Automated Installation Using Agama

WHAT?

This article describes how to automatically install SUSE Linux Enterprise Server using Agama. The information presented here applies to unattended installation of the product on both bare metal and virtual machines.

WHY?

Read this article to understand the process of automatically installing SUSE Linux Enterprise Server using Agama.

EFFORT

You need about 30 minutes to read and understand the most important sections of this article. The time required for customization of Agama profiles and storage configuration depends on deployment requirements.

GOAL

Learn how to perform automated or unattended installation of SUSE Linux Enterprise Server using Agama.

REQUIREMENTS

- A bare-metal server or a virtual machine. For server installations without any desktop environment, SUSE recommends a minimum of 1 CPU, 2 GB memory and 32 GB storage (which includes storage for Btrfs snapshots in the root partition, swap space, and storage for software packages).
- An image file for the product you want to install, downloaded from the SUSE Customer Center.
- An active registration code for the product you want to install. You can generate a registration code for the product and activate its subscription for your organization at the SUSE Customer Center.

Note: Optional registration

Certain images that are signed with the developer's key may allow you to skip registration before or during installation. Besides, certain images may contain all installable packages for your operating system that you can use as an offline package repository. In such cases, you may not need an active registration code before installation. However, if you use software packages from the official online repositories, SUSE recommends registering your product with the SUSE Customer Center.

Publication Date: 07 Nov 2025

Contents

1 Introduction to automated installation using Agama

This article describes how to use Agama for automated and unattended installation of SUSE Linux Enterprise Server. You can use JSON *profiles* describing different aspects of the intended system, and Agama installs accordingly. While not a fully backward compatible replacement for AutoYaST, it simplifies the task of automated installation and provides multiple clients for interactive and automated installation.

1.1 What is Agama

Agama is a service-based Linux installer capable of performing both interactive and unattended installations. You can provide Agama with a JSON profile file detailing the initial system state, such as user authentication, partitioning, networking and software selection. On receiving the profile and instructions for installation from one of its supported clients, Agama installs your target system accordingly. Users can interact with and control the installation process using Agama's Web-based user interface, command-line interface and HTTP API, facilitating automation and integration into existing workflows.

While Agama reuses many principles and internal components from previous SUSE installers like YaST and AutoYaST, and offers a high level of backward compatibility for unattended installations, it is not a 100% compatible drop-in replacement for all AutoYaST features. Agama focuses only on the installation process rather than being a general configuration tool.

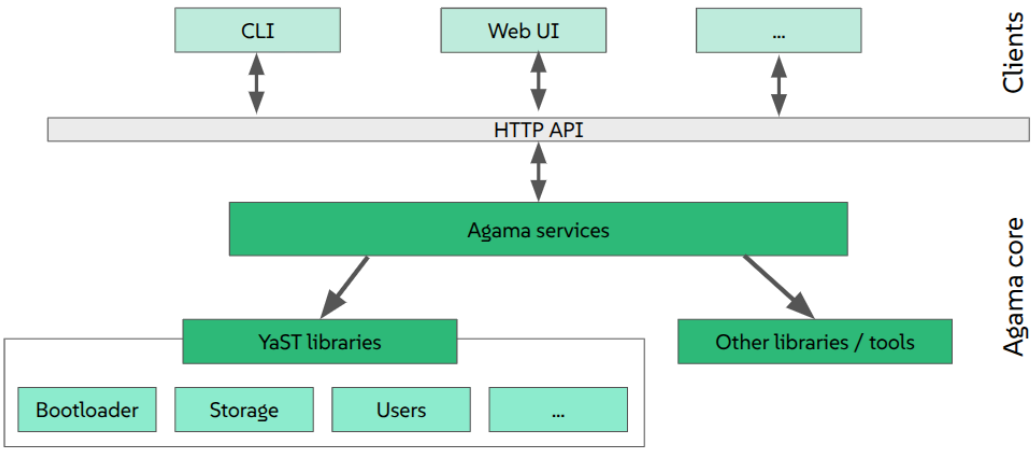

FIGURE 1: AGAMA CLIENT-SERVER ARCHITECTURE

1.2 Why use Agama for automated installation?

Agamaoffers its installation service through an HTTP API, which you can use interactively from a Web-based interface and a command-line interface (CLI), or provide a JSON profile to Agama for automated installation of a target system. Using the HTTP API, you can also integrate with custom scripts and deployment tools. The benefits of using Agama for automated installation are as follows:

Focus on core installation

Agama focuses on core installation tasks such as user authentication, network configuration, storage setup and software installation, delegating further configuration to other tools such as Ansible, Salt, Cockpit or OpenSCAP.

Profile-based installation

You can define installation parameters for the target system in an easily readable and editable JSON or Jsonnet profile. Existing XML-based AutoYaST profiles are also supported with some exceptions.

Comprehensive profile configuration

The profile allows detailed setup including user authentication, product registration, network connections, storage (drives, partitions, LVM, RAID, encryption, resizing, deleting), software selection by patterns and packages, localization (language, keyboard, time zone) and many other aspects of the target system that are not exposed in the graphical or Web-based interface. This provides a more granular level of control over the installation parameters.

Dynamic profiles

Agama supports dynamic profiles using Jsonnet, injecting hardware information that can be processed at runtime. This avoids reliance on AutoYaST's rules or ERB for dynamic configurations.

AutoYaST compatibility

Agama offers a mechanism to reuse existing AutoYaST profiles to a great extent. It supports some dynamic features such as pre-scripts, rules/classes, and Embedded Ruby (ERB) when using AutoYaST profiles. A `legacyAutoyastStorage` section allows direct use of the AutoYaST profile's `partitioning` section for backward compatibility.

Custom scripts

Profiles can define pre-installation, post-partitioning, chroot, and init scripts that run at specific stages. You can include scripts by URL, location in the hard drive, or embed the script content in the profile itself.

Easy initiation

The typical way to start an unattended installation from an ISO image is using the `inst.auto` kernel boot option, pointing to the profile URL or its location in the hard drive. You can also use the **`agama profile import`** command from the Agama CLI to load a profile, followed by the **`agama install`** command. The CLI also allows inspection, modification and validation of the profile, and subsequent monitoring of the installation process.

2 Understanding Agama installation profiles

For automated installations, Agama relies on a *profile*, which is a configuration file that specifies how the system should be set up. This profile describes various aspects of the installation, including partitioning, networking, software selection, and other options. The concept of using a profile for automated installation is similar to AutoYaST. Agama focuses specifically on the installation process itself and delegates further system configuration to other tools. Agama aims for a high level of backward compatibility with AutoYaST profiles for automated installations.

Note: Difference between Agama and AutoYaST profiles

Agama and AutoYaST profiles are largely compatible for all common use cases. However, Agama profiles are not fully compatible with AutoYaST profiles, and cannot be used as a drop-in replacement without checking the compatibility. There are certain aspects of the AutoYaST profiles that are currently supported in Agama profiles, or may be supported in the future. However, there are certain other aspects that are neither currently supported in Agama profiles, nor will be supported in the future. For more information, refer to the section *Section 8, "Compatibility between AutoYaST and Agama profiles"*.

2.1 Introduction to the Agama profile structure

The Agama profile configuration is defined using a JSON document. It contains several sections that are necessary for describing the installation parameters for a customized system. At a high level, the profile consists of the following sections:

```
{
  "product": {}, (1)
  "root": {}, (2)
  "user": {}, (3)
  "localization": {}, (4)
  "hostname": {}, (5)
  "software": {}, (6)
  "storage": {}, (7)
  "bootloader": {}, (8)
  "network": {}, (9)
  "security": {}, (10)
  "scripts": {}, (11)
  "files": {}, (12)
  "legacyAutoyastStorage": {}, (13)
  "iscsi": {}, (14)
  "dasd": {} (15)
}
```

1. `product`: Identifies the OS/product to be installed.
2. `root`: Root credentials for administrative access.
3. `user`: First non-root user account.
4. `localization`: Language, keyboard, and time zone settings.
5. `hostname`: Static/transient hostname settings.
6. `software`: Packages and patterns to be installed.
7. `storage`: Partitioning and mount configuration.
8. `bootloader`: Boot loader config and kernel params.
9. `network`: Network interface configuration.
10. `security`: SSL certs and other security settings.
11. `scripts`: Pre/post install scripting.
12. `files`: Inject additional files post-install.
13. `legacyAutoyastStorage`: Support for legacy AutoYaST JSON-style storage.
14. `iscsi`: iSCSI disk/target configuration.

15 `dasd`: DASD disk support for IBM Z (s390x).

You can also describe profiles using Jsonnet, which is a superset of JSON. Jsonnet offers features like variables, functions and more convenient syntax, making profiles more readable, concise, and dynamic for injecting hardware information at runtime.

For more information on the JSON and Jsonnet profiles, refer to the resources mentioned in *Section 9, "For more information"*. The upstream resources usually contain most updated information and examples about the profiles.

2.2 A minimal example of an Agama profile

A minimal Agama JSON profile must at least include sections for product identification, product registration, and credentials for the root user. Agama uses the defaults for the rest of the profile. As a best practice, you should also configure the following:

A host name
A non-root user
Minimal localization settings

EXAMPLE 1: A MINIMAL AGAMA PROFILE FOR AUTOMATED INSTALLATION

```
{
  "product": {
    "id": "SUSE Linux Enterprise Server 16.0",
    "registrationCode": "REGISTRATION-CODE",
    "registrationEmail": "EMAIL",
  },
  "hostname": {
    "static": "STATIC-HOSTNAME",
    "transient": "TRANSIENT-HOSTNAME"
  },
  "root": {
    "hashedPassword": true,
    "password": "HASHED-ROOT-PASSWORD", (1)
    "sshPublicKey": "SSH-PUBLIC-KEY", (2)
  },
  "user": {
    "hashedPassword": false,
    "autologin": false,
    "fullName": "FULL-NAME",
    "userName": "USERNAME",
    "password": "PLAINTEXT-PASSWORD"
  },
  "localization": {
```

```
    "language": "LANGUAGE",
    "keyboard": "KEYBOARD-LAYOUT",
    "timezone": "TIMEZONE"
  }
}
```

1. You can generate a hashed password by running the following command:

   ```
   > sudo openssl passwd -6
   ```

2. You can generate an SSH public key by running the following command:

   ```
   > sudo ssh-keygen -t rsa -b 4096 -C "YOUR-EMAIL@EXAMPLE.COM"
   ```

   Based on your requirements, choose the key type and the key size. However, it is better to adopt stronger security.

