Setting Up a PXE Boot Server

Preboot Execution Environment (PXE) is a network boot standard that enables systems to boot and install an operating system over the network instead of using local installation media. A PXE boot server simplifies and automates operating system deployment across multiple machines, reducing manual effort and improving consistency. On SUSE Linux Enterprise Server 16.1, you can configure a PXE boot server with support for UEFI Secure Boot and the Agama installer to create a scalable, network-based installation environment.

Prerequisites

- A SUSE Linux Enterprise Server 16.1 system with administrative privileges
- Internet connection to fetch ISO images
- Static IP configuration for the PXE server

Publication Date: 05 Nov 2026

Contents

1 Overview of PXE booting with SUSE Linux Enterprise Server 16.1 3

2 Preparing the network for PXE boot services 6

3 Installing the required PXE server components 10

4 Creating GRUB 2 NetBoot directories for PXE server 14

5 Preparing the installer image content 18

6 Configuring GRUB 2 for PXE boot 29

7 Configuring TFTP for PXE booting 42

8 Configuring nginx for HTTP delivery 46

9 Configuring a DNS server using dnsmasq 51

10 Configuring an NTP server using `chrony` 55

11 Configuring IPv6 router advertisement 59

12 Configuring a DHCP server using dnsmasq 64

13 Configuring a DHCP server using Kea 71

14 Configuring a DHCP server using ISC DHCP 81

15 Validating PXE server setup 91

16 Legal Notice 99

A GNU Free Documentation License 100

1 Overview of PXE booting with SUSE Linux Enterprise Server 16.1

PXE booting enables machines to boot over the network into an installation or runtime environment without local storage. This section explains how PXE works with SUSE Linux Enterprise Server 16.1 using Agama and live installer images, with a focus on GRUB 2.

1.1 What is PXE booting?

Preboot Execution Environment is a method that allows systems to retrieve boot loaders and OS installers from a network server using DHCP and TFTP or HTTP. It is widely used for provisioning machines without physical media or preinstalled operating systems.

1.2 Benefits of PXE booting

PXE booting simplifies provisioning by removing the need for local installation media or manual setup. It enables:

- Unattended installation of many systems over the network
- Centralized management of installer versions and boot configurations
- Support for diverse architectures and firmware types, including UEFI Secure Boot
- Dynamic selection of installers or installation parameters using GRUB 2 menus

1.3 How PXE booting works in SUSE Linux Enterprise Server 16.1

PXE booting in SUSE Linux Enterprise Server 16.1 uses GRUB 2 as the boot loader and the Agama installer as the installation interface. Boot loaders and installer files are provided over the network using HTTP or TFTP, with GRUB 2 fetching the kernel, initrd and live image. PXE clients can use a variety of firmware (including the most common ones such as BIOS or UEFI), boot loader executable or image formats, as required by their architectures such as AMD64/Intel 64, AArch64, ppc64le and s390x. In addition, they must work in both IPv4 and IPv6 networks.

The boot loader passes kernel parameters such as `root=live:` to load the squashfs-based root file system from a live ISO image, starting the Agama interface either locally or as a Web service for a remote Web UI.

1.3.1 Backward compatibility with SLES 15.x

The information in this article applies primarily to SUSE Linux Enterprise Server 16.0 and later. It focuses on PXE booting workflows that integrate with the Agama installer and rely on live installation images. In the context and scope of this article, SLES 16.0 and later versions differ from SLES 15.x in the following ways:

Installer

Uses `dracut` and Agama instead of `linuxrc` and YaST.

DHCP server

Use of ISC DHCP is discontinued (EOL 2022). For a DHCP server, use either Kea or dnsmasq instead.

Boot parameters

Uses `root=live:` parameter to load the agama installer image and the optional `inst.install_url=` for the non-default installation repository, instead of the `install=` parameter.

The boot loader choice (GRUB 2, pxelinux, etc.) remains flexible and is not version-dependent.

1.3.2 Different possible setups and steps

This article consists of mandatory setup steps and optional or alternative configurations. Follow only the sections relevant to your deployment and skip any alternatives that do not apply to your deployment.

Mandatory

Tasks like installing components, preparing the installer image, configuring GRUB 2, and validating the server must be completed in all setups.

File delivery method

An HTTP server (recommended with Agama) like `nginx` and/or a TFTP server like `tftp` or `dnsmasq`.



Note: TFTP is required for ppc64le

The ppc64le architecture uses IEEE 1275 (OpenFirmware), not UEFI. Its GRUB 2 netboot image (`core.elf`) is retrieved exclusively over TFTP. HTTP Boot is not supported on ppc64le. A TFTP server is therefore mandatory in any setup that includes ppc64le clients, regardless of whether HTTP is used for other architectures.

DHCP server

Choose either Kea or dnsmasq.



Note: Limitations and features of your chosen method

- Use **Kea**—the new DHCP server from ISC—as a modern replacement for ISC DHCP. For more information on Kea, refer to https://www.isc.org/kea/. For the ISC DHCP EOL notice, see https://www.isc.org/dhcp/. Kea is a DHCP server and requires separate TFTP server software. The Kea DHCP server supports options for TFTP/PXE boot over IPv4 and IPv6 as well as for HTTP boot via IPv4. The HTTP boot via IPv6 requires that the DHCPv6 server can send the `Vendor Class Option` (see `RFC3315, Section 22.16`), supposed to be used "by a client to identify the vendor," back to the client and is currently not supported.

