally. Then you cannot specify conflicting rules (see below).

`title`

Caption of the pop-up dialog

```
<title>Desktop Selection</title>
```

Optional

`question`

Question shown in the pop-up behind the check box.

```
<question>GNOME Desktop</question>
```

Optional. If you do not configure a text here, the name of the XML file that is triggered by this rule will be shown instead.

`timeout`

Timeout in seconds after which the dialog will automatically “press” the okay button. Useful for a non-blocking installation in combination with rules dialogs.

```
<timeout config:type="integer">30</timeout>
```

Optional. A missing timeout will stop the installation process until the dialog is confirmed by the user.

`conflicts`

A list of element ids (rules) that conflict with this rule. If this rule matches or is selected by the user, all conflicting rules are deselected and disabled in the pop-up. Take care that you do not create deadlocks.

```
<conflicts config:type="list">
  <element config:type="integer">1</element>
  <element config:type="integer">5</element>
  ...
</conflicts>
```

Optional

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>
```

6.2. 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>
```

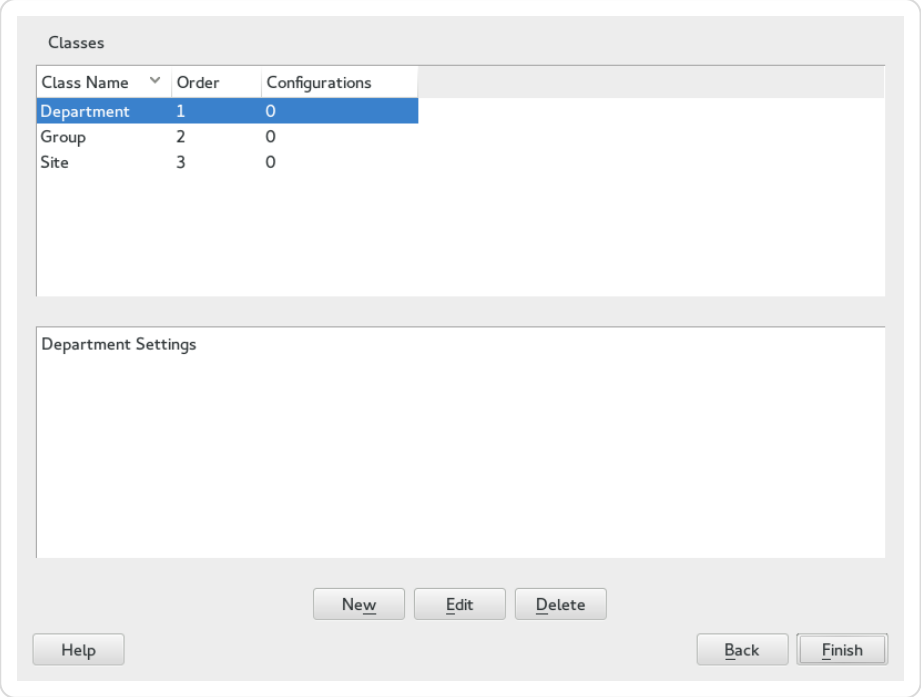
In the example above, the file `Software.xml` must be placed in the subdirectory `classes/TrainingRoom/`. It will be fetched from the same location as the AutoYaST control file and rules.

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 6.2. 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.

6.3. 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.

6.4. Merging of rules and classes

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

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

```
<partitioning config:type="list">
  <drive>
    <partitions config:type="list">
      <partition>
        <filesystem config:type="symbol">swap</filesystem>
        <format config:type="boolean">true</format>
        <mount>swap</mount>
        <partition_id config:type="integer">130</partition_id>
        <size>2000mb</size>
      </partition>
      <partition>
        <filesystem config:type="symbol">xfs</filesystem>
        <partition_type>primary</partition_type>
        <size>4Gb</size>
        <mount>/data</mount>
      </partition>
    </partitions>
  </drive>
</partitioning>
```

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

You might expect the control file to contain 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>
```

Chapter 7. ERB templates

ERB templates are for embedding Ruby code within an AutoYaST profile to modify the profile during the installation. With this approach, you can inspect the system and adjust the profile by setting values, adding or skipping sections, and so on.

To activate the ERB processing, the profile must have the extension `.erb` (for example, `autoyast.xml.erb`). Hence, it is not possible to combine rules/classes and ERB templates.

7.1. What is ERB?

ERB stands for *Embedded Ruby*. ERB uses the power of the Ruby programming language to generate different kinds of content. With ERB, you can include some Ruby code in your profiles to adapt them at runtime, depending on the installation system.

When using ERB, the Ruby code is enclosed between <% and %> signs. Use an equals sign, =, to include command output in the resulting profile.

Example 7.1. Including a file using ERB

```
<bootloader>
  <% require "open-uri" %>
  <%= URI.open("http://192.168.1.1/profiles/bootloader-common.xml").read %>
</bootloader> <!-- this line gets replaced with the content of bootloader-
common.xml -->
```

You can use Ruby facilities to run arbitrary commands. If you want to get the output of a command, then enclose it between backticks. If you want to know whether a command was successful or not, run the command with the `system` function.

Example 7.2. Running commands with Ruby

```
<% files = `ls` %> <!-- files contains the output of the command (for instance
"file1\nfile2\nfile3") -->
<% success = system("dmidecode | grep some-model") %> <!-- success contains true
or false -->
```

Also, you can use more advanced Ruby code structures such as conditions and loops.

Example 7.3. Using Ruby structures