To evaluate the correctness of the profile, run the following command:

```
> sudo agama profile validate AGAMA-PROFILE.json
```

3 Details of an Agama installation profile

The Agama profile contains various sections to configure different aspects of the system installation. For real deployments where you would want to simultaneously install multiple systems with the same initial configuration, prepare a customized profile with all the necessary details. You can start with the template of the minimal example and add details progressively. This section gives you an idea of the most useful details you should consider for real deployments.

3.1 Product configuration for an Agama installation profile

The `product` section defines the SUSE product to be installed and includes optional registration data and add-on modules. This is essential for systems requiring access to subscription repositories or additional functionality.

EXAMPLE 2: SAMPLE `product` CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"product": {
  "id": "SUSE Linux Enterprise Server 16.0",
  "registrationCode": "REGISTRATION-CODE",
```

```
  "registrationEmail": "EMAIL-ADDRESS",
  "registrationUrl": "REGISTRATION-URL",
  "addons": [
    {
      "id": "ADDON-ID",
      "version": "ADDON-VERSION",
      "registrationCode": "ADDON-REGISTRATION-CODE"
    }
  ]
}
```

This section contains the following fields:

id

The product identifier used to select the base SUSE product to be installed.

registrationCode

The registration code for the product obtained from the SUSE Customer Center and used to activate repositories and receive updates.

registrationEmail

The e-mail address associated with the registration account used during product activation.

registrationUrl

The full URL of the registration server. If you are using the SUSE Customer Center, you can omit this field. However, it is useful when registering from a custom server.

addons

A list of *optional* add-on modules or extensions to be activated alongside the base product.

- `id`: Identifier of the add-on. For example, `sle-ha` for High Availability.
- `version`: Specific version of the add-on to be installed. This is required if multiple versions are available.
- `registrationCode`: Optional registration code for the add-on if separate activation is required.

3.2 Host name configuration for an Agama installation profile

The `hostname` section sets the system's static and transient host name. The static host name is persistent across reboots, while the transient host name is used temporarily at runtime and may be overridden by network services like DHCP.

EXAMPLE 3: SAMPLE `hostname` CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"hostname": {
  "static": "STATIC-HOSTNAME",
  "transient": "TRANSIENT-HOSTNAME"
}
```

This section contains the following fields:

static

The persistent host name written to `/etc/hostname`. This name remains consistent across system reboots and is used by default if no transient host name is specified.

transient

A temporary host name applied at runtime. This may be used during deployment or installation to reflect an ephemeral identity. For example, it can be set via DHCP or by installation tooling like Agama.

3.3 Root authentication for an Agama installation profile

The `root` section defines authentication settings for the system's `root` account. This includes a root password (either plain or pre-hashed) and an optional SSH public key for remote access.

EXAMPLE 4: SAMPLE `root` CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"root": {
  "hashedPassword": true,
  "password": "HASHED-ROOT-PASSWORD",
  "sshPublicKey": "SSH-PUBLIC-KEY"
}
```

This section contains the following fields:

hashedPassword

Boolean flag indicating whether the `password` field contains a hashed value. If set to `true`, the password must be a SHA-512 crypt hash (starting with `$6$`). If `false` or omitted, the value is treated as plain text.

password

The root user's password. If `hashedPassword` is `true`, this must be a pre-generated hash (for example, using **`openssl passwd -6`**). Otherwise, plain text is accepted and will be hashed during installation.

You can generate a hashed password by running the following command:

```
> sudo openssl passwd -6
```

sshPublicKey

An optional SSH public key to be added to the root user's `~/.ssh/authorized_keys` file. This allows passwordless root login over SSH. The key must be in OpenSSH format. For example, starting with `ssh-rsa` or `ssh-ed25519`.

You can generate an SSH public key by running the following command:

```
> sudo ssh-keygen -t rsa -b 4096 -C "YOUR-EMAIL@EXAMPLE.COM"
```

Based on your requirements, choose the key type and the key size. However, it is better to adopt stronger security.

3.4 User configuration for an Agama installation profile

The `user` section defines the initial non-root user account created during installation. This includes the user's full name, login name, password (plain or hashed), and optional automatic login preference.

EXAMPLE 5: SAMPLE `user` CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"user": {
  "hashedPassword": false,
  "fullName": "FULL-NAME",
  "userName": "LOGIN-NAME",
  "password": "USER-PASSWORD",
  "autologin": false
}
```

This section contains the following fields:

hashedPassword

Boolean flag indicating whether the `password` field contains a hashed value. If set to `true`, the password must be a pre-hashed SHA-512 crypt value. Otherwise, the plain text password will be hashed during installation.

fullName

The full name of the user, typically used for display purposes in graphical environments. For example, `Jane Doe`.

userName

The system login name for the user. This becomes their Linux username and home directory under `/home`. For example, `jane`.

password

The user's password, in plain text or pre-hashed, depending on the `hashedPassword` flag. If plain-text is provided, it will be automatically hashed.

autologin

Optional boolean that determines whether the user should be automatically logged into a graphical session at boot. This setting is only relevant for desktop installations.

3.5 Localization configuration for an Agama installation profile

The `localization` section defines the system language, keyboard layout, and time zone settings. These parameters determine the default locale and input behavior after installation.

EXAMPLE 6: SAMPLE `localization` CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"localization": {
  "language": "LANGUAGE-ID",
  "keyboard": "KEYBOARD-LAYOUT",
  "timezone": "TIMEZONE"
}
```

This section contains the following fields:

language

The system language and locale specified as a locale string. For example, `en_US.UTF-8` or `de_DE`. This controls messages, number formats, date formats, and default encoding.

keyboard

The default keyboard layout identifier. For example, `us` or `de`. This affects key mapping on both text consoles and graphical desktops.

timezone

The system time zone using a region/location format. For example, `Europe/Berlin`. This sets the default system clock and affects date/time display.

3.6 Software configuration for an Agama installation profile

The `software` section defines which software components are installed on the system via SUSE's pattern and package management infrastructure.

EXAMPLE 7: SAMPLE `software` CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"software": {
  "patterns": [
    "minimal_base",
    "app_server"
  ],
  "packages": [
    "vim",
    "htop",
    "curl"
  ]
}
```

This section contains the following fields:

patterns

A list of software patterns to be installed. Patterns are curated collections of packages designed to serve a functional role. For example, `minimal_base`, `gnome`, `app_server`. For a complete list of patterns available for SUSE Linux Enterprise Server and your target architecture, refer to the SUSE Customer Center.

packages

A list of individual packages to install in addition to those brought in by selected patterns. For example, `vim`, `htop` and `curl`. For a complete list of packages available from the official SUSE Linux Enterprise Server repositories for your target architecture, refer to the SUSE Customer Center.

Note: Trust the GPG key for the Package Hub repository

When enabling SUSE Package Hub during a manual installation, users are prompted to trust the repository's GPG key. To trust the key automatically during an unattended installation, use the following snippet:

```
{
  product: {
    id: 'SLES',
    registrationCode: 'SLES_REG_CODE',
    addons: [
      {
        id: 'PackageHub',
      }
    ]
  },
  questions: {
```

```
    policy: 'auto',
    answers: [
      {
        answer: 'Trust',
        class: 'software.import_gpg',
        data: {
          fingerprint: 'BF3F 9A67 D3A2 FF98 A73F 5E07 488C 583D 287A 0027',
          name: 'openSUSE Backports for SUSE Linux 16 sle-
backports-202500514@opensuse.org',
          id: '488C583D287A0027'
        }
      }
    ]
  }
}
```

3.7 Storage configuration for an Agama installation profile

The `storage` section defines the system's target disk layout, such as devices, partitions and volume management, to be applied during installation. This field references the Agama storage schema, which is referenced from the profile schema.

Note: Advanced storage configuration

An exhaustive description of all possible storage configuration using Agama is beyond the scope of this section, as it will need careful consideration of the storage model schema. For information on advanced storage configuration, refer to *Section 7, "Advanced storage configuration using Agama profiles"*.

EXAMPLE 8: `storage` SECTION SYNTAX FOR AN AGAMA INSTALLATION PROFILE

```
"storage": {
  "drives": [ ... ],
  "volumeGroups": [ ... ],
  "mdRaids": [ ... ],
  "boot": { ... }
}
```

A `storage` section contains several entries describing how to configure the corresponding storage devices, and some extra entries such as `boot` to setup some general aspects that influence the final layout.

Each volume group or software RAID can represent a new logical device to be created, or an existing device from the system to be processed. Entries below `drives` represent devices that can be used as regular disks, which includes removable and fixed disks, SD cards, DASD or zFCP devices, iSCSI disks, and multipath devices. Those entries always correspond to devices that can be found at the system, because Agama cannot create that kind of devices.

Note: Compatibility with legacy AutoYaST storage

In some cases, `storage` can be replaced by the `legacyAutoyastStorage` section. This section supports everything offered by the `partitioning` section of the AutoYaST profile. However, Agama does not validate this special section—be careful to provide valid AutoYaST options.

3.8 Boot loader configuration for an Agama installation profile

The `bootloader` section defines boot-time behavior, including whether to pause at the boot menu and what extra kernel parameters to pass. It affects the installed system's GRUB configuration.