- **dnsmasq** is a combination of a DNS server, a DHCP server and a TFTP server. You can use it to serve the boot loader, kernel, initrd (and other files) over TFTP. For more information on dnsmasq, refer to https://thekelleys.org.uk/dnsmasq/doc.html. The dnsmasq DHCP server supports options for TFTP/PXE boot over IPv4 and IPv6 as well as for HTTP boot via IPv4. The HTTP boot via IPv6 requires that the DHCPv6 server can send the `Vendor Class Option` (see `RFC3315, Section 22.16`), supposed to be used "by a client to identify the vendor," back to the client and is currently not supported.

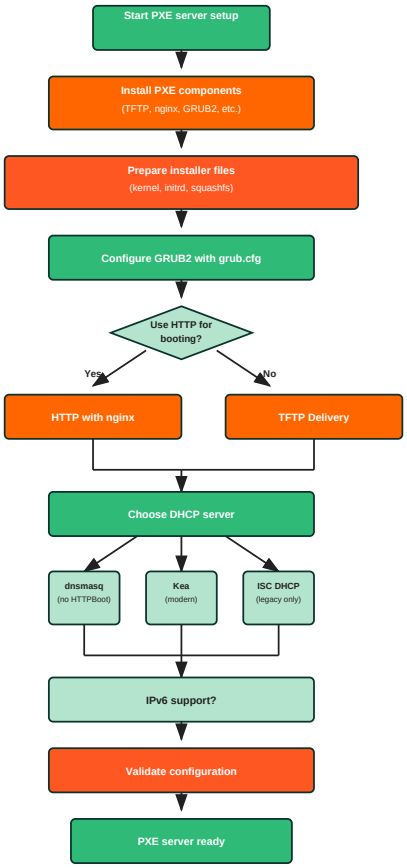


FIGURE 1: SAMPLE PXE SERVER SETUP WORKFLOW

Note: Reading the workflow diagram for ppc64le

The “HTTP with nginx” and “TFTP Delivery” branches in the figure apply to UEFI-capable architectures (AMD64/Intel 64, AArch64) only. As noted above, TFTP delivery is mandatory for ppc64le, and HTTP Boot does not apply to it.

2 Preparing the network for PXE boot services

This module describes the network infrastructure requirements for deploying PXE boot services on SUSE Linux Enterprise Server 16.1.

2.1 PXE network requirements

A PXE server comprises three servers: a DHCP server providing the address and boot file (boot loader) location and a TFTP and/or HTTP server to retrieve the files. In addition, there might be a DNS server, an NTP server and a router with IPv6 support. These are usually separate from the PXE server in a production network. A PXE server running SUSE Linux Enterprise Server 16.1 may also need a specific network interface setup and certain persistent rules added to the firewall. This section illustrates a sample network with suitable IP ranges and the necessary firewall rules.

2.2 Configuring a sample PXE network environment

In this article, we assume the following:

- The PXE server is running on the `eno1` network interface with the following network configuration:

  TABLE 1: SAMPLE PXE NETWORK CONFIGURATION

	IPv4	IPv6	DNS Name
PXE Network	192.168.1.0/24	2001:db8:0:1::/64	example.net
PXE Server	192.168.1.200	2001:db8:0:1::200	pxe.example.net
PXE Gateway	192.168.1.1	2001:db8:0:1::1	
DNS Server	192.168.1.200	2001:db8:0:1::200	
NTP Server	192.168.1.1	2001:db8:0:1::1	

- By default, the router, NTP and DNS servers are external and running on another machine. This article provides some hints, but does not cover their complete configuration.

2.3 Configuring network interfaces and firewall access for PXE services

Configure the network interface and firewall to allow the network services required by the PXE server. `SELinux` permissions for specific PXE services, such as TFTP and HTTP, are covered in their respective sections later in this article.

1. Verify and assign the PXE network interface to the appropriate firewalld zone.

a. Check the currently active zones and their assigned interfaces:

```
> sudo firewall-cmd --get-active-zones
```

b. If `eno1` is not assigned to the `public` zone, assign it:

```
> sudo firewall-cmd --zone=public --change-interface=eno1
```

c. Make the interface assignment persistent across reboots:

```
> sudo firewall-cmd --permanent --zone=public --add-interface=eno1
```

2. Configure the firewall for DNS service access.

a. Allow DNS service for the current session:

```
> sudo firewall-cmd --zone=public --add-service=dns
```

b. Make the change persistent:

```
> sudo firewall-cmd --permanent --zone=public --add-service=dns
```

3. Configure the firewall for NTP service access.

a. Allow NTP service for the current session:

```
> sudo firewall-cmd --zone=public --add-service=ntp
```

b. Make the change persistent:

```
> sudo firewall-cmd --permanent --zone=public --add-service=ntp
```

4. Configure the firewall for DHCP (IPv4) service access.

a. Allow DHCP service for the current session:

```
> sudo firewall-cmd --zone=public --add-service=dhcp
```

b. Make the change persistent:

```
> sudo firewall-cmd --permanent --zone=public --add-service=dhcp
```

5. Configure the firewall for DHCPv6 service access.

a. Allow DHCPv6 service for the current session:

```
> sudo firewall-cmd --zone=public --add-service=dhcpv6
```

b. Make the change persistent:

```
> sudo firewall-cmd --permanent --zone=public --add-service=dhcpv6
```