```
<% ip_forward = File.read("/proc/sys/net/ipv4/ip_forward").strip %>
<% if ip_forward == "1" %>
  <!-- something -->
<% end %>

<% files = `ls /tmp/config/*.xml` %>
<% files.split.each do |file| %>
  <%= file.read %>
<% end %>
```

AutoYaST offers a small set of *helper functions* to retrieve information from the underlying system, like `disks` or `network_cards`. You can check the list of helpers and their values in the *the section called “Template helpers”* section.

7.2. Template helpers

Template helpers are sets of Ruby methods that can be used in the profiles to retrieve information about the installation system.

7.2.1. `boot_efi?`

`boot_efi?` is a boolean helper that returns whether the system is booted using EFI. In the example below, the profile configures the bootloader according to the current boot mode.

Example 7.4. Configuring the boot loader

```
<% if env.boot_efi? %>
  <loader_type>grub2-efi</loader_type>
<% else %>
  <loader_type>grub2</loader_type>
<% end %>
```

7.2.2. `disks`

The `disks` helper returns a list of the detected disks. Each element of the list contains some basic information like the device name or the size.

Key	Type	Value
`:device`	String	Device kernel name (for example, `sda`).
`:model`	String	Disk model
`:serial`	String	Serial number
`:size`	Integer	Disk size (is a count of disk sectors)
`:udev_names`	Array<String>	List of disk udev names. You can use any of them to refer to the device.
`:vendor`	String	Disk vendor's name

The profile in the example below installs the system on the largest disk. It sorts the list of existing disks by size and takes the last one. Then it uses the `:device` key as value for the `device` element.

Example 7.5. Using the largest disk

```
<partitioning t="list">
  <drive>
    <% disk = disks.sort_by { |d| d[:size] }.last %> <!-- find the largest disk
-->
    <device><%= disk[:device] %></device> <!-- print the disk device name -->
    <initialize t="boolean">true</initialize>
    <use>all</use>
  </drive>
</partitioning>
```

7.2.3. `network_cards`

The `network_cards` helper returns a list of network cards, including their names, status information (for example, if they are connected or not).

Key	Type	Value
`:device`	String	Device name (for example, `eth0` or `enp3s0`)
`:mac`	String	MAC address
`:active`	Boolean	Whether the device is active or not
`:link`	Boolean	Whether the device is connected or not
`:vendor`	String	Disk vendor's name

The following example finds the first network card that is connected to the network and configures it to use DHCP.

Example 7.6. Configure the connected network cards

```
<interfaces t="list">
  <% with_link = network_cards.sort_by { |n| n[:name] }.find { |n| n[:link] } %>
  <% if with_link %>
    <interface>
      <device><%= with_link[:device] %></device>
      <startmode>auto</startmode>
      <bootproto>dhcp</bootproto>
      </interface>
  <% end %>
</interfaces>
```

7.2.4. `os_release`

The `os_release` helper returns the operating system information, which is included in the `/etc/os-release` file.

Key	Type	Value
`:id`	String	Distribution ID (for example, `sles`, `opensuse-tumbleweed`)
`:name`	String	Distribution name (for example, `SLES` or `openSUSE Tumbleweed`)
`:version`	String	Distribution version (for example, `15.2`)

You might use this information to decide which product to install, using pretty much the same profile for all of them (SLE or openSUSE distributions).

Example 7.7. Reusing the same profile for different distributions

```
<products t="list">
  <% if os_release[:id] == 'sle' %>
  <product>SLES</product>
  <% else %>
  <product>openSUSE</product>
  <% end %>
</products>
```

7.2.5. `hardware`

The `hardware` helper provides additional hardware information. It returns all the information from the `hwinfo` command. You can use this helper as a fallback for those cases in which the information available through the described helpers is not enough. In the next example, the `hardware` helper is used to filter USB devices. Check *the section called "Running ERB helpers"* to learn how to inspect all the information provided by the `hardware` helper.

Example 7.8. Filtering USB devices

```
<% usb_disks = hardware["disk"].select { |d| d["driver"] != "usb-storage" } %>
```

7.3. Running ERB helpers

You can use the Ruby console to run AutoYaST ERB helpers and find out what they offer. All ERB helpers are accessed through an instance of the `Y2Autoinstallation::Y2ERB::TemplateEnvironment` class. Start the Ruby interactive interpreter by running, as root: `irb -ryast -rautoinstall/y2erb`.

Example 7.9. Running helpers

```
irb > env = Y2Autoinstallation::Y2ERB::TemplateEnvironment.new  # the env
variable gives access to the helpers

irb > env.disks
=>
[{:vendor=>"WDC", :device=>"sda", ...},
 {:vendor=>"TOSHIBA", :device=>"sdb", ...},
...]

irb > env.hardware.keys
=>
["architecture",
 "bios",
 "bios_video",
 ...]

irb > env.hardware["architecture"]
=>
"x86_64"
```

7.4. Rendering ERB profiles

The AutoYaST command line provides a `check-profile` command that can be used to generate a profile from an ERB file. This command asks AutoYaST to parse, run the ERB code, and generate the resulting profile. You can inspect the rendered profile to check that everything worked as expected. See the command help for all the options it supports: `autoyast check-profile --help`. In the following example, `check-profile` asks AutoYaST to download and parse the profile, interpret the ERB code and run the pre-scripts. The result will be dumped to the `result.xml` file.

Example 7.10. Rendering profile