EXAMPLE 9: SAMPLE `bootloader` CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"bootloader": {
  "stopOnBootMenu": false,
  "timeout": 5,
  "extraKernelParams": "KERNEL-PARAMETERS"
}
```

This section contains the following fields:

stopOnBootMenu

Boolean flag that, if set to `true`, forces the system to stop at the GRUB boot menu instead of proceeding automatically. This is useful for debugging or choosing alternate boot options manually.

timeout

Number of seconds the GRUB boot menu is shown before continuing with the default entry. Set to `0` to boot immediately.

extraKernelParams

Additional kernel command-line parameters to append to the default ones during boot. These are passed directly to the Linux kernel. For example, `console=ttyS0` or `quiet splash`.

3.9 Network configuration for an Agama installation profile

The `network` section defines one or more network connections to be configured during installation. Each connection supports IP setup, interface binding, wireless settings, bonding, bridging, and optional enterprise-grade authentication mechanisms like IEEE 802.1X.

EXAMPLE 10: SAMPLE `network` CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"network": {
  "connections": [
    {
      "id": "ETH0-CONNECTION",
      "interface": "eth0",
      "method4": "manual",
      "addresses": ["192.168.100.10/24"],
      "gateway4": "192.168.100.1",
      "nameservers": ["1.1.1.1", "8.8.8.8"],
      "autoconnect": true
    },
    {
      "id": "WIFI-HOME",
      "interface": "wlan0",
      "method4": "auto",
      "wireless": {
        "ssid": "MYSSID",
        "password": "MYWIFIPASSWORD",
        "security": "wpa-psk"
      }
    },
    {
      "id": "BRIDGE0",
      "interface": "br0",
      "method4": "auto",
      "bridge": {
        "stp": true,
        "forwardDelay": 15,
        "ports": ["eth0", "eth1"]
      }
    },
    {
      "id": "BOND0",
      "interface": "bond0",
      "method4": "manual",
      "addresses": ["10.0.0.100/24"],
      "gateway4": "10.0.0.1",
      "bond": {
        "mode": "active-backup",
```

```
        "options": "miimon=100",
        "ports": ["eth2", "eth3"]
      }
    },
    {
      "id": "SECURE-ETH",
      "interface": "eth4",
      "method4": "auto",
      "ieee-8021x": { (1)
        "eap": ["peap"],
        "identity": "USERNAME",
        "password": "PASSWORD",
        "caCert": "/etc/certs/ca.pem"
      }
    }
  ]
}
```

(1)

Note

Support for IEEE 802.1X authentication is intended for advanced enterprise deployments where authentication is required at the link layer, before an IP is assigned. This commonly involves integration with RADIUS and certificate-based trust. Misconfiguration can result in complete network inaccessibility. Refer to `systemd-networkd` documentation for authoritative guidance.

This section contains the following connection attributes:

id

Unique name for the network connection.

interface

Name of the network interface to bind to (for example, `eth0`).

method4

IPv4 addressing method (`auto`, `manual`, etc.).

method6

IPv6 addressing method (`auto`, `manual`, etc.).

addresses

List of static addresses in CIDR format.

gateway4 / gateway6

IPv4 and IPv6 default gateway addresses.

nameservers

List of DNS server IPs.

autoconnect

Boolean. Specifies whether the connection is brought up automatically.

wireless

Defines wireless-specific settings:

- `ssid`: Wi-Fi network name.
- `password`: Wi-Fi passphrase.
- `security`: Key management (for example, `wpa-psk`).

bridge

Defines bridge-specific settings:

- `stp`: Boolean. Enables the Spanning Tree Protocol.
- `forwardDelay`: STP forwarding delay in seconds.
- `ports`: Interfaces to include in the bridge.

bond

Defines bonding configuration:

- `mode`: Bonding mode (for example, `active-backup`).
- `options`: Optional bonding parameters (for example, `miimon=100`).
- `ports`: List of interfaces in the bond.

ieee-8021x

Defines enterprise authentication settings:

- `eap`: List of EAP methods (for example, `peap`, `tls`).
- `identity`: Login identity (typically username).
- `password`: Authentication password (if needed).
- `caCert`: Path to trusted CA certificate.

3.10 Security configuration for an Agama installation profile

The `security` section allows you to add trusted SSL certificates to the installed system. This is useful when connecting to internal package mirrors, registration servers, or other TLS services that require non-default certificate authorities.

EXAMPLE 11: SAMPLE `security` CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"security": {
  "sslCertificates": [
    {
      "fingerprint": "FINGERPRINT",
      "algorithm": "SHA256"
    }
  ]
}
```

This section contains the following fields:

sslCertificates

A list of custom SSL certificates to install into the system's trust store. Each item specifies the certificate's fingerprint and the hashing algorithm used.

- `fingerprint`: The cryptographic fingerprint of the certificate, formatted as a colon-separated hex string. For example, `A8:DE:08:B1:57:52:FE:70:DF:D5:31:EA:E3:53:BB:39:EE:01:FF:B9`.
- `algorithm`: The fingerprint algorithm used to compute the hash. Supported values are `SHA1` and `SHA256`.

Warning: Use SHA256 for better security

`SHA1` is cryptographically broken and should not be used. Use `SHA256` wherever possible.

To verify the fingerprint of a certificate in PEM format:

```
> sudo openssl x509 -in FILE.pem -noout -fingerprint -sha256
```

Trusted certificates can be installed permanently by placing them in `/etc/pki/trust/anchors` and then executing:

```
> sudo update-ca-certificates
```

3.11 Scripts configuration for an Agama installation profile

The `scripts` section allows you to define custom shell scripts to be executed at different stages of the SUSE installation lifecycle. These scripts can be embedded inline, fetched from a URL, and optionally executed in a chroot environment (where applicable).

EXAMPLE 12: SAMPLE `scripts` CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"scripts": {
  "pre": [
    {
      "name": "PRE-CHECK-DISK.sh",
      "content": "#!/bin/bash\nif [ ! -e /dev/sda ]; then echo 'Disk not found' >&2; exit
 1; fi"
    }
  ],
  "postPartitioning": [
    {
      "name": "CREATE-MOUNTS.sh",
      "url": "http://EXAMPLE.COM/SCRIPTS/MOUNTS.sh"
    }
  ],
  "post": [
    {
      "name": "FINALIZE-INSTALL.sh",
      "content": "#!/bin/bash\necho 'Installation complete'",
      "chroot": true
    }
  ],
  "init": [
    {
      "name": "FIRST-BOOT.sh",
      "content": "#!/bin/bash\necho 'System booted for the first time'"
    }
  ]
}
```

This section defines the following script categories:

pre

Scripts executed before the installation begins. Useful for pre-flight checks or environment preparation.

postPartitioning

Scripts run immediately after partitioning is completed, but before packages are installed.

post

Scripts run after installation finishes. These can optionally execute within the target system's root via chroot.

init

Scripts executed during the first boot of the installed system. These are useful for final configuration, logging, or notifications.

Each script object may contain one of the following keys:

name

File name used to identify the script on disk. Required for all scripts.

content

Inline script body. Must begin with a shebang (for example, `#!/bin/bash`).

url

HTTP/HTTPS location from which to fetch the script. Cannot be combined with `content`.

chroot

Boolean. If `true`, the script is executed inside the installed system's root via chroot. Applies only to `post` scripts.

3.12 Files configuration for an Agama installation profile

The `files` section allows deployment of custom user-defined files into the installed system. These files are written just before post-installation scripts run and can be useful for configuring services, dropping keys, or overriding system files.

EXAMPLE 13: SAMPLE `files` CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"files": [
  {
    "destination": "ABSOLUTE-FILE-PATH",
    "content": "FILE-CONTENT",
    "permissions": "0644",
    "user": "USERNAME",
    "group": "GROUPNAME"
  }
]
```

This section contains an array of file definitions. Each entry supports the following fields:

destination

Required absolute path where the file will be written inside the target system. For example, `/etc/MYAPP/CONFIG.YAML`.

content

Inline string representing the content of the file. This field is mutually exclusive with `url`; one of the two must be present.

url

URL (relative or absolute) to fetch the file content from. Used instead of inline `content`. One of `url` or `content` is required.

permissions

Optional file mode string (octal), such as `0644` or `0755`, to set on the created file.

user

Optional owner username to assign to the file. The user must already exist in the installed system.

group

Optional owner group to assign to the file. The group must already exist in the installed system.

3.13 Legacy AutoYaST storage configuration for an Agama installation profile

The `legacyAutoyastStorage` section allows reuse of AutoYaST-style storage definitions by expressing them in JSON. It accepts an array of opaque objects directly representing the legacy partitioning structure, allowing migration or backward compatibility for existing storage configurations.

EXAMPLE 14: SAMPLE `legacyAutoyastStorage` CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"legacyAutoyastStorage": [
  {
    "partitions": {
      "partition": [
        {
          "device": "/dev/sda1",
          "mount": "/",
          "size": "20G",
          "filesystem": "ext4"
        },
        {
```

```
          "device": "/dev/sda2",
          "mount": "swap",
          "size": "4G",
          "filesystem": "swap"
        }
      ]
    }
  }
]
```

This section contains the following:

legacyAutoyastStorage

An array of JSON objects compatible with the XML structure used in AutoYaST's `partitioning` section. This allows experienced administrators to reuse complex partitioning logic without switching to Agama-native storage syntax.

3.14 iSCSI configuration for an Agama installation profile

The `iscsi` section defines parameters required for configuring iSCSI targets that should be discovered and mounted during system installation. This is particularly relevant for systems that boot from SAN or use iSCSI-based storage volumes.