6. Configure the firewall for TFTP service access.

a. Allow TFTP service for the current session:

```
> sudo firewall-cmd --zone=public --add-service=tftp
```

b. Make the change persistent:

```
> sudo firewall-cmd --permanent --zone=public --add-service=tftp
```

7. Configure the firewall for HTTP service access.

a. Allow HTTP service for the current session:

```
> sudo firewall-cmd --zone=public --add-service=http
```

b. Make the change persistent:

```
> sudo firewall-cmd --permanent --zone=public --add-service=http
```

8. Configure the firewall for HTTPS service access.

a. Allow HTTPS service for the current session:

```
> sudo firewall-cmd --zone=public --add-service=https
```

b. Make the change persistent:

```
> sudo firewall-cmd --permanent --zone=public --add-service=https
```

2.4 Verifying PXE network readiness

This procedure ensured that the PXE server's network interface and firewall were correctly configured for secure and functional operation.

- Verified and assigned the PXE-serving interface (`eno1` in this example) to the `public` firewalld zone.
- Opened the required firewall services for PXE operation, including `dns`, `ntp`, `dhcp`, `dhcpv6`, `tftp`, `http` and `https`.

With these steps complete, the PXE server is securely configured and ready to serve client machines over the network using either IPv4 or IPv6.

3 Installing the required PXE server components

This section explains how to install the necessary packages to support PXE booting in SUSE Linux Enterprise Server 16.1, including GRUB 2, DHCP, TFTP and/or HTTP components.

3.1 Required services and packages for a PXE boot server

To configure a PXE boot server on SUSE Linux Enterprise Server 16.1, you need to install several services and tools. Depending on your setup, you may need the following:

- The `dnsmasq` package provides a combination of a DNS server, a TFTP server and a DHCP server (DHCPv4 and DHCPv6) with limited IPv6 router advertisement (RA) support. It offers the following:
  - dnsmasq DHCP server: Supports conditional delivery of DHCP options depending on request and client architecture for:
    - PXE boot requests using DHCPv4 and DHCPv6
    - HTTP boot requests using DHCPv4

    ### Note: Limitations of dnsmasq for HTTP boot over DHCPv6

    Currently, dnsmasq does not support sending the required `vendor-class` DHCPv6 option.

  - dnsmasq TFTP server: Delivers boot loader files, kernel and initrd over TFTP during PXE boot.

- dnsmasq DNS server: Provides recursive resolution of domain names and IP addresses for client firmware and `/etc/resolv.conf` in the Installer/OS.
- dnsmasq IPv6 RA: Supports sending IPv6 RA when the PXE server is also acting as a router (configurability limited to a “common RA pattern”).

- The `kea` package is a DHCP server and a successor of the ISC DHCP server. It supports conditional delivery of DHCP options depending on request and client architecture for:
  - PXE boot requests using DHCPv4 and DHCPv6
  - HTTP boot requests using DHCPv4



Note: Limitations of Kea for HTTP Boot over DHCPv6

Currently, Kea does not support sending the required `vendor-class` DHCPv6 option. For more information, see https://kea.readthedocs.io/en/latest/arm/dhcp6-srv.html#id4.

- A TFTP server delivers the boot loader files, kernel and initrd over TFTP. When using Kea for PXE boot, the `tftp` package provides this; it is not required for HTTP Boot. If you are using `dnsmasq`, you do not need the `tftp` package.
- A Web server such as the `nginx` package to serve installer images over HTTP.



Note: Necessity for HTTP servers

An HTTP/HTTPS server like nginx is almost always required. Its use extends beyond just HTTP Boot. In particular, you may need it in the following scenarios:

- It is a basic requirement for HTTP Boot.
- We recommend it for providing the `squashfs.img`. You can use `root=live:tftp://.../squashfs.img` in the boot command line.
- We also recommend it for providing RPMs to Agama in the `inst.install_url=http://.../install/` boot command-line parameter on a `SLES-16.0-Full-*.inline.iso`, along with installation profiles and other files for unattended installation.

- The GRUB 2 boot loader packages provide network boot for supported architectures and methods. For example, the AMD64/Intel 64 architecture offers two methods for network boot: BIOS and UEFI. Additionally, UEFI generally supports PXE (TFTP) and HTTP Boot. Other boot loaders, such as pxelinux, do not support UEFI and HTTP Boot.
- Optionally, a router advertisement daemon for IPv6, such as the `radvd` package. It is required on SLES if it is also acting as a router for an installer network to perform the following:
  - Configure routing on a network for PXE or HTTP Boot clients.
  - Enable DHCPv6 use on a network for PXE or HTTP Boot clients.

3.2 Prerequisites for installing PXE server components

- A system running SUSE Linux Enterprise Server 16.1 with administrative privileges, registered to the SUSE Customer Center and configured with access to the appropriate online repositories using SUSEConnect.
- Enabled SLE modules: Server Applications Module, Legacy Module and Base System Module.
- Access to the SLE module repositories for network services and boot loaders.
- A working internet connection to fetch packages.

3.3 Installing PXE boot server packages and services

Use the following steps to install the core packages required for the PXE boot server.