```
>sudo yast2 autoyast check-profile filename=http://192.168.1.100/autoinst.erb
output=result.xml run-scripts=true run-erb=true
```

`check-profile` permissions

In most cases, `check-profile` requires root permissions, so be careful when running pre-installation scripts and ERB profiles as root. Use only profiles that you trust.

7.5. Debugging ERB profiles

For those cases in which you want to stop the ERB evaluation and check what is happening, YaST offers integration with the `byebug` debugger. Install the `rubygem(byebug)` package and set the `Y2DEBUGGER` environment variable to 1.

Example 7.11. Preparing the debug environment

```
>sudo zypper --non-interactive in "rubygem(byebug)"
>sudo Y2DEBUGGER=1 yast2 autoyast check-profile ...
```

Adding breakpoints is as easy as adding <% byebug %> where you want to stop. For more information about `byebug`, see https://github.com/deivid-rodriguez/byebug.

Example 7.12. Adding a breakpoint

```
<% byebug %>
<% if system("dmidecode | grep some-model") %>
  <!-- do something -->
%<% end %>
```

7.6. ERB compared to rules and classes

Although both ERB and rules/classes enable generating profiles dynamically, in general ERB profiles are easier to read and understand. One important difference is that rules and classes can merge profiles, and ERB cannot. See more about merging profiles in *Chapter 6, Rules and classes*. On the other hand, ERB brings all the power of a high-level language, Ruby. Let's see an example using both. In the following example, we want to place `/home` directory in `/dev/sdb` if it exists.

Example 7.13. Rules and classes

```
<rule>
  <custom1>
    <script>
if blkid | grep /dev/sdb > /dev/null; then
echo -n "yes"
else
echo -n "no"
fi;
    </script>
    <match>yes</match>
    <match_type>exact</match_type>
  </custom1>
  <result>
    <profile>classes/sdb_home.xml</profile>
    <dont_merge config:type="list">
      <element>partition</element>
    </dont_merge>
  </result>
</rule>
```

Example 7.14. ERB

```
<% home_in_sdb = disks.map { |d| d[:device] }.include?("sdb") %>

<partitioning config:type="list">
  <drive>
    ...
  </drive>
  <% if home_in_sdb %>
  <drive>
    <device>/dev/sdb</device>
    <disklabel>none</disklabel>
    <partitions t="list">
      <partition>
        <format t="boolean">true</format>
        <filesystem t="symbol">xfs</filesystem>
        <mount>/home</mount>
      </partition>
    </partitions>
  </drive>
  <% end %>
</partitioning>
```

Chapter 8. Combining ERB templates and scripts

the section called "Pre-scripts" already describes how to use a pre-script to modify the current profile. In a nutshell, if the script creates a `/tmp/profile/modified.xml` file, AutoYaST imports that profile and forgets about the initial one.

This is a pretty flexible approach and the only limitation is that you need to rely on the languages and libraries that are available in the installation media.

8.1. Embedding ERB in your scripts

Unlike with *rules*, it is possible to combine ERB templates with scripts. AutoYaST will evaluate any ERB tag that you include in your script before running it. This behavior only applies to the scripts defined inside the profile and not to the external ones.

The script in the example below downloads a profile whose name is based on the MAC address. Saving the file as `/tmp/profile/modified.xml` will cause AutoYaST to load and use the downloaded profile.

Example 8.1. Using the MAC address to get the profile

```
<scripts>
  <pre-scripts config:type="list">
    <script>
      <interpreter>shell</interpreter>
      <filename>load_profile.sh</filename>
      <% mac = network_cards.first >
      <source><![CDATA[#!/bin/bash
wget -O /tmp/profile/modified.xml http://myserver/<%= network_cards.first[:mac]
%>.xml
]]></source>
    </script>
  </pre-scripts>
</scripts>
```

8.2. Accessing ERB helpers from Ruby scripts

It is possible to use the ERB helpers in Ruby scripts. To use those helpers, you need to *require* the `yast` and `autoinstall/y2erb` libraries and use the `Y2Autoinstall::Y2ERB::TemplateEnvironment` class to access them.

Example 8.2. Accessing ERB helpers from a Ruby script

```
<scripts>
  <pre-scripts config:type="list">
    <script>
      <interpreter>/usr/bin/ruby</interpreter>
      <filename>load_profile.rb</filename>
      <source><![CDATA[#!/usr/bin/env ruby
require "yast"
require "autoinstall/y2erb"
helpers = Y2Autoinstallation::Y2ERB::TemplateEnvironment.new
# Now you can use the TemplateEnvironment instance to call the helpers
disk_devices = helpers.disks.map { |d| d[:device] }
File.write("/root/disks.txt", disk_devices.join("\n"))
]]></source>
    </script>
  </pre-scripts>
</scripts>
```

Part IV. Understanding the auto-installation process

Chapter 9. The auto-installation process

9.1. 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.

9.1.1. X11 interface (graphical)

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

9.1.2. 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.

9.1.3. 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.

9.2. 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`*.

Alternatively, you can place the `autoinst.xml` into a device, mounted either physically or virtually, that is labeled `OEMDRV`. In this case, you do not need to specify the location of the `autoinst.xml` on the kernel command line. The `autoinst.xml` must be located in the root directory of the device.

9.2.1. 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.

Copying the installation medium image to a removable flash disk

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

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

IMAGE needs to be replaced with the path to the `SLE-15-SP7-Online-`*`ARCH`*`-GM-media1.iso` or `SLE-15-SP7-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:

```
#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:

```
#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-SP7-Full-`*`ARCH`*`-GM-media1.iso` would be:

```
dd if=SLE-15-SP7-Full-ARCH-GM-media1.iso of=/dev/sdc bs=4M && sync
```

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

9.2.2. Booting from the SUSE Linux Enterprise installation medium

You can use the SUSE Linux Enterprise installation medium (`SLE-15-SP7-Online-`*`ARCH`*`-GM-media1.iso` or `SLE-15-SP7-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.

9.2.3. Booting via PXE over the network

Booting via PXE requires a DHCP and a TFTP server in your network. The computer will then boot without a physical medium. For instructions on setting up the required infrastructure, see Chapter 13, Remote installation in “Deployment Guide”.

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.

9.3. Invoking the auto-installation process

9.3.1. 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:

autoupgrade

Initiate an automatic upgrade using AutoYaST; see *the section called "Upgrade"*.

autoyast

Location of the control file for automatic installation; see *AutoYaST control file locations* for details.

ifcfg

Configure and start the network. Required if the AutoYaST is to be fetched from a remote location. See *the section called "Advanced network setup"* for details.

insmod

Kernel modules to load

install

Location of the installation directory, for example `install=nfs://192.168.2.1/CDs/`.

Disabling SSL checks

When you are using HTTPS, SSL checking is enabled by default. If necessary, you can disable SSL checking by appending `ssl_verify=no` to your HTTPS URL, like the following examples:

```
install=https://192.168.2.1/CDs/?ssl_verify=no
```

If you are passing multiple query options, separate them with ampersands:

```
install=https://192.168.2.1/CDs/?foo=bar&ssl_verify=no
```

See the "FTP/HTTP/HTTPS directory tree" section of **`man 8 zypper`** for more information.

instmode

Installation mode, for example `nfs`, `http` etc. (not needed if `install` is set).

rootpassword

Password for root user if not specified in AutoYaST profile

server

Server (NFS) to contact for source directory

serverdir

Directory on NFS Server

y2confirm

Even with <confirm>no</confirm> in the control file, the confirm proposal comes up.

These variables and keywords will bring the system up to the point where YaST can take over with the main control file. Currently, the source medium is automatically discovered, which in some cases makes it possible to initiate the auto-install process without giving any instructions to **`linuxrc`**.

The traditional **`linuxrc`** configuration file (`info`) has the function of giving the client enough information about the installation server and the location of the sources. Usually, this file is not required, but it is needed in special network environments where DHCP and BOOTP are not used or when special kernel modules need to be loaded.

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.

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=repo:/*PATH*

Retrieve the control file from the specified path. The path must be relative to the installation source.

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 SUSE Linux Enterprise Server (`SLE-15-SP7-Online-`*`ARCH`*`-GM-media1.iso` or `SLE-15-SP7-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 SUSE Linux Enterprise Server installation media

When using the original installation media (`SLE-15-SP7-Online-`*`ARCH`*`-GM-media1.iso` or `SLE-15-SP7-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.

NetworkAccess 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://..`.

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.

   ```
   >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 9.1. Determine HEX code for an IP address

```
>/usr/bin/gethostip 10.10.0.1
10.10.0.1 10.10.0.1 0A0A0001
```

9.3.2. Auto-installing a single system

The easiest way to auto-install a system without any network connection is to use the original SUSE Linux Enterprise Server DVD-ROMs and a flash disk (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.

9.3.3. 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 9.2. `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.

9.4. 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.

9.4.1. 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 *the section called “General options”*).

9.4.2. 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.

Part V. Uses for AutoYaST on installed systems

Chapter 10. Running AutoYaST in an installed system

In some cases it is useful to run AutoYaST in a running system. Keep in mind that the `partitioning` section is ignored in this scenario.

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 (*the section called "Installing packages in stage 2"*), a user (*the section called "Users"*) section and an NTP client (*the section called "NTP client"*) section:

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://
www.suse.com/1.0/configns">
  <ntp-client>
    <peers config:type="list">
      <peer>
        <address>us.pool.ntp.org</address>
        <comment/>
        <options> iburst</options>
        <type>server</type>
      </peer>
    </peers>
    <start_at_boot config:type="boolean">true</start_at_boot>
    <start_in_chroot config:type="boolean">false</start_in_chroot>
    <sync_interval config:type="integer">5</sync_interval>
    <synchronize_time config:type="boolean">false</synchronize_time>
  </ntp-client>
  <software>
    <post-packages config:type="list">
      <package>ntp</package>
      <package>yast2-ntp-client</package>
      <package>foo</package>
    </post-packages>
  </software>
  <users config:type="list">
    <user>
      <encrypted config:type="boolean">false</encrypted>
      <fullname>Foo user</fullname>
      <gid>100</gid>
      <home>/home/foo</home>
      <password_settings>
        <expire/>
        <flag/>
        <inact/>
        <max>99999</max>
        <min>0</min>
        <warn>7</warn>
      </password_settings>
      <shell>/bin/bash</shell>
      <uid>1001</uid>
      <user_password>linux</user_password>
      <username>foo</username>
    </user>
  </users>
</profile>
```

Store this file as `/tmp/install_foo.xml` and start the AutoYaST installation process by calling:

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

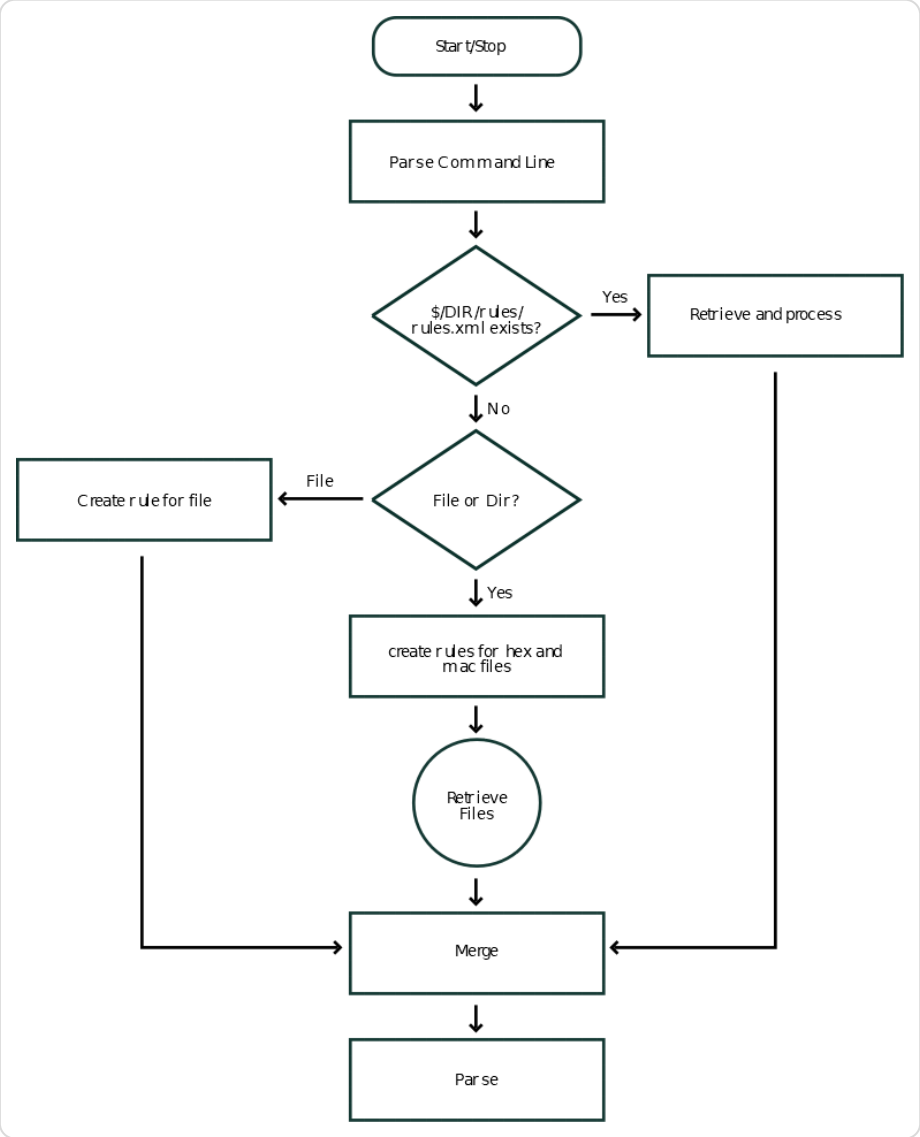
For more information, run **yast2 ayast_setup longhelp**

Part VI. Appendixes

Appendix A. Handling rules

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

Figure A.1. Rules retrieval process



Appendix B. AutoYaST FAQ—frequently asked questions

1.

How do I invoke an AutoYaST installation?

On all SUSE Linux Enterprise Server versions, the automatic installation gets invoked by adding `autoyast=<PATH_TO_PROFILE>` to the kernel parameter list. So for example adding `autoyast=`*`http://MYSERVER/MYCONFIG.xml`* will start an automatic installation where the profile with the AutoYaST configuration gets fetched from the Web server `myserver`. See *the section called “Invoking the auto-installation process”* for more information.

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.

3.

How do I create an AutoYaST profile?

The easiest way to create an AutoYaST profile is to use an existing SUSE Linux Enterprise Server system as a template. On an already installed system, start *YaST* > *Miscellaneous* > *Autoinstallation*. Now select *Tools* > *Create Reference Profile* from the menu. Choose the system

components you want to include in the profile. Alternatively, create a profile containing the complete system configuration by running **`sudo yast clone_system`** from the command line.

Both methods will create the file `/root/autoinst.xml`. The version created on the command line can be used to set up an identical clone of the system on which the profile was created. However, usually you will want to adjust the file to make it possible to install several machines that are very similar, but not identical. This can be done by adjusting the profile using your favorite text/XML editor.

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 *the section called "Creating/editing a control file manually"* for details.

5.

What is smallest AutoYaST profile that makes sense?

If a section has not been defined in the AutoYaST profile the settings of the general YaST installation proposal will be used. However, you need to specify at least the `root` password to be able to log in to the machine after the installation.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://
www.suse.com/1.0/configns">
  <users config:type="list">
    <user>
      <encrypted config:type="boolean">false</encrypted>
      <user_password>linux</user_password>
      <username>root</username>
    </user>
  </users>
</profile>
```