EXAMPLE 15: SAMPLE `iSCSI` CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"iscsi": {
  "initiatorName": "IQN-OF-INITIATOR",
  "targets": [
    {
      "address": "TARGET-IP",
      "port": 3260,
      "target": "IQN-OF-TARGET",
      "user": "CHAP-USERNAME",
      "password": "CHAP-PASSWORD",
      "autoLogin": true
    }
  ]
}
```

This section contains the following configuration keys:

initiatorName

The iSCSI initiator name (IQN) of the client system. Must follow the iSCSI naming convention. For example, `iqn.2025-05.com.suse:agama-client`.

targets

An array of target definitions to connect to. Each target supports the following fields:

- `address`: IP address or host name of the iSCSI target.
- `port`: TCP port of the target. Default is `3260`.
- `target`: IQN of the iSCSI target to connect to.
- `user` and `password`: CHAP authentication credentials, if required by the target.
- `autoLogin`: Boolean flag indicating whether to automatically log in to the target during boot.

3.15 DASD devices configuration for an Agama installation profile

The `dasd` section is used to activate, configure, or format Direct Access Storage Device (DASD) volumes on IBM Z (s390x) systems. It is relevant only when installing SUSE on mainframe hardware.

EXAMPLE 16: SAMPLE `dasd` DEVICES CONFIGURATION FOR AN AGAMA INSTALLATION PROFILE

```
"dasd": {
  "devices": [
    {
      "channel": "0.0.1234",
      "state": "active",
      "format": true,
      "diag": false
    },
    {
      "channel": "0.0.5678",
      "state": "offline"
    }
  ]
}
```

This section defines the following keys:

devices

A list of DASD devices to configure. Each device is represented as an object with the following fields:

- `channel`: Required. The device channel path, typically in the format `0.0.xxxx`.
- `state`: Optional. Indicates whether the device should be made `active` or put `offline`. Defaults to `active`.

- `format`: Optional. Boolean indicating whether the device should be formatted. If unspecified, formatting happens only if necessary.
- `diag`: Optional. Boolean indicating whether the device should have its diagnostic (diag) flag set. If unspecified, the existing state is preserved.

4 Using AutoYaST profiles with Agama

Agama introduces a modern, declarative installation framework that diverges significantly from the legacy AutoYaST system, even though both aim to automate SUSE Linux Enterprise Server deployments. While partial reuse of existing AutoYaST profiles is possible, direct compatibility is limited due to schema differences, semantic mismatches, and architectural shifts. This topic outlines how to load AutoYaST profiles in Agama, identifies supported modules, and offers practical guidance for converting legacy profiles using recommended tools and conventions.

4.1 Benefits of using AutoYaST profiles in Agama

Reusing existing AutoYaST profiles in Agama provides a pragmatic starting point for teams migrating to the new installer without discarding prior investments. Although direct compatibility is limited, leveraging AutoYaST profiles accelerates transition efforts by retaining core configuration logic, organizational conventions, and validated deployment workflows.

Using AutoYaST profiles in Agama has the following benefits:

Reduced duplication of effort

Existing infrastructure-as-code assets can inform Agama profile structure, minimizing rework.

Faster onboarding

Administrators familiar with AutoYaST can map known modules to Agama fields incrementally.

Incremental migration

Supported AutoYaST elements can be reused while unsupported ones are refactored or omitted over time.

Validation of system assumptions

Reviewing legacy profiles helps surface deprecated patterns and adapt them to Agama's declarative model.

4.2 Limitations of using AutoYaST profiles in Agama

While reusing AutoYaST profiles in Agama may provide a head start during migration, it also introduces significant limitations. The fundamental differences in schema structure, execution model, and configuration philosophy mean that AutoYaST-based profiles can constrain the effectiveness and clarity of Agama workflows if carried over directly.

Using AutoYaST profiles in Agama has the following limitations:

Procedural bias

AutoYaST profiles often rely on execution order, embedded scripts, and imperative constructs, which have no counterpart in Agama's declarative design.

Semantic mismatch

Many AutoYaST modules encapsulate behavior or assumptions not explicitly modeled in Agama, leading to subtle incompatibilities or misconfigurations during reuse.

Reduced transparency

Profiles imported from AutoYaST tend to obscure the declarative simplicity of Agama, making troubleshooting and peer review harder.

Missed modernization opportunities

Clinging to legacy profiles may prevent users from fully adopting Agama's modular, readable, and cloud-native configuration style.

4.3 Loading AutoYaST profiles with Agama

Agama supports loading AutoYaST profiles as part of its transitional support for legacy automation systems. This allows administrators to reuse existing configuration assets while gradually migrating to the native Agama profile format. Several loading mechanisms are available depending on the deployment context and profile structure.

PROCEDURE 1: LOADING AN AUTOYAST PROFILE USING AGAMA

Use the following steps to load an AutoYaST profile in Agama. Profiles can be supplied either through kernel boot parameters or imported using the Agama CLI.

1. Select a method for providing the AutoYaST profile to Agama:

   - Load the profile using a kernel boot parameter. Add the `inst.auto` parameter to the kernel command line and specify the URL of the AutoYaST profile:

     ```
     > sudo linux inst.auto=http://EXAMPLE.NET/AGAMA/SLES.xml
     ```

     This method is commonly used in PXE boot setups or custom ISO builds.

   - Import the profile using the Agama CLI. Run the following command to fetch and preprocess the AutoYaST profile:

     ```
     > sudo agama profile import URL
     ```

     Supported formats include:

     - Agama profiles: `.json`, `.jsonnet`, `.sh`
     - AutoYaST profiles: `.xml`, `.erb`, and directories such as `rules/` or `classes/`

     For more information on supported URL types, refer to https://agama-project.github.io/docs/user/urls.
     When importing AutoYaST content, the CLI automatically evaluates dynamic features such as:

     - Rules and classes for conditional profile selection
     - Embedded Ruby (ERB) for template-based profile generation
     - Pre-installation scripts to dynamically modify profile content

2. Display the loaded or imported profile, or pipe it to a JSON file:

   ```
   > sudo agama config show > profile.json
   ```

4.4 Best practices for converting AutoYaST profiles to Agama profiles

Converting AutoYaST profiles to Agama profiles involves transforming the original XML into Agama's JSON or Jsonnet format. This procedure outlines the recommended steps using the Agama CLI.

PROCEDURE 2: BEST PRACTICES FOR CONVERTING AUTOYAST PROFILES TO AGAMA PROFILES

1. Convert the AutoYaST profile to a JSON file by piping the CLI output to a destination file:

```
> sudo agama profile autoyast http://EXAMPLE.NET/AUTOYAST.xml > profile.json
```

   This command fetches and processes the AutoYaST profile, then writes the resulting Agama-compatible JSON to the specified file.

2. Validate the converted profile:

```
> sudo agama profile validate profile.json
```

   This ensures schema compliance and helps identify any unsupported or misconverted fields.

3. Manually rework or remove unsupported sections, using Agama's schema documentation as a reference.

4. If you require dynamic behavior, convert the profile to Jsonnet. You can then evaluate it to JSON:

```
> sudo agama profile evaluate profile.jsonnet > profile.json
```

5. Test the final profile by loading it into an Agama installation session:

```
> sudo agama config load profile.json
```

6. Make final edits to the loaded profile before starting installation:

```
> sudo agama config edit
```

5 Initiating automated installation using Agama

This topic guides you through starting an automated installation using Agama. You can initiate the process either by specifying boot parameters or by using the Agama command-line interface.

5.1 Initiating the unattended installation

PROCEDURE 3: INITIATING THE UNATTENDED INSTALLATION

1. Start the installation using kernel boot parameters.

Add the `inst.auto` parameter to the kernel command line to specify the location of the Agama profile:

```
> sudo inst.auto=http://example.net/profile.json
```

This method is suitable for PXE boot setups, custom ISO builds, or cloud-init workflows.

2. If you want to access the Agama CLI later, you need to log in as root using a text console. For that purpose, set a password using the `live.password=PASSWORD` boot parameter.

3. Log in as root through a text console, using the password set earlier by editing the boot parameters. If using a software such as the Virtual Machine Manager, use the *Send Keys* menu to send the signal `Ctrl` – `Alt` – `F1` for a text console. To return to a graphical console, use `Ctrl` – `Alt` – `F2` or `Ctrl` – `Alt` – `F7`.
   After logging in as root using a text console, you will have the Agama CLI available.

4. Start the installation using the Agama CLI.

   a. Load the profile with the following command:

   ```
   # agama config load profile.json
   ```

   b. Edit the profile configuration with the following command:

   ```
   # agama config edit profile.json
   ```

   c. Validate the profile:

   ```
   # agama config validate profile.json
   ```

   d. Initiate the installation with the following command:

   ```
   # agama install
   ```

   e. Monitor the installation with the following command:

   ```
   # agama monitor
   ```

   f. Finish the installation with the following command:

   ```
   # agama finish
   ```

6 Activating multipath during installation of SUSE Linux Enterprise using Agama

This topic guides you to activate multipath for storage devices while installing SUSE Linux Enterprise using Agama. You can force activation by modifying boot parameters, or conditionally activate it by modifying Agama profiles.

6.1 Understanding multipath activation during installation using Agama

Multipath in Linux is a device mapper framework that provides redundancy and improved performance for storage devices by creating a single logical device from multiple physical paths to the same storage target. This architecture prevents storage I/O interruptions due to hardware failures while enabling load balancing across paths, and is commonly used in enterprise environments.

To enable this capability, the multipath subsystem must be activated by Agama during installation of SUSE Linux Enterprise. Once activated, multipath affects all devices system-wide, meaning all disks must be referenced exclusively by their corresponding multipath device names. Agama currently provides several mechanisms for users to activate multipath support.

6.2 Requirements

- The Agama installer, which is available with the `.iso` file you use for installing SUSE Linux Enterprise.
- For *forced* activation of multipath, access to command line boot parameters.
- For *conditional* activation of multipath, access to the Agama web interface, or the Agama CLI, or Agama profiles.

By default, you should have access to all the different ways of forced and conditional activation of multipath.

6.3 Activating multipath using boot parameters

If you are certain that the system uses multipath, the most reliable method to ensure that Agama activates the corresponding subsystems is to enable the environment variable `LIBSTORAGE_MULTIPATH_AUTOSTART` at the time of boot.

To enable multipath during installation, perform the following steps:

1. When the boot menu is displayed, select `e` to edit the boot parameters.
2. Add the `LIBSTORAGE_MULTIPATH_AUTOSTART` parameter and set it to `1`.

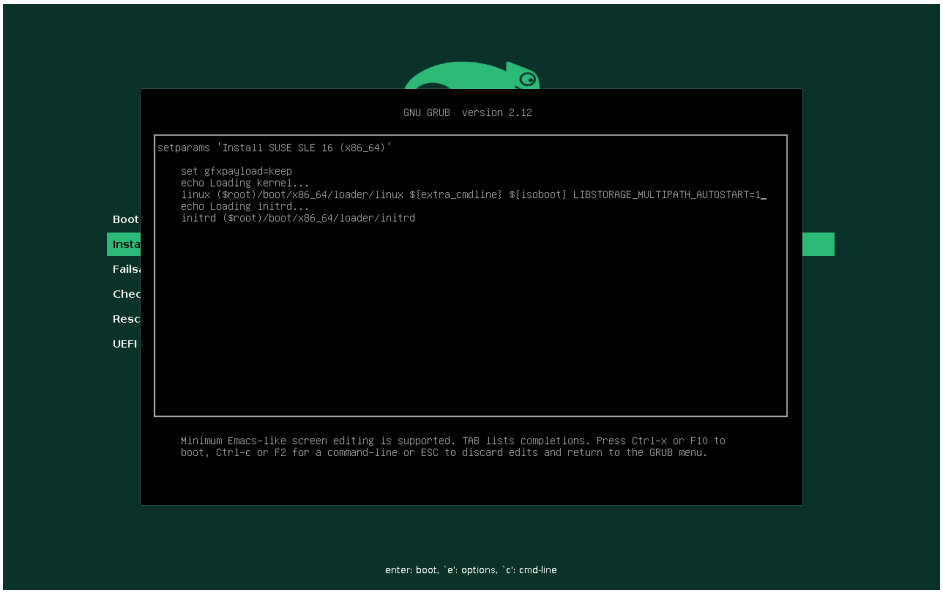

FIGURE 2: ACTIVATION OF MULTIPATH USING BOOT PARAMETERS

3. Select `Ctrl` – `x` or `F10` to exit the edit window and continue with the boot.

6.4 Activating multipath using Agama web interface, CLI, or profiles

If multipath activation is not enforced through the boot parameter, Agama attempts to detect multipath devices in the system. When detected, Agama either prompts you to confirm multipath activation, or proceeds with unattended installation based on the answer provided in the Agama profile.

Warning: Unreliable multipath detection

Detection of multipath devices using Agama is not fully reliable. If you are sure that the system uses multipath devices, we recommend using the method of enabling the necessary boot parameters.

Depending on whether you are performing an interactive or an unattended installation, perform the necessary steps as described below.

- In an **interactive** installation, Agama prompts you to confirm whether to activate multipath. The prompt can be answered interactively using the web interface, or in the command-line interface using the `agama questions` command.
  - If using the web interface, confirm when you see a prompt similar to the following:

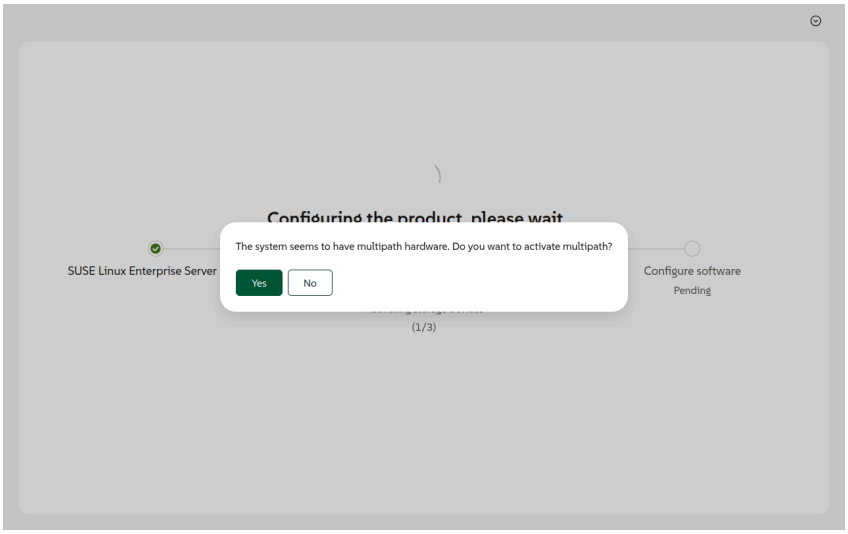

FIGURE 3: ACTIVATION OF MULTIPATH USING THE AGAMA WEB INTERFACE

  - If using the Agama CLI, perform the following:

    a. Edit the Agama profile to include the following JSON configuration:

```
{
  "questions": {
    "answers": [
      {
        "class": "storage.activate_multipath",
        "answer": "yes"
      }
    ]
  }
```

```
}
```

b. Provide the path to answers of Agama questions using the following command:

```
# agama questions answers PATH/TO/JSON/FILE
```

- In an **unattended** installation, Agama looks for an answer to the question about multipath activation in the Agama profile.

a. Edit the Agama profile to include the following JSON configuration:

```
{
  "questions": {
    "answers": [
      {
        "class": "storage.activate_multipath",
        "answer": "yes"
      }
    ]
  }
}
```

b. Provide the path to the Agama profile using the `inst.auto=URL/PATH/TO/AGAMA/JSON/PROFILE` boot parameter.

6.5 Summary of multipath activation

If you are performing a manual installation of SUSE Linux Enterprise using Agama, you can activate multipath support by adding the `LIBSTORAGE_MULTIPATH_AUTOSTART=1` boot parameter, or confirming multipath activation using the web or command-line interfaces of Agama during the installation process.

If you are performing an unattended installation using Agama, edit the Agama profile to include answer to the question of multipath activation, and pass the path to the profile as the value of the `inst.auto` boot parameter.

6.6 Troubleshooting multipath activation

Depending on the chosen method of multipath activation, use one of more of the following troubleshooting tips:

- If using forced activation, ensure that you correctly add the `LIBSTORAGE_MULTIPATH_AUTOSTART=1` boot parameter. After the boot, you can verify it in the content of the GRUB 2 configuration file of your system.
- If using answers to Agama questions, ensure that you have used the correct JSON snippet for multipath activation, by running the **`agama config show`** command.

7 Advanced storage configuration using Agama profiles

Storage configuration in Agama is one of the most powerful and flexible components of the automated installation process. It allows you to declaratively define everything from simple partition layouts to sophisticated combinations—all before the system is booted for the first time.

This section illustrates several common use cases with examples. For an exhaustive reference of all elements, refer to the schema https://github.com/agama-project/agama/blob/master/rust/agama-lib/share/storage.schema.json.

7.1 Basic structure of the `storage` schema

A `storage` section contains several entries describing how to configure the corresponding storage devices, and some extra entries such as `boot` to setup some general aspects that influence the final layout.

Each volume group or software RAID can represent a new logical device to be created, or an existing device from the system to be processed. Entries below `drives` represent devices that can be used as regular disks, which includes removable and fixed disks, SD cards, DASD or zFCP devices, iSCSI disks, and multipath devices. Those entries always correspond to devices that can be found at the system, because Agama cannot create that kind of devices.

EXAMPLE 17: TOP-LEVEL `storage` SECTION SYNTAX

```
"storage": {
```

```
  "drives": [ ... ],
  "volumeGroups": [ ... ],
  "mdRaids": [ ... ],
  "boot": { ... }
}
```

Note: Compatibility with legacy AutoYaST storage

In some cases, `storage` can be replaced by the `legacyAutoyastStorage` section. This section supports everything offered by the `partitioning` section of the AutoYaST profile. However, Agama does not validate this special section—be careful to provide valid AutoYaST options.

The following sections illustrate certain common use-cases with examples. Before execution, it is recommended to validate the configurations using the tools provided by Agama.

7.2 Describing the devices

The `drives` collection contains several optional fields, some of which are mutually exclusive.

EXAMPLE 18: ELEMENTS IN THE `drive` COLLECTION

```
{
  "alias": "...",
  "search": { ... },
  "encryption": { ... },
  "filesystem": { ... },
  "partitions": [ ... ],
  "ptableType": "..."
}
```

Usually, a device represented by a `drive` entry is divided into several partitions. Each entry of `partitions` has the following structure with several optional fields:

EXAMPLE 19: STRUCTURE OF `partitions`

```
{
  "alias": "...",
  "search": { ... },
  "id": "...",
  "size": { ... },
```

```
  "encryption": { ... },
  "filesystem": { ... },
  "delete": ...,
  "deleteIfNeeded": ...
}
```

Drives and partitions can be combined such that the one disk is used to create partitions and the other is directly formatted.

EXAMPLE 20: COMBINATION OF DISKS WITH PARTITIONS AND DIRECT FORMATTING

```
"storage": {
  "drives": [
    {
      "partitions": [
        {
          "filesystem": { "path": "/" },
          "size": { "min": "10 GiB" }
        },
        {
          "filesystem": { "path": "swap" },
          "size": "2 GiB"
        }
      ]
    },
    {
      "filesystem": { "path": "/home" },
    }
  ]
}
```

An entry from `volumeGroups` can have the following properties:

EXAMPLE 21: STRUCTURE OF `volumeGroups`

```
{
  "alias": "...",
  "name": "...",
  "search": { ... },
  "physicalVolumes": [ ... ],
  "logicalVolumes": [ ... ],
  "peSize": ... ,
  "delete": ...
}
```

Entries of `logicalVolumes` are relatively similar to the ones used to describe partitions.

EXAMPLE 22: STRUCTURE OF `logicalVolumes`

```
{
  "alias": "...",
  "search": { ... },
  "name": "...",
  "size": { ... },
  "encryption": { ... },
  "filesystem": { ... },
  "pool": ...,
  "usedPool": "...",
  "stripes": ...,
  "stripeSize": ...,
  "delete": ...,
  "deleteIfNeeded": ...
}
```

To understand how all the previously described elements fit together, consider the following example in which the first disk of the system is partitioned and a volume group is created on top of that partition after encryption, to allocate two file systems.

EXAMPLE 23: A COMBINATION OF DISK PARTITION, VOLUME GROUPS AND FILE SYSTEMS

```
"storage": {
  "drives": [
    {
      "partitions": [
        {
          "alias": "pv",
          "id": "lvm",
          "size": { "min": "12 GiB" },
          "encryption": {
            "luks2": { "password": "my secret passphrase" }
          }
        }
      ]
    }
  ],
  "volumeGroups": [
    {
      "name": "system",
      "physicalVolumes": [ "pv" ],
      "logicalVolumes": [
        {
```

```
          "size":   { "min": "10 GiB" },
          "filesystem": { "path": "/", "type": "btrfs" }
        },
        {
          "size":   "2 GiB",
          "filesystem": { "path": "swap", "type": "swap" }
        }
      ]
    }
  ]
}
```

Agama can also manage software-defined MD RAID arrays represented as entries at the `mdRaids` collection.

EXAMPLE 24: STRUCTURE OF `mdRaids` COLLECTION

```
{
  "alias": "...",
  "name": "...",
  "search": { ... },
  "level": "...",
  "chunkSize": ... ,
  "devices": [ ... ],
  "size": { ... },
  "encryption": { ... },
  "filesystem": { ... },
  "partitions": [ ... ],
  "ptableType": "...",
  "delete": ...
}
```

The `devices` property is used to specify the devices that act as members of the RAID.

EXAMPLE 25: AN EXAMPLE WITH `mdRaids` AND `devices`