PROCEDURE 1: INSTALLING NECESSARY PACKAGES FOR A PXE BOOT SERVER

1. Install the GRUB 2 boot loader and the nginx HTTP server as common requirements.

   ```
   > sudo zypper install grub2 nginx
   ```

2. Run any of the following commands to install the required packages for your approach:

   - `kea` for the DHCP server, `tftp` for the TFTP server

     ```
     > sudo zypper install kea tftp
     ```

   - `dnsmasq` as the common provider for DHCP, DNS and TFTP servers

```
> sudo zypper install dnsmasq
```



Note: Limitations of DHCP servers provided by Kea and dnsmasq

HTTP boot over IPv6 is *currently* not supported by DHCP servers provided by the `kea` and `dnsmasq` packages. It does not support sending a `vendor-class` option back to the HTTP client, as required by the UEFI specification.

3. Optionally, install additional architecture-specific GRUB 2 targets if you plan to support other platforms.

   - For AMD64/Intel 64 architecture:

     ```
     > sudo zypper install grub2-x86_64-efi grub2-i386-pc
     ```

   - For AArch64 architecture:

     ```
     > sudo zypper install grub2-arm64-efi
     ```

   - For ppc64le architecture:

     ```
     > sudo zypper install grub2-powerpc-ieee1275
     ```



Note: How the PXE server delivers GRUB 2 packages to clients that are different from the server machine's architecture

The GRUB 2 architecture-specific `noarch.rpm` packages are included in the `noarch` subdirectory of the installation media/repository, regardless of the architecture of the machine on which the PXE server is set up. That is, you can install `grub2-arm64-efi` and `grub2-powerpc-ieee1275` packages in a PXE server running on an AMD64/Intel 64 machine, in order to support clients with other architectures.

4. Optionally, you may install the `shim` package if you need UEFI Secure Boot for AMD64/Intel 64 or AArch64, but do not want to use the files from the installation media ISO.

   ```
   > sudo zypper install shim
   ```

5. Optionally, install the router advertisement daemon `radvd` if you want to use the PXE server as a router (not recommended for production networks).

```
> sudo zypper install radvd
```

6. Install the `rsync` utility to conveniently copy or sync the ISO and directory tree.

```
> sudo zypper install rsync
```

7. Ensure the services are installed but not yet started. Configuration will be covered in later sections.

4 Creating GRUB 2 NetBoot directories for PXE server

This section explains creating GRUB 2 NetBoot directories for PXE servers using **`grub2-mknetdir`**, which generates architecture-specific directories for AMD64/Intel 64 (UEFI and BIOS), AArch64 and ppc64le systems. For Secure Boot support, administrators must copy signed EFI files from installation media or use the `shim` package to replace the default unsigned boot loader files.

4.1 GRUB 2NetBoot directory structure for PXE boot

This section describes how to set up GRUB 2 NetBoot directories for PXE server deployment across multiple architectures. The **`grub2-mknetdir`** command creates architecture-specific directories under `/srv/tftpboot/boot/grub2/` for different platforms. For example, AMD64/Intel 64 systems generate both UEFI (`x86_64-efi`) and legacy BIOS (`i386-pc`) directories, and AArch64 systems create their UEFI directory in `arm64-efi`. ppc64le systems (`powerpc-ieee1275`) use OpenFirmware/OPAL rather than UEFI, and have their own Secure Boot mechanism separate from UEFI Secure Boot; the GRUB 2 boot loader is in `/boot/grub2/grub.elf` on the ISOs.

Important

In the context of this section, Secure Boot applies to AMD64/Intel 64 and AArch64 architectures. GRUB 2 PXE Secure Boot for ppc64le (which uses a different platform-specific mechanism) is not covered here.

For Secure Boot support, which is not provided by the default unsigned `core.efi` files, administrators must obtain signed EFI files. The signed files can be obtained from either of the following sources:

- The `shim` package installed on the PXE server (**`zypper install shim`**), which provides `shim.efi`, `grub.efi` and `MokManager.efi` under `/usr/share/efi/ARCH/`. Note that the `shim` package provides files only for the architecture of the PXE server itself. To support a different architecture, use the installation media ISO or manually extract the files from the architecture-specific `shim` package.
- The installation media ISO (mounted and copied from `/mnt/EFI/BOOT/*.efi`), which provides signed EFI files for the target architecture regardless of the PXE server architecture.

The files must be copied to the appropriate architecture directories. This ensures proper symbolic link resolution to keep all files within the TFTP root directory.

4.2 Requirements for creating GRUB 2 NetBoot directories

- Ensure that you have installed the following packages: `grub2`, `tftp` and the GRUB 2 platform package for every architecture you want to serve: `grub2-x86_64-efi`, `grub2-i386-pc`, `grub2-arm64-efi` and `grub2-powerpc-ieee1275`.
- Ensure that you have either the installation media (ISO) available for mounting, or the `shim` package installed on the system. You can download the installation media (ISO) for your target architecture from the SUSE Customer Center.

4.3 Preparing NetBoot directories and configuring UEFI Secure Boot

This procedure creates the required GRUB 2 directory structure for PXE network booting and optionally configures UEFI Secure Boot support across multiple architectures.

1. Create a GRUB 2 NetBoot directory structure:

   ```
   > sudo grub2-mknetdir --net-directory=/srv/tftpboot --subdir=/boot/grub2
   ```

   Without an explicit `--directory` option, **`grub2-mknetdir`** generates a directory for every GRUB 2 platform package installed on the PXE server, along with the shared `fonts/` and `locale/` directories. With the packages listed under *Section 4.2, “Requirements for creating GRUB 2 NetBoot*

directories" installed, this creates `/srv/tftpboot/boot/grub2/x86_64-efi`, `/srv/tftpboot/boot/grub2/i386-pc`, `/srv/tftpboot/boot/grub2/arm64-efi`, and `/srv/tftpboot/boot/grub2/powerpc-ieee1275`.