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>
```

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`.

8.

How can I test a merging process on the command line?

To merge two profiles, `a.xml` with `base.xml`, run the following command:

```
>/usr/bin/xsltproc --novalid --param replace "'false'" \
--param dontmerge1 "'package'" --param with "'a.xml'" --output out.xml \
/usr/share/autoinstall/xslt/merge.xslt base.xml
```

This requires sections in both profiles to be in alphabetical order (<software>, for example, needs to be listed after <add-on>). If you have created the profile with YaST, profiles are automatically sorted correctly.

The `dontmerge1` parameter is optional and an example of what to do when you use the `dont_merge` element in your profile. See *the section called "Merging of rules and classes"* for more information.

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>
```

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.

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).

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 *the section called "Advanced network setup"* for more information.

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/configns">
  <partitioning config:type="list">
    <drive>
      <device>/dev/nfs</device>
      <initialize config:type="boolean">false</initialize>
      <type config:type="symbol">CT_NFS</type>
      <partitions config:type="list">
        <partition>
          <filesystem config:type="symbol">nfs</filesystem>
          <fstopt>nolock</fstopt>
          <device>10.10.1.53:/tmp/m4</device>
          <mount>/</mount>
        </partition>
      </partitions>
      <use>all</use>
    </drive>
  </partitioning>
</profile>
```

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 https://lists.opensuse.org/opensuse-autoinstall/.

Appendix 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.

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.

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
`addswap: 0|3|/dev/sda5`	If 0, never ask for swap; if the argument is a positive number n, activate the swap partition; if the argument is a partition name, activate this swap partition.
`autoyast: ftp://AUTOYASTFILE`	Location of the auto installation file; activates auto installation mode. See *AutoYaST control file locations* for details.
`bootptimeout: 10`	10 seconds timeout for BOOTP requests.
`bootpwait: 5`	Sleep 5 seconds between network activation and starting bootp.
`display: color|mono|alt`	Set the menu color scheme.
`exec: COMMAND`	Run *command*.
`forceinsmod: 0|1`	Use the `-f` option (force) when running **insmod** commands.
`forcerootimage: 0|1`	Load the installation system into RAM disk.
`ifcfg: NETWORK_CONFIGURATION`	Set up and start the network. See *the section called "Advanced network setup"* for more information.

Keyword: Example Value	Description
`insmod: MODULE`	Load *MODULE*.
`install: URL`	Install from the repository specified with *URL*. For the syntax of *URL* refer to https://en.opensuse.org/SDB:Linuxrc#url_descr.
`keytable: de-lat1-nd`	Virtual console keyboard map to load.
`language: de_DE`	Language preselected for the installation.
`loghost: 10.10.0.22`	Enable remote logging via syslog via UDP port 514
`loghost: @10.10.0.22`	Enable remote logging via syslog via TCP port 514
`memloadimage: 50000`	Load installation system into RAM disk if free memory is above 50000 KB.
`memlimit: 10000`	Ask for swap if free memory drops below 10000 KB.
`memYaST: 20000`	Run YaST in text mode if free memory is below 20000 KB.
`memYaSTText: 10000`	Ask for swap before starting YaST if free memory is below 10000 KB.
`proxy: http://10.10.0.1:3128`	Defines an HTTP proxy server. For the full parameter syntax, refer to https://en.opensuse.org/SDB:Linuxrc#p_proxy.
`rescue: 1|nfs://server/dir`	Load the rescue system; the URL variant specifies the location of the rescue image explicitly.
`rescueimage: /suse/images/rescue`	Location of the rescue system image.
`rootimage: /suse/images/root`	Location of the installation system image.
`textmode: 1`	Start YaST in text mode.
`usbwait: 4`	Wait four seconds after loading the USB modules.
`y2confirm`	Overrides the confirm parameter in a control file and requests confirmation of installation proposal.

C.3. Advanced network setup

Even if parameters like `hostip`, `nameserver`, and `gateway` are passed to **`linuxrc`**, the network is only started when it is needed (for example, when installing via SSH or VNC). 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 is done by using the parameter `ifcfg`. `ifcfg` directly controls the content of the `/etc/sysconfig/network/ifcfg-*` files.

DHCP network configuration

The general syntax to configure DHCP is

```
ifcfg=INTERFACE=DHCP*,OPTION1=VALUE1,OPTION2=VALUE2
```

where *INTERFACE* is the interface name, for example `eth0`, or `eth*` for all interfaces. *DHCP** can either be `dhcp` (IPv4 and IPv6), `dhcp4`, or `dhcp6`.

To set up DHCP for `eth0` use:

```
ifcfg=eth0=dhcp
```

To set up DHCP on all interfaces use:

```
ifcfg=eth*=dhcp
```

Static network configuration

The general syntax to configure a static network is

```
ifcfg=INTERFACE=IP_LIST,GATEWAY_LIST,NAMESERVER_LIST,DOMAINSEARCH_LIST,\
OPTION1=value1,...
```

where *INTERFACE* is the interface name, for example `eth0`. If using `eth*`, the first device available will be used. The other parameters need to be replaced with the respective values in the given order. Example:

```
ifcfg=eth0=192.168.2.100/24,192.168.5.1,192.168.1.116,example.com
```

When specifying multiple addresses for a parameter, use spaces to separate them and quote the complete string. The following example uses two name servers and a search list containing two domains.