```
"storage": {
  "drives": [
    {
      "search": "/dev/sda",
      "partitions": [
        { "alias": "sda-40", "size": "40 GiB" }
      ]
    },
    {
```

```
      "search": "/dev/sdb",
      "partitions": [
        { "alias": "sdb-40", "size": "40 GiB" }
      ]
    }
  ],
  "mdRaids": [
    {
      "devices": [ "sda-40", "sdb-40" ],
      "level": "raid0"
    }
  ]
}
```

7.3 Searching existing devices

When a section in the profile describes modification and deletion of devices, the description must match with one or more devices from the system. If a description matches several devices, the same operations are applied to all of them. This approach is useful in several situations, such as applying the same partitioning schema to several disks or deleting all partitions of a disk that match a given criteria.

Matching is performed using a `search` subsection, as illustrated below. By default, all devices in the scope fitting the conditions are matched. You can limit the number of devices that match using `max`. The following example shows how you can use several search sections to find the three biggest disks in the system, delete all partitions bigger than 1 GiB within them and create new partitions of type RAID.

EXAMPLE 26: USAGE OF THE `search` SECTION

```
"storage": {
  "drives": [
    {
      "search": {
        "sort": { "size": "desc" },
        "max": 3
      },
      "partitions": [
        {
          "search": {
            "condition": { "size": { "greater": "1 GiB" } }
          },
          "delete": true
        },
        {
          "alias": "newRaidPart",
```

```
          "id": "raid",
          "size": { "min": "1 GiB" }
        }
      ]
    }
  ]
}
```

The scope of each search depends on the place in the profile of the `search` section. In the example, the devices from the system that are considered as possible candidates. A search section inside the description of an MD RAID only matches the software RAID devices. A search section inside the `partitions` subsection of that RAID description only matches the partitions of RAIDs that have matched the conditions of the most external search.

A device can never match two different sections of the Agama profile. When several sections at the same level contain a search subsection, devices are matched in the order the sections appear on the profile.

EXAMPLE 27: ORDER OF DEVICE MATCHING

```
"storage": {
  "drives": [
    {
      "search": {
        "sort": { "size": "desc" },
        "max": 1
      },
      "alias": "biggest"
    },
    {
      "search": {
        "sort": { "size": "desc" },
        "max": 1
      },
      "alias": "secondBiggest"
    }
  ]
}
```

An empty search matches all devices in the scope. For example, the configuration given below deletes all the partitions of the chosen disk, but only if the disk contains partitions.

EXAMPLE 28: EMPTY SEARCH MATCHING ALL DEVICES IN THE SCOPE

```
"storage": {
```

```
  "drives": [
    {
      "partitions": [
        { "search": {}, "delete": true }
      ]
    }
  ]
}
```

If there is not a single system device matching the scope and the conditions of a given search, then `ifNotFound` is used. If the value is `skip`, the device definition is ignored. If the value is `"error"` the whole process is aborted.

Entries on drives are different from all other subsections describing devices because drives can only be matched to existing devices. If search is omitted for a drive, it is considered to contain the following configuration:

EXAMPLE 29: BEHAVIOR OF `search` IF OMITTED FOR A DRIVE

```
{
  "search": {
    "sort": "name",
    "max": 1,
    "ifNotFound": "error"
  }
}
```

When the syntax of a search subsection becomes cumbersome, you can use simple strings.

You can use search to find a device by its name. For example:

EXAMPLE 30: SEARCHING USING DEVICE NAME

```
{ "search": "/dev/sda" }
```

The string `*` allows to match all the devices from the current context, if any. This is specially useful to match all partitions or logical volumes in a device, irrespective of whether there is any. For example, the two following search sections are equivalent:

EXAMPLE 31: EXAMPLE OF EQUIVALENT SEARCH

```
{ "search": "*" }
```

```
{ "search": { "ifNotFound": "skip" } }
```

7.4 Referencing other devices

At certain times, it is necessary to reference other devices as part of the specification of an LVM volume group or RAID. Those devices can be existing system devices, or devices that will be created as response to another entry of the Agama profile. For that purpose, you can use `alias`.

EXAMPLE 32: USING `alias` IN STORAGE CONFIGURATION

```
"storage": {
  "drives": [
    {
      "partitions": [
        { "size": "50 GiB", "id": "lvm", "alias": "newPV" }
      ]
    }
   ],
   "volumeGroups": [
    {
      "name": "newVG",
      "physicalVolumes": [ "newPV" ],
      "logicalVolumes": [ { "name": "data", "size": "20 GiB" } ]
    }
  ]
}
```

If a section matching several existing devices contains an alias, that alias is considered as a reference to all the devices. Consider the following equivalent examples that assumes there are at least two disks in the system:

EXAMPLE 33: EQUIVALENT EXAMPLES OF ALIAS

```
"storage": {
  "drives": [
    {
      "search": {
        "sort": { "size": "desc" },
        "max": 1,
      },
      "alias": "biggest"
    },
    {
      "search": {
```

```
        "sort": { "size": "desc" },
        "max": 1,
      },
      "alias": "secondBiggest"
    }
  ],
  "mdRaids": [
    {
      "devices": [ "biggest", "secondBiggest" ],
      "level": "raid0"
    }
  ]
}

"storage": {
  "drives": [
    {
      "search": {
        "sort": { "size": "desc" },
        "max": 2,
      },
      "alias": "big"
    }
  ],
  "mdRaids": [
    {
      "devices": [ "big" ],
      "level": "raid0"
    }
  ]
}
```

7.5 Specifying the size of a device

When configuring storage in the Agama profile, you must specify the desired size for a new device or the target size when resizing an existing one. The schema allows for flexible size specification. The most common method is using a human-readable string that can be parsed into a valid size. For example, `10 GiB`. Alternatively, you can provide a size as an array (a tuple) containing a minimum size and an optional maximum size. The resulting size will be between these two thresholds. If the maximum is omitted, the device will expand to consume all available contiguous space, respecting other specified size constraints.

For configurations targeting existing partitions or Logical Volumes (LVs)—which must include a search section—the special keyword `current` can be used as a minimum or maximum size limit. The use of `current` and how it affects resizing the corresponding devices is explained separately.

If the size property is completely omitted for an existing device (for example, combined with search), Agama acts as if both minimum and maximum limits were set to `current`. This characteristic implies that the partition or logical volume are not be resized. If the size is omitted for a device that will be created but includes a file system entry specifying a mount point, Agama can determine the size limits by applying the settings of the installation product. In Agama terminology, the product is the operating system being installed, and it specifies the default size ranges for its relevant file systems, such as `/`, `swap` and `/home`.

7.6 Partition needed for booting

You can use the `boot` entry to configure whether Agama should calculate and create extra partitions needed for booting. The behavior is same when using an alias. If the device is not specified, Agama takes the location of the root file system as reference.

EXAMPLE 34: PARTITIONS NEEDED FOR BOOTING

```
"storage": {
  "drives": [
    {
      "search": "/dev/sda",
      "alias": "bootDisk"
    },
    {
      "search": "/dev/sdb",
      "partitions": [
        { "filesystem": { "path": "/" } }
      ]
    }
  ],
  "boot": {
    "configure": true,
    "device": "bootDisk"
  }
}
```

7.7 Keeping an existing file system or encryption layer

The entries for both `filesystem` and `encryption` contains a `reuse` flag with a default value of `false`. You can use it in combination with `search` to specify that the device must not be re-formatted or re-encrypted.

7.8 Deleting and shrinking existing devices

The storage configuration proposal must allow defining how to manage existing storage components, including partitions, LVM logical volumes, MD RAIDs, and LVM volume groups. A search mechanism is employed to match a partition or LVM logical volume definition with one or more devices already present on the system. Once a match is made, you can specify the required action.

The component can be marked to be deleted unconditionally, or deleted if needed to free space for newly defined devices. It can also be shrunk to a necessary size or shrunk or extended to a specific size or range. It is even possible to express combined actions, like attempting to shrink a component first and only proceeding to delete it if shrinking doesn't free up enough space.

Deletion is achieved with the corresponding `delete` flag for unconditional deletion or the `deleteIfNeeded` flag for conditional deletion. If either of these flags is active for a partition, it is illogical to specify any other usage for it, such as declaring a file system. For example, you can configure the proposal to unconditionally delete partition number `1` and then conditionally delete other partitions as needed to secure the space for a new `30 Gib` partition.

EXAMPLE 35: DELETING AND SHRINKING EXISTING DEVICES

```
"storage": {
  "drives": [
    {
      "partitions": [
        {
          "search": {
            "condition": { "number": 1 }
          },
          "delete": true
        },
        { "search": {}, "deleteIfNeeded": true },
        { "size": "30 GiB" }
      ]
    }
  ]
}
```

Often some partitions or logical volumes are shrunk only to make space for the declared devices. But because resizing is not a destructive operation, you can declare a given partition to be resized (shrunk or extended), then formatted and/or mounted.

Note: Limitation of resizing

Resizing a partition can be limited depending on its content and the file system type.

Combining `search` and `resize` is enough to indicate that Agama is expected to resize a given partition, if possible. The keyword `current` can be used as `min` and/or `max` for the size range, and it is always equivalent to the exact original size of the device. The simplest way to use `current` is to just specify that the matched device should keep its original size. That is the default for searched (and found) devices, if size is completely omitted.

```
"storage": {
  "drives": [
    {
      "partitions": [
        {
          "search": {
            "condition": { "number": 1 }
          },
          "size": { "min": "current", "max": "current" }
        }
      ]
    }
  ]
}
```

You can use other combinations to specify how a device could be resized, if possible. See the following examples with explanatory file system labels.

Warning

The condition `fsLabel` is not yet implemented.

EXAMPLE 36: DELETING AND SHRINKING OF EXISTING DEVICES INCLUDING THE `fsLabel` CONDITION FOR FILE SYSTEM LABELS

```
"storage": {
  "drives": [
    {
      "partitions": [
        {
          "search": {
            "condition": { "fsLabel": "shrinkIfNeeded" }
          },
          "size": { "min": 0, "max": "current" }
        },
        {
          "search": {
            "condition": { "fsLabel": "resizeToFixedSize" }
          },
```

```
          "size": "15 GiB"
        },
        {
          "search": {
            "condition": { "fsLabel": "resizeByRange" }
          },
          "size": { "min": "10 GiB", "max": "50 GiB" }
        },
        {
          "search": {
            "condition": { "fsLabel": "growAsMuchAsPossible" }
          },
          "size": { "min": "current" }
        },
      ]
    }
  ]
}
```

When the size limits are specified as a combination of `current` and a fixed value, ensure that the resulting `min` is not bigger than the resulting `max`.

Both `deleteIfNeeded` and a size range can be combined to indicate that Agama should make space first, by shrinking the partitions and deleting them only if shrinking is not enough.