Warning

Do not manually overwrite the `grub.cfg` file created by **`grub2-mknetdir`**.

2. You can use the `/srv/tftpboot/boot/grub2/ARCH-efi/core.efi` file installed by the **`grub2-mknetdir`** command for AMD64/Intel 64 or AArch64 architectures for UEFI PXE as well, without performing any of the steps below. However, it is *not signed*: a client with UEFI Secure Boot enabled in firmware rejects it and fails to boot. This is not a concern if UEFI Secure Boot is disabled in the client's firmware, which is the default on most hardware. Only perform the following steps if your PXE clients must keep UEFI Secure Boot enabled during boot.



Note: `core.efi` and subdirectory setups

The `core.efi` file generated by **`grub2-mknetdir`** has `/boot/grub2` hardcoded as an embedded prefix. GRUB 2 always fetches modules and configuration from this prefix relative to the TFTP root, regardless of where `core.efi` itself was loaded from. If you need to serve multiple environments from subdirectories under one TFTP root, use `shim` and `grub.efi` from the installation media ISO or the `shim` package instead, as described below.

To optionally enable UEFI Secure Boot for the supported AMD64/Intel 64 and AArch64 architectures, perform any of the following steps:

- Copy the necessary files from the installation media ISO:

  a. Mount the ISO image.

  ```
  > sudo mount -o loop /PATH/TO/SLES.ISO /mnt
  ```

  b. Copy EFI files.

  ```
  > sudo cp -v /mnt/EFI/BOOT/*.efi
  /srv/tftpboot/boot/grub2/ARCH-efi/ (1)
  ```

  (1) Replace *ARCH*`-efi` with `x86_64-efi` or `arm64-efi`, the architectures that support Secure Boot.

c. Unmount the installation media ISO.

```
> sudo umount /mnt
```

- For AMD64/Intel 64 and AArch64 architectures, use the `shim` package if you do not want to use the files from the installation media ISO:

  a. If not already installed, install the `shim` package.

  ```
  > sudo zypper install shim
  ```

  b. Copy the signed boot loader files for the necessary architecture:

  i. Copy the `shim.efi` file.

  - For AMD64/Intel 64 architecture:

  ```
  > sudo cp -v -p -L /usr/share/efi/x86_64/shim.efi /srv/tftpboot/
  boot/grub2/x86_64-efi/bootx64.efi
  ```

  - For AArch64 architecture:

  ```
  > sudo cp -v -p -L /usr/share/efi/aarch64/shim.efi /srv/
  tftpboot/boot/grub2/arm64-efi/bootaa64.efi
  ```

  ii. Copy the `grub.efi` file.

  - For AMD64/Intel 64 architecture:

  ```
  > sudo cp -v -p -L /usr/share/efi/x86_64/grub.efi /srv/tftpboot/
  boot/grub2/x86_64-efi/
  ```

  - For AArch64 architecture:

  ```
  > sudo cp -v -p -L /usr/share/efi/aarch64/grub.efi /srv/
  tftpboot/boot/grub2/arm64-efi/
  ```

  iii. Copy the `MokManager.efi` file.

  - For AMD64/Intel 64 architecture:

  ```
  > sudo cp -v -p -L /usr/share/efi/x86_64/MokManager.efi /srv/
  tftpboot/boot/grub2/x86_64-efi/
  ```

- For AArch64 architecture:

```
> sudo cp -v -p -L /usr/share/efi/aarch64/MokManager.efi /srv/
tftpboot/boot/grub2/arm64-efi/
```



Note

The `-L` flag resolves symbolic links to ensure files stay within the TFTP root.

5 Preparing the installer image content

This section describes how to extract and organize installer files from SUSE Linux Enterprise Server 16.1 installation media for PXE boot environments. It covers both `.install.iso` images and RPM packages, with specific instructions for different architectures and installation types.

5.1 Installer file requirements for Agama

SUSE Linux Enterprise Server 16.1 provides installer files in multiple formats to support different PXE boot scenarios. The Agama installer requires three files: the kernel image (`linux`), the initial RAM disk (`initrd`) and the compressed root file system (`squashfs.img`). These files must be extracted from the installation media and organized in a directory structure accessible via TFTP and HTTP.

This section covers preparation methods for both `.install.iso` images and RPM packages, ensuring compatibility with various architectures and installation types supported by SUSE Linux Enterprise Server 16.1.