```
ifcfg="eth0=192.168.2.100/24,192.168.5.1,192.168.1.116
192.168.1.117,example.com example.net"
```

For more information refer to https://en.opensuse.org/SDB:Linuxrc#Network_Configuration.

Appendix D. Differences between AutoYaST profiles in SLE 12 and 15

D.1. Product selection

For backward compatibility with profiles created for pre-SLE 15 products, AutoYaST implements a heuristic that selects products automatically. This heuristic will be used when the profile does not contain a `product` element. Automatic product selection is based on the package and pattern selection in the profile. However, whenever possible, avoid relying on this mechanism and adapt old profiles to use explicit product selection.

For information about explicit product selection, refer to *the section called "Product selection"*.

If automatic product selection fails, an error is shown and the installation will not be continued.

D.2. Software

The SUSE Linux Enterprise Server15 SP7 installation medium only contains a very minimal set of packages to install. This minimal set only provides an installation environment and does not include any server applications or advanced tools. Additional software repositories, providing more packages are offered as "modules" or "extensions". They are provided via two alternatives:

- via a registration server (the SUSE Customer Center or a local SMT/RMT proxy)
- via the SLE-15-SP7-Full-*ARCH*-GM-media1.iso image containing all modules and extensions. Using this medium does not require access to a registration server during installation. It can be shared on the local network via an installation server.

Maintenance updates

Only using the registration server will grant access to all maintenance updates at installation time. Maintenance updates are not available when using the DVD medium without registration.

D.2.1. Adding modules or extensions using the registration server

To add a module or extension from the registration server, use the `addons` tag for each module/extension in the `suse_register` section. Extensions require an additional registration code, which can be specified with the `reg_code` tag.

The following XML code adds the Basesystem and Server Applications modules and the Live Patching extension. To get the respective values for the tags `name`, `version`, and `arch`, run the command **`SUSEConnect --list-extensions`** on a system that already has SLE 15 SP7 installed.

Example D.1. Adding modules and extensions (online)

```
<suse_register>
 <addons config:type="list">
  <addon>
   <name>sle-module-basesystem</name>
   <version>15.7</version>
   <arch>x86_64</arch>
  </addon>
  <addon>
   <name>sle-module-server-applications</name>
   <version>15.7</version>
   <arch>x86_64</arch>
   </addon>
  <addon>
   <name>sle-module-live-patching</name>
   <version>15.7</version>
   <arch>x86_64</arch>
   <reg_code>REGISTRATION_CODE</reg_code>
  </addon>
 </addons>
</suse_register>
```

Refer to *the section called “System registration and extension selection”* for more information.

D.2.2. Adding modules or extensions using the SLE-15-SP7-Full-*ARCH*-GM-media1.iso image

To add a module or extension using the SLE-15-SP7-Full-*ARCH*-GM-media1.iso image, create entries in the add-on section as shown in the example below. The following XML code adds the Basesystem module from a USB flash drive that contains the image:

Example D.2. Adding modules and extensions (offline)

```
<add-on>
 <add_on_products config:type="list">
  <listentry>
   <media_url><![CDATA[dvd:///?devices=/dev/sda%2C/dev/sdb%2C/dev/sdc%2C/dev/
sdd]]></media_url>
   <product_dir>/Module-Basesystem</product_dir>
   <product>sle-module-basesystem</product>
  </listentry>
 </add_on_products>
</add-on>
```

Product name matching

The `product` tag must match the internal product name contained in the repository. If the product name does not match at installation AutoYaST will report an error.

Using the installation media image from a local server

You can share the content of the USB flash drive on the local network via an NFS, FTP or HTTP server. To do so, replace the URL in the `media_url` tag so it points to root of the medium on the server, for example:

```
<media_url>ftp://ftp.example.com/sle_15_sp7_full/</media_url>
```

D.2.3. Renamed software patterns

Software patterns have also changed since SUSE Linux Enterprise Server 15. Some patterns have been renamed; a short summary is provided in the following table.

Old SLE 12 Pattern	New SLE 15 Pattern
Basis-Devel	devel_basis
gnome-basic	gnome_basic
Minimal	enhanced_base
printing	print_server
SDK-C-C++	devel_basis
SDK-Doc	technical_writing
SDK-YaST	devel_yast

Carefully check if all required packages are available in the defined patterns and adjust the profiles accordingly. Additionally, the required patterns and packages need to be available in the activated extensions or modules.

Notes

- The pattern renames in the table above are not 1:1 replacements; the content of some patterns has been changed as well, some packages were moved to a different pattern or even removed from SUSE Linux Enterprise Server 15.
- Check that the required packages are still included in the used patterns, and optionally use the `packages` tag to specify additional packages.
- The list above might be incomplete, as some products have not been released for SUSE Linux Enterprise Server 15, yet.

D.3. Registration of module and extension dependencies

Starting with SUSE Linux Enterprise Server 15, AutoYaST automatically reorders the extensions according to their dependencies during registration. This means the order of the extensions in the AutoYaST profile is not important.

Also AutoYaST automatically adds the dependent modules even though they are missing in the profile. This means you are not required to specify all required modules. However, if an extension depends on another extension, it needs to be specified in the profile, including the registration key. Otherwise the registration would fail.

You can list the available extensions and modules in a registered system using the **`SUSEConnect --list-extensions`** command.

D.4. 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, SUSE Linux Enterprise Server 15 comes with some general changes, which are explained below.

D.4.1. 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 prefer to retain the old behavior, you can explicitly indicate this in the profile by setting the `disklabel` element to `msdos`.

D.4.2. 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 *the section called “Partition configuration”* for more information.

D.4.3. Forcing primary partitions

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

D.4.4. 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.3. 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.4. 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.4.5. 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.4.6. Reading an existing `/etc/fstab` is no longer supported

On SUSE Linux Enterprise Server 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.4.7. 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.4.8. Using the `type` to define a 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 called "Logical volume manager (LVM)"* for further details.

D.5. Firewall configuration

In SUSE Linux Enterprise Server 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 SLE 15 as outlined below. If keeping SLE12 profiles, we recommend to check the final configuration to avoid unexpected behavior or network security threats.

Table D.1. AutoYaST firewall configuration in SLE 15: backward compatibility

Supported (but deprecated)	Unsupported
`FW_CONFIGURATIONS_\{DMZ, EXT, INT}`	`FW_ALLOW_FW_BROADCAST_\{DMZ, EXT, INT}`
`FW_DEV_\{DMZ, EXT, INT}`	`FW_IGNORE_FW_BROADCAST_\{DMZ, EXT, INT}`
`FW_LOG_DROP_ALL`	`FW_IPSECT_TRUST`
`FW_LOG_DROP_CRIT`	`FW_LOAD_MODULES`
`FW_MASQUERADE`	`FW_LOG_ACCEPT_ALL`
`FW_SERVICES_\{DMZ, INT, EXT}_\{TCP, UDP, IP}`	`FW_LOG_ACCEPT_CRIT`
	`FW_PROTECT_FROM_INT`
	`FW_ROUTE`
	`FW_SERVICES_\{DMZ, EXT, INT}_RPC`
	`FW_SERVICES_ACCEPT_RELATED_\{DMZ, EXT, INT}`

Configuration options from SuSEfirewall2 that are no longer available either have no equivalent mapping in `firewalld` or will be supported in future releases of SUSE Linux Enterprise Server. Some `firewalld` features are not yet supported by YaST and AutoYaST—you can use them with

post installation scripts in your AutoYaST profile. See *the section called "Custom user scripts"* for more information.

Enabling and starting the firewall

Enabling and starting the `systemd` service for `firewalld` is done with the same syntax as in SLE 12. 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 SLE 15 syntax:

D.5.1. 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

`firewalld` (SLE 15)	**SuSEfirewall2 (SLE 12)**
dmz	DMZ
external	EXT with `FW_MASQUERADE` set to `yes`
public	EXT with `FW_MASQUERADE` set to `no`
internal	INT with `FW_PROTECT_FROM_INT` set to `yes`
trusted	INT with `FW_PROTECT_FROM_INT` set to `no`
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.5.1.1. Default configuration

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

Example D.5. 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.6. Assigning zones: default configuration (SLE 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.5.1.2. 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.7. 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.8. Masquerading and protecting internal zones (SLE 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.5.2. 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.9. 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.10. Opening ports (SLE 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.5.3. 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.11. 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.12. Opening services (SLE 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
- `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.5.4. More information

- SuSEfirewall2/AutoYaST Documentation for SLE 12
- Official `firewalld` Documentation

D.6. 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 SLE12 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.13. NTP configuration (SLE 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.7. 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. 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.8. The CA management module has been dropped

The module for CA Management (`yast2-ca-management`) has been removed from SUSE Linux Enterprise Server 15, and for the time being there is no replacement available. In case you are reusing SLE12 profile, make sure it does not contain a `ca_mgm` section.

D.9. Upgrade

D.9.1. Software

SLE 12 has two modes of evaluating which packages need to be upgraded. In SUSE Linux Enterprise Server15 SP7, 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.

D.9.2. Registration

When upgrading a registered system, all old repositories are removed. This is done to avoid possible conflicts between the new and old repositories and to clean-up the repositories for the dropped products. If you need to keep custom repositories, add them again using the `add-on` option.

Example D.14. Minimal registration configuration for upgrade

```
<suse_register>
  <do_registration config:type="boolean">true</do_registration>
</suse_register>
```

If the registration server returns more than one possible migration target, AutoYaST will automatically select the first one. Currently you cannot select a different migration target.

After upgrading an unregistered system or skipping registration upgrade by omitting the `suse_register` option, you might need to adjust the repository setup manually.

Appendix E. GNU licenses

This appendix contains the GNU Free Documentation License version 1.2.

E.1. GNU Free Documentation License

0. PREAMBLE

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall

subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

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

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

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

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

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

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

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

4. MODIFICATIONS

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

1. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
2. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
3. State on the Title page the name of the publisher of the Modified Version, as the publisher.
4. Preserve all the copyright notices of the Document.
5. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
6. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
7. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
8. Include an unaltered copy of this License.
9. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
10. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years

before the Document itself, or if the original publisher of the version it refers to gives permission.

11. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
12. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
13. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
14. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
15. Preserve any Warranty Disclaimers.

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special

permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

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

9. TERMINATION

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

ADDENDUM: How to use this License for your documents

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

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

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

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

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