```
"storage": {
  "drives": [
    {
      "partitions": [
        {
          "search": {},
          "size": { "min": 0, "max": "current" },
          "deleteIfNeeded": true
        }
      ]
    }
  ]
}
```

7.9 Generating default volumes

Every product provides a configuration which defines the storage volumes. For example, feasible file systems for root and default partitions to create. The default or mandatory product volumes can be automatically generated by using a `generate` section in the `partitions` or `logicalVolumes` sections.

```
"storage": {
  "drives": [
    {
      "partitions": [
        { "generate": "default" }
      ]
    }
  ]
}
```

The `generate` section allows creating the product volumes without explicitly writing all of them. The configuration above would be equivalent to the following:

```
"storage": {
  "drives": [
    {
      "partitions": [
        { "filesystem": { "path": "/" } },
        { "filesystem": { "path": "/home" } },
        { "filesystem": { "path": "swap" } }
      ]
    }
  ]
}
```

If any path is explicitly defined, the `generate` section will not generate a volume for it. For example, with the following configuration, only root and swap would be automatically added.

```
"storage": {
  "drives": [
    {
      "partitions": [
        { "generate": "default" },
        { "filesystem": { "path": "/home" } }
      ]
    }
  ]
}
```

The auto-generated volumes can be also configured. For example, for encrypting the partitions:

```
"storage": {
  "drives": [
    {
      "partitions": [
        {
```

```
          "generate": {
            "partitions": "default",
            "encryption": {
              "luks1": { "password": "12345" }
            }
          }
        }
      ]
    }
  ]
}
```

The `mandatory` keyword can be used for generating only the mandatory partitions or logical volumes:

```
"storage": {
  "volumeGroups": [
    {
      "name": "system",
      "logicalVolumes": [
        { "generate": "mandatory" }
      ]
    }
  ]
}
```

7.10 Generating physical volumes

You can configure volume groups to explicitly use a set of devices as physical volumes. The aliases of the devices to use are added to the list of physical volumes:

```
"storage": {
  "drives": [
    {
      "search": "/dev/vda",
      "partitions": [
        { "alias": "pv2", "size": "100 GiB" },
        { "alias": "pv1", "size": "20 GiB" }
      ]
    }
  ],
  "volumeGroups": [
    {
      "name": "system",
      "physicalVolumes": ["pv1", "pv2"]
    }
```

```
  ]
}
```

The physical volumes can be automatically generated too, by simply indicating the target devices in which to create the partitions. For that, a `generate` section is added to the list of physical volumes:

```
"storage": {
  "drives": [
    {
      "search": "/dev/vda",
      "alias": "pvs-disk"
    }
  ],
  "volumeGroups": [
    {
      "name": "system",
      "physicalVolumes": [
        { "generate": ["pvs-disk"] }
      ]
    }
  ]
}
```

If the auto-generated physical volumes have to be encrypted, then the encryption config is added to the generate section:

```
"storage": {
  "drives": [
    {
      "search": "/dev/vda",
      "alias": "pvs-disk"
    }
  ],
  "volumeGroups": [
    {
      "name": "system",
      "physicalVolumes": [
        {
          "generate": {
            "targetDevices": ["pvs-disk"],
            "encryption": {
              "luks2": { "password": "12345" }
            }
          }
        }
      ]
    }
```