5.2 Preparing sources and environment for installer file extraction

- SUSE Linux Enterprise Server 16.1 installation media, as available in SUSE Customer Center. Choose from:
  - Online ISO: Installer-only for network installations

        - For AMD64/Intel 64: `SLES-16.0-Online-x86_64-BUILD.install.iso`
        - For AArch64: `SLES-16.0-Online-aarch64-BUILD.install.iso`
        - For ppc64le: `SLES-16.0-Online-ppc64le-BUILD.install.iso`
    - Full ISO: Installer with installation repository
        - For AMD64/Intel 64: `SLES-16.0-Full-x86_64-BUILD.install.iso`
        - For AArch64: `SLES-16.0-Full-aarch64-BUILD.install.iso`
        - For ppc64le: `SLES-16.0-Full-ppc64le-BUILD.install.iso`
    - RPM packages:
        - For AMD64/Intel 64: `tftpboot-agama-installer-SUSE_SLE_16_PXE-x86_64`
        - For AArch64: `tftpboot-agama-installer-SUSE_SLE_16_PXE-aarch64`
        - For ppc64le: `tftpboot-agama-installer-SUSE_SLE_16_PXE-ppc64le`
- A temporary mount point, such as `/mnt`.
- Sufficient disk space under `/srv/tftpboot` and `/srv/www/htdocs/install` for your chosen installation method.
- Administrative privileges to create directories and copy files.



Note: The ${BUILD} placeholder

Both the file names above and the commands below reference the downloaded ISO through a `${BUILD}` shell variable. Export it once with the build identifier of the file you download, then every file name and command that follows can be copied and pasted exactly as shown:

```
> export BUILD=Build135.5
> wget https://.../SLES-16.0-Online-x86_64-${BUILD}.install.iso
```

The build identifier is whatever the download source offers for your version. On https://www.suse.com/download/sles/ it is a release tag such as `GM` (and later maintenance updates such as `QU2` or `QU3`); on the internal build service it is a build number such as `Build135.5`. It changes with every new build, so set it to match the file you actually download.

5.3 Preparing installer files from ISO images

ISO images provide a straightforward method for extracting installer files. The following procedures cover both Online and Full ISO types for different architectures.

5.3.1 Using Online ISO images

Online ISO images contain only the installer components, requiring network access to installation repositories during system installation. This corresponds to the `SLES-16.0 Online Installation` boot menu entry in GRUB 2.

PROCEDURE 2: EXTRACTING FILES FROM ONLINE ISO

1. Create the directory structure for installer files.
   - For AMD64/Intel 64:

     ```
     > sudo mkdir -p /srv/tftpboot/boot/images/SLES-16.0/x86_64/
     ```

   - For AArch64:

     ```
     > sudo mkdir -p /srv/tftpboot/boot/images/SLES-16.0/aarch64/
     ```

   - For ppc64le:

     ```
     > sudo mkdir -p /srv/tftpboot/boot/images/SLES-16.0/ppc64le/
     ```

2. Mount the Online ISO image.
   - For AMD64/Intel 64:

     ```
     > sudo mount -oro,loop
     /srv/www/htdocs/install/iso/SLES-16.0-Online-x86_64-${BUILD}.install.iso
     /mnt
     ```

   - For AArch64:

     ```
     > sudo mount -oro,loop
     /srv/www/htdocs/install/iso/SLES-16.0-Online-aarch64-${BUILD}.install.iso
     /mnt
     ```

   - For ppc64le:

     ```
     > sudo mount -oro,loop
     ```

```
/srv/www/htdocs/install/iso/SLES-16.0-Online-ppc64le-${BUILD}.install.iso
/mnt
```

3. Copy the kernel and initrd files and the compressed root file system.

   - For AMD64/Intel 64:

     ```
     > sudo cp /mnt/boot/x86_64/loader/linux
     /srv/tftpboot/boot/images/SLES-16.0/x86_64/
     ```

     ```
     > sudo cp /mnt/boot/x86_64/loader/initrd
     /srv/tftpboot/boot/images/SLES-16.0/x86_64/
     ```

     ```
     > sudo cp /mnt/LiveOS/squashfs.img
     /srv/tftpboot/boot/images/SLES-16.0/x86_64/
     ```

   - For AArch64:

     ```
     > sudo cp /mnt/boot/aarch64/loader/linux
     /srv/tftpboot/boot/images/SLES-16.0/aarch64/
     ```

     ```
     > sudo cp /mnt/boot/aarch64/loader/initrd
     /srv/tftpboot/boot/images/SLES-16.0/aarch64/
     ```

     ```
     > sudo cp /mnt/LiveOS/squashfs.img
     /srv/tftpboot/boot/images/SLES-16.0/aarch64/
     ```

   - For ppc64le:

     ```
     > sudo cp /mnt/boot/ppc64le/linux
     /srv/tftpboot/boot/images/SLES-16.0/ppc64le/
     ```

     ```
     > sudo cp /mnt/boot/ppc64le/initrd
     /srv/tftpboot/boot/images/SLES-16.0/ppc64le/
     ```

     ```
     > sudo cp /mnt/LiveOS/squashfs.img
     /srv/tftpboot/boot/images/SLES-16.0/ppc64le/
     ```



Note: Path variation for ppc64le installation media

The `loader` subdirectory may or may not be present on ppc64le installation media depending on the version. If `linux` and `initrd` are not found at `/mnt/boot/ppc64le/linux` and `/mnt/boot/ppc64le/initrd`, try `/mnt/boot/ppc64le/loader/linux` and `/mnt/boot/ppc64le/loader/initrd` instead.

4. Unmount the ISO image:

```
> sudo umount /mnt
```

5.3.2 Using Full ISO images

Full ISO images include both the installer and installation repositories, enabling local installations without external network dependencies. It corresponds to the `SLES-16.0 Local Installation` boot menu entry in GRUB 2 with the additional `inst.install_url=http://pxe.example.net/install/SLES-16.0/${arch}` parameter.