```
  ]
}
```

8 Compatibility between AutoYaST and Agama profiles

AutoYaST has long been the standard for unattended and automated installations in SUSE Linux Enterprise Server systems. With the advent of the Agama installer, a new approach to system configuration and deployment has emerged—designed to be modular, declarative and extensible using modern formats and APIs.

This section provides a detailed comparative view of the configuration models in AutoYaST and Agama, highlighting conceptual differences and offering practical guidance for transitioning to the Agama profile format. The goal is to equip experienced AutoYaST users with a clear roadmap for migrating existing profiles to the new Agama schema.

Where applicable, compatibility matrices are provided to indicate which AutoYaST modules and fields are currently supported, planned, undecided, or explicitly unsupported in Agama. These mappings are based on the upstream reference maintained by the Agama project.

8.1 Conceptual differences

This table highlights the fundamental differences in design philosophy and approach between AutoYaST and Agama.

TABLE 1: AUTOYAST VS AGAMA DESIGN COMPARISON

AutoYaST	Agama
XML-based, verbose syntax	YAML/JSON-based, declarative syntax
Feature-rich and legacy-compatible	Minimalist and cloud-native
Granular configuration of every detail	Relies on sane defaults and abstraction
Imperative and monolithic structure	Composable and modular design
Tightly coupled with YaST modules	Engineered independently with API support

8.2 Mapping AutoYaST sections to Agama schema

This section provides a detailed comparison and translation map between the major sections and modules of AutoYaST and their equivalents (or lack thereof) in the Agama profile schema. Each subsection addresses a particular functional area, indicating how configuration responsibilities are split or restructured in Agama, and clearly states where support is partial, planned, or unavailable.

Note: Granular support status for AutoYaST elements

For a more granular information on the compatibility and support status for AutoYaST elements in Agama profiles as compared to what is presented here, refer to the upstream documentation https://agama-project.github.io/docs/user/autoyast/reference.

8.2.1 System identity and localization

This section covers the basic configuration for setting the system's host name, language, keyboard layout, time zone, and the installed product identity. These are foundational parameters during the installation and are typically mapped one-to-one between AutoYaST and Agama.

TABLE 2: SYSTEM IDENTITY AND LOCALIZATION MAPPING

AutoYaST element	Agama field	Support status	Comment
hostname	hostname.static / hostname.transient	Fully supported	Agama distinguishes static and transient host names.
language	localization.language	Fully supported	Uses standard locale codes (for example, `en_US.UTF-8`).
keyboard	localization.keyboard	Fully supported	Set using layout ID (for example, `us`, `de`).
timezone	localization.timezone	Fully supported	Time zone IDs follow the standard time zone database names (for example, `Europe/Berlin`).

AutoYaST element	Agama field	Support status	Comment
product / base	product.id	Fully supported	Matches product identifiers from the system's installed products metadata.
product / register	product.registrationCode, product.registrationEmail, product.registrationUrl	Fully supported	Used for SUSEConnect-based registration. Add-ons can be declared under `product.addons`.

8.2.2 User management and authentication

TABLE 3: USER AND AUTHENTICATION MAPPINGS

AutoYaST element	Support status	Agama field	Comment
/root/password	Fully supported	root.password	Can be plain text or hashed using supported schemes
/root/hashed_password	Fully supported	root.hashedPassword = true	Indicates password is hashed
/root/ssh_authorized_keys	Fully supported	root.sshPublicKey	Single key string; no support for multiple entries
/users/user	Fully supported	user	Only one non-root user supported directly
/users/user/encrypted	Fully supported	user.hashedPassword	Same semantics as root
/users/user/password	Fully supported	user.password	Plain or hashed password
/users/user/username	Fully supported	user.userName	Login name

AutoYaST element	Support status	Agama field	Comment
/users/user/fullname	Fully supported	user.fullName	Human-readable full name
/users/user/autologin	Fully supported	user.autologin	Primarily used in desktop environments
/users/user/uid, gid, shell, home	Not supported	—	Agama does not currently support these fine-grained user parameters

8.2.3 Network configuration

This section details how network settings are defined in AutoYaST and Agama, covering interface setup, DHCP/static addressing, bonding, bridging, and other advanced networking configurations.

TABLE 4: NETWORK CONFIGURATION MAPPING

AutoYaST element	Agama field	Support status	Comment
network/config	network.interfaces	Fully supported	Supports configuring individual interfaces with static or DHCP settings.
network/dns	network.dns	Fully supported	Includes configuration of nameservers and search domains.
network/routing	network.routes	Fully supported	Static route configuration is supported per interface.
network/hostname	hostname	Fully supported	System host name can be set independently of network block.

AutoYaST element	Agama field	Support status	Comment
network/bridge	network.interfaces[].type: bridge	Fully supported	Bridge devices are supported using dedicated interface types.
network/bonding	network.interfaces[].type: bond	Fully supported	Bonding configuration supports mode, primary interface, and slaves.
network/ieee8021x	network.interfaces[].ieee8021x	Partially supported	Supports basic 802.1x authentication with identity, password, and method. Certificate support is limited.
network/proxy	—	Not supported	No native proxy configuration support in Agama. Should be handled post-install.

8.2.4 Storage and partitioning

This section compares the storage configuration capabilities of AutoYaST and Agama. It covers traditional partitions, logical volumes, file systems, encryption, RAID, and other storage-specific aspects of system setup.

TABLE 5: STORAGE CONFIGURATION MAPPING

AutoYaST element	Agama field	Support status	Comment
partitioning	storage.devices[].partitions	Fully supported	Traditional partitioning with labels, mount points, and formats are fully supported.
filesystems	storage.devices[].partitions[].filesystem	Fully supported	Common file systems such as ext4, XFS, and Btrfs are supported

AutoYaST element	Agama field	Support status	Comment
			with mount and format options.
lvm	storage.devices[].partitions[].lvm	Fully supported	Volume groups and logical volumes are fully supported using declarative syntax.
raid	storage.devices[].partitions[].raid	Fully supported	Software RAID levels (0, 1, 5, etc.) are supported, including metadata and spare settings.
btrfs	storage.devices[].partitions[].btrfs	Fully supported	Subvolumes, compression, and Btrfs-specific mount options are available.
encryption	storage.devices[].partitions[].luks	Fully supported	Supports LUKS encryption with passphrase, key file, and reuse options.
reuse/initialize	storage.devices[].partitions[].reformat / preserve	Fully supported	Reusing existing devices or forcing format is declaratively specified.
bootloader-location	—	Not supported	Installation location of the bootloader is not configurable via storage profile.
partition-id type	storage.devices[].partitions[].type	Fully supported	Allows specifying Linux native, EFI, swap, BIOS boot partitions, etc.

AutoYaST element	Agama field	Support status	Comment
partition flags	storage.devices[].partitions[].flags	Fully supported	Supports marking partitions as bootable, ESP, hidden, etc.
complex criteria (for example, by-id)	storage.devices[].match	Fully supported	Devices can be selected using labels, device paths, UUIDs, or custom match rules.

8.2.5 Software selection and patterns

This section maps how software selection is handled in AutoYaST and Agama, including individual package installation and pattern-based selections.

TABLE 6: SOFTWARE AND PATTERN MAPPING BETWEEN AUTOYAST AND AGAMA

AutoYaST element	Agama field	Support status	Comment
/autoinst/software/patterns	software.patterns	Fully supported	Pattern selection is supported directly via a list of strings.
/autoinst/software/packages	software.packages	Fully supported	Individual package names can be specified as strings.
/autoinst/software/remove-packages	—	Not supported	No mechanism currently exists in Agama to specify packages for removal.
/autoinst/software/do_online_update	—	Not supported	Agama does not support configuring online updates during installation.

8.2.6 Boot loader settings

This section maps bootloader configuration options between AutoYaST and Agama profiles.

TABLE 7: BOOT LOADER CONFIGURATION MAPPING

AutoYaST element	Agama field	Support status	Comment
bootloader—timeout	bootloader.timeout	Fully supported	Sets boot menu timeout before booting default entry.
bootloader—kernel_parameters	bootloader.extraKernelParams	Fully supported	Additional kernel command-line parameters.
bootloader—flag (for example, no_timeout)	bootloader.stopOnBootMenu	Fully supported	Controls whether the bootloader stops at the boot menu.
bootloader—location	—	Not supported	Agama currently does not support choosing GRUB installation location.
bootloader—gfxmode / theme	—	Not supported	Graphical bootloader themes and resolutions are not yet configurable.

8.2.7 Security, certificates, and registration

This section compares how AutoYaST and Agama handle security settings, certificate deployment, and system registration during installation.

TABLE 8: SECURITY AND REGISTRATION ELEMENT MAPPING

AutoYaST element	Agama field	Support status	Comment
security/sshd	security.ssh.enable	Fully supported	Enables or disables the SSH service.
security/certificates	security.trustedCertificates	Fully supported	Supports placement of trusted root certificates

AutoYaST element	Agama field	Support status	Comment
			in `/etc/pki/trust/anchors`.
security/ssh_import_authorized_keys	root.authorizedKeys / user.authorizedKeys	Fully supported	SSH public keys can be configured per user for key-based authentication.
register / suse_register	product.registration	Fully supported	Handles system registration to SUSE Customer Center (SCC) or RMT.
security/selinux	—	Not supported	Agama currently does not offer SELinux configuration; SUSE systems use AppArmor by default.

8.2.8 Pre-install, post-install, and init scripts

This section maps the script execution phases between AutoYaST and Agama profiles.

TABLE 9: SCRIPT PHASES IN AUTOYAST VS. AGAMA

AutoYaST element	Agama field	Support status	Comment
pre-scripts	scripts.pre	Fully supported	Runs before installation begins.
postpartitioning-scripts	scripts.postPartitioning	Fully supported	Executed immediately after disk partitioning.
post-scripts	scripts.post	Fully supported	Runs after installation finishes, with optional chroot control.
init-scripts	scripts.init	Fully supported	Executes on first boot of the target system.

8.2.9 File deployment and customization

This section compares how custom files can be deployed during installation using AutoYaST and Agama profiles.

TABLE 10: COMPARISON OF FILE DEPLOYMENT ELEMENTS

AutoYaST element	Agama field	Support status	Comment
`files`	`files`	Fully supported	Supports deployment of custom files with content, permissions, and ownership.
`sysconfig`	—	Not supported	Environment-specific configuration via sysconfig is not directly handled in Agama.
`etc`	—	Not supported	Configuration drop-ins for `/etc` are not explicitly mapped in Agama.

8.2.10 Miscellaneous hardware-specific sections

This section covers specialized hardware-related configuration elements from AutoYaST and their equivalents (or lack thereof) in Agama.

TABLE 11: HARDWARE-SPECIFIC CONFIGURATION MAPPING

AutoYaST element	Agama field	Support status	Comment
dasd	dasd	Fully supported	Required for IBM Z (s390x) environments. Allows activation and formatting of DASD devices.
iscsi	iscsi	Fully supported	Supports target discovery, authentication, and login configuration for iSCSI volumes.

AutoYaST element	Agama field	Support status	Comment
zipl	—	Not supported	Boot loader configuration on s390x is partially handled by other fields like `bootloader` but no direct equivalent for zipl.
kdump	—	Not supported	Agama does not currently support configuring Kdump crash kernels.
udev	—	Not supported	Custom udev rules are not handled in Agama. Can be added post-install via scripts.

8.3 Unsupported AutoYaST profile elements in Agama

The following table lists AutoYaST profile sections that are currently not supported by Agama. These modules either have no equivalent functionality in Agama, are considered legacy or niche, or are planned for future implementation. This list is essential for users migrating from AutoYaST to avoid misconfiguration or unmet expectations.

TABLE 12: UNSUPPORTED AUTOYAST MODULES IN AGAMA

AutoYaST element	Support status	Comment
audit-laf	Not supported	Not planned; used for audit logging configuration.
auth-client	Not supported	No direct equivalent; should be handled post-install.
clientconfig	Not supported	Custom YaST client settings not exposed in Agama.

AutoYaST element	Support status	Comment
configuration_management	Not supported	No built-in support for Puppet, Chef, SaltStack, etc.
cron	Not supported	Scheduling tasks must be configured after installation.
deploy_image	Not supported	Image deployment not in scope for Agama profiles.
dhcp-server	Not supported	Service configuration is beyond the profile’s scope.
dns-server	Not supported	No direct DNS service setup support in Agama.
fcoe-client	Not supported	FCoE setup must be done manually or via other tools.
firewall	Not supported	Firewall configuration is not managed by Agama.
firstboot	Not supported	No support for post-install first boot workflows.
ftp-server	Not supported	No FTP service configuration support.
general	Not supported	Legacy catch-all section no longer used.
groups	Not supported	Group creation must be done via post-install scripts.
host	Not supported	Deprecated; handled through host name and networking.
http-server	Not supported	No direct Apache/lighttpd setup supported.
kdump	Support planned	Not yet implemented but on roadmap.

AutoYaST element	Support status	Comment
mail	Not supported	No MTA/MUA configuration support.
nfs	Not supported	Client NFS mounts must be configured later.
nfs_server	Not supported	NFS server setup is out of scope.
nis	Not supported	NIS authentication not available.
nis_server	Not supported	No equivalent configuration for NIS server.
ntp-client	Not supported	Time sync must be managed post-install.
printer	Not supported	No CUPS or printer configuration support.
proxy	Not supported	Proxy settings must be applied via scripts or after deployment.
report	Not supported	Install-time reporting is not implemented.
samba-client	Not supported	Samba integration must be manually configured.
sound	Not supported	No sound system setup.
squid	Not supported	Proxy server setup not applicable to installation profiles.
ssh_import	Not supported	SSH key import handled differently in Agama.
sysconfig	Not supported	Low-level sysconfig modifications not available.

AutoYaST element	Support status	Comment
tftp-server	Not supported	Service configurations are expected post-install.
udev	Not supported	Custom udev rules are not supported declaratively.
upgrade	Not supported	In-place upgrades are not part of the installer's job.
usersDefaults	Not supported	No support for user templates or defaults.

9 For more information

For more information on Agama and automated installation, refer to the following resources:

Warning: Use upstream information with caution

The upstream resources listed below may contain code or information not covered under the terms of service by SUSE. Use them with caution only as a reference for clarity and inspiration.

- Agama user documentation (https://agama-project.github.io/docs/user): Organizes information by user perspective and covers a wide range of topics, including interactive installation, unattended installation, AutoYaST support, boot options, URLs, command-line reference, and remote access.
  - Agama boot options (https://agama-project.github.io/docs/user/boot_options): Explains the kernel boot parameters that can be used to influence the Agama installation process, including how to specify the URL for an unattended installation profile using `inst.auto`. It also mentions other useful options like `inst.info`, `inst.register_url`, `inst.install_url` and `inst.finish`.
  - Agama storage configuration (https://agama-project.github.io/docs/user/unattended/storage): A deep dive into storage configuration of target deployments using Agama. Essential for users with complex storage devices and partitions.

    - Agama AutoYaST compatibility reference (https://agama-project.github.io/docs/user/autoyast/reference): Essential for users migrating from AutoYaST or planning to reuse AutoYaST profiles. It details the support status of various AutoYaST profile sections and elements within Agama.
    - Agama CLI reference (https://agama-project.github.io/docs/user/cli): A complete list of all the Agama commands.
- Agama project on GitHub (https://github.com/agama-project/agama): Contains the source code for the Agama installer, which you can use to deep dive into the installer's internals.
    - Agama profile schema (https://github.com/agama-project/agama/blob/master/rust/agama-lib/share/profile.schema.json)
    - Agama profile example in Jsonnet (https://github.com/agama-project/agama/blob/master/rust/agama-lib/share/examples/profile.jsonnet)
    - Agama storage schema (https://github.com/agama-project/agama/blob/master/rust/agama-lib/share/storage.schema.json)
    - Agama storage examples (https://github.com/agama-project/agama/tree/master/rust/agama-lib/share/examples/storage)
- For information on multipath support in Linux, refer to https://en.wikipedia.org/wiki/Linux_DM_Multipath.

10 Legal Notice

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

A 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:

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

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

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

D. Preserve all the copyright notices of the Document.

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

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

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

H. Include an unaltered copy of this License.

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

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

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

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

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

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

O. Preserve any Warranty Disclaimers.

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

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

9. TERMINATION

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

10. FUTURE REVISIONS OF THIS LICENSE

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