PROCEDURE 3: EXTRACTING FILES FROM FULL ISO

1. Create the directory for installer files.

   - For AMD64/Intel 64:

     ```
     > sudo mkdir -p /srv/tftpboot/boot/images/SLES-16.0/x86_64/
     ```

   - For AArch64:

     ```
     > sudo mkdir -p /srv/tftpboot/boot/images/SLES-16.0/aarch64/
     ```

   - For ppc64le:

     ```
     > sudo mkdir -p /srv/tftpboot/boot/images/SLES-16.0/ppc64le/
     ```

2. Create the directory for the installation repository:

```
> sudo mkdir -p /srv/www/htdocs/install/SLES-16.0
```

3. Mount the Full ISO image:

   - For AMD64/Intel 64:

     ```
     > sudo mount -oro,loop
     /srv/www/htdocs/install/iso/SLES-16.0-Full-x86_64-${BUILD}.install.iso
     /mnt
     ```

   - For AArch64:

     ```
     > sudo mount -oro,loop
     /srv/www/htdocs/install/iso/SLES-16.0-Full-aarch64-${BUILD}.install.iso
     /mnt
     ```

   - For ppc64le:

     ```
     > sudo mount -oro,loop
     /srv/www/htdocs/install/iso/SLES-16.0-Full-ppc64le-${BUILD}.install.iso
     /mnt
     ```

4. Copy the kernel files.

   - For AMD64/Intel 64:

     ```
     > sudo cp /mnt/boot/x86_64/loader/linux  /srv/tftpboot/boot/images/SLES-16.0/
     x86_64/
     ```

   - For AArch64:

     ```
     > sudo cp /mnt/boot/aarch64/loader/linux  /srv/tftpboot/boot/images/SLES-16.0/
     aarch64/
     ```

   - For ppc64le:

     ```
     > sudo cp /mnt/boot/ppc64le/linux /srv/tftpboot/boot/images/SLES-16.0/ppc64le/
     ```

5. Copy the initrd files.

   - For AMD64/Intel 64:

     ```
     > sudo cp /mnt/boot/x86_64/loader/initrd /srv/tftpboot/boot/images/SLES-16.0/
     x86_64/
     ```

- For AArch64:

```
> sudo cp /mnt/boot/aarch64/loader/initrd /srv/tftpboot/boot/images/SLES-16.0/
aarch64/
```

- For ppc64le:

```
> sudo cp /mnt/boot/ppc64le/initrd /srv/tftpboot/boot/images/SLES-16.0/ppc64le/
```



Note: Path variation for ppc64le installation media

The `loader` subdirectory may or may not be present on ppc64le installation media depending on the version. If `linux` and `initrd` are not found at `/mnt/boot/ppc64le/linux` and `/mnt/boot/ppc64le/initrd`, try `/mnt/boot/ppc64le/loader/linux` and `/mnt/boot/ppc64le/loader/initrd` instead.

6. Copy the compressed root file system.

   - For AMD64/Intel 64:

```
> sudo cp /mnt/LiveOS/squashfs.img
/srv/tftpboot/boot/images/SLES-16.0/x86_64/
```

   - For AArch64:

```
> sudo cp /mnt/LiveOS/squashfs.img
/srv/tftpboot/boot/images/SLES-16.0/aarch64/
```

   - For ppc64le:

```
> sudo cp /mnt/LiveOS/squashfs.img
/srv/tftpboot/boot/images/SLES-16.0/ppc64le/
```

7. Copy the installation repository for local HTTP server access.

   - For AMD64/Intel 64:

```
> sudo rsync -avP /mnt/install/
/srv/www/htdocs/install/SLES-16.0/x86_64/
```

   - For AArch64:

```
> sudo rsync -avP /mnt/install/
```

```
/srv/www/htdocs/install/SLES-16.0/aarch64/
```

- For ppc64le:

```
> sudo rsync -avP /mnt/install/
/srv/www/htdocs/install/SLES-16.0/ppc64le/
```

8. Unmount the ISO image:

```
> sudo umount /mnt
```

5.4 Preparing installer files from RPM packages

RPM packages provide an alternative method for obtaining Online Installer files.

PROCEDURE 4: INSTALLING AND USING TFTPBOOT RPM PACKAGES

1. Install the required packages.

   - For AMD64/Intel 64:

   ```
   > sudo zypper in
   tftpboot-agama-installer-SUSE_SLE_16-x86_64
   ```

   - For AArch64:

   ```
   > sudo zypper in
   tftpboot-agama-installer-SUSE_SLE_16-aarch64
   ```

   - For ppc64le:

   ```
   > sudo zypper in
   tftpboot-agama-installer-SUSE_SLE_16-ppc64le
   ```

2. Create the directory for tftpboot and navigate to it.

   - For AMD64/Intel 64:

   ```
   > sudo mkdir -p
   /srv/tftpboot/boot/images/SLES-16.0/x86_64
   ```

   ```
   > sudo cd
   /srv/tftpboot/boot/images/SLES-16.0/x86_64
   ```

- For AArch64:

```
> sudo mkdir -p
/srv/tftpboot/boot/images/SLES-16.0/aarch64
```

```
> sudo cd
/srv/tftpboot/boot/images/SLES-16.0/aarch64
```

- For ppc64le:

```
> sudo mkdir -p
/srv/tftpboot/boot/images/SLES-16.0/ppc64le
```

```
> sudo cd
/srv/tftpboot/boot/images/SLES-16.0/ppc64le
```

3. Copy the `linux`, `initrd` and `squashfs.img` files into the tftpboot directory.

- For AMD64/Intel 64:

```
> sudo cp -v \
/usr/share/tftpboot-installation/agama-installer-SUSE_SLE_16/x86_64/loader/
linux \
/usr/share/tftpboot-installation/agama-installer-SUSE_SLE_16/x86_64/loader/
initrd \
/usr/share/tftpboot-installation/agama-installer-SUSE_SLE_16/x86_64/loader/
squashfs.img \
.
```

- For AArch64:

```
> sudo cp -v \
/usr/share/tftpboot-installation/agama-installer-SUSE_SLE_16/aarch64/loader/
linux \
/usr/share/tftpboot-installation/agama-installer-SUSE_SLE_16/aarch64/loader/
initrd \
/usr/share/tftpboot-installation/agama-installer-SUSE_SLE_16/aarch64/loader/
squashfs.img \
.
```

- For ppc64le:

```
> sudo cp -v \
/usr/share/tftpboot-installation/agama-installer-SUSE_SLE_16/ppc64le/loader/
linux \
```

```
/usr/share/tftpboot-installation/agama-installer-SUSE_SLE_16/ppc64le/loader/
initrd \
/usr/share/tftpboot-installation/agama-installer-SUSE_SLE_16/ppc64le/loader/
squashfs.img \
.
```

5.5 Recommended directory structure

Organize the extracted files according to the following directory layout to ensure consistency and ease of maintenance. This structure supports multiple architectures and installation types.

EXAMPLE 1: COMPLETE PXE SERVER DIRECTORY LAYOUT

```
/srv/tftpboot/
├── boot/
│   ├── grub2/
│   │   ├── x86_64-efi/
│   │   │   ├── bootx64.efi
│   │   │   └── grub.cfg
│   │   ├── i386-pc/
│   │   │   └── core.0
│   │   ├── arm64-efi/
│   │   │   └── bootaa64.efi
│   │   └── powerpc-ieee1275/
│   │       └── core.elf
│   └── images/
│       └── SLES-16.0/
│           ├── x86_64/
│           │   ├── linux (1)
│           │   ├── initrd (2)
│           │   └── squashfs.img (3)
│           ├── aarch64/
│           └── ppc64le/
/srv/www/htdocs/install/
└── SLES-16.0/
    ├── x86_64/ (4)
    ├── aarch64/
    └── ppc64le/
```

1. Kernel image extracted from installation media
2. Initial RAM disk image
3. Compressed root file system for Agama installer
4. Installation repository from `install` directory from the Full ISO (optional)

5.6 Resetting SELinux labels

Reset the SELinux labels of the `/srv/tftpboot` directory to those defined in the policy.

```
> sudo restorecon -Rv /srv/tftpboot
```

5.7 Verifying the installation

After extracting and organizing the installer files, verify that all required components are present and accessible.

PROCEDURE 5: VERIFICATION STEPS

1. Check that the required files are present.

   - For AMD64/Intel 64:

     ```
     > ls -la
     /srv/tftpboot/boot/images/SLES-16.0/x86_64/*
     ```

   - For AArch64:

     ```
     > ls -la
     /srv/tftpboot/boot/images/SLES-16.0/aarch64/*
     ```

   - For ppc64le:

     ```
     > ls -la
     /srv/tftpboot/boot/images/SLES-16.0/ppc64le/*
     ```

2. Ensure proper file permissions:

   ```
   > sudo find /srv/tftpboot/boot/images/ -type d -exec chmod 0755 {} \;
   ```

   ```
   > sudo find /srv/tftpboot/boot/images/ -type f -exec chmod 0644 {} \;
   ```



Important: File accessibility

Ensure that all extracted files are readable by the TFTP and HTTP services. PXE clients access these files directly during boot, so incorrect permissions or service configuration will cause boot failures.

5.8 Next steps: Configure PXE services

The installer files are now extracted and organized. Next:

- Configure GRUB 2 for PXE boot with menu entries referencing these files.
- Set up HTTP and TFTP services to serve the extracted content.
- Configure DHCP to direct PXE clients to the appropriate boot loaders.

The extracted files will be referenced in GRUB 2 configuration using paths, such as:

- For AMD64/Intel 64: `root=live:http://pxe.example.net/boot/images/SLES-16.0/x86_64/squashfs.img`
- For AArch64: `root=live:http://pxe.example.net/boot/images/SLES-16.0/aarch64/squashfs.img`
- For ppc64le: `root=live:http://pxe.example.net/boot/images/SLES-16.0/ppc64le/squashfs.img`

6 Configuring GRUB 2 for PXE boot

This section describes how to configure the GRUB 2 boot loader for PXE-based booting on SUSE Linux Enterprise Server 16.1. It covers creating the network boot directory structure, setting up architecture-specific boot loaders and building a boot menu that supports multiple architectures and installation types.

6.1 GRUB 2 PXE boot configuration structure

GRUB 2 serves as the network boot loader for PXE clients, loading kernel and initrd files to start the Agama installer. This section shows how to build a GRUB 2 configuration that detects the client architecture, selects the correct network interface and provides one boot menu covering multiple installation types and target architectures.

The configuration is split into separate files for architecture detection, variable definitions and boot menu entries. This split lets you add machine-specific configurations and automated installation profiles without duplicating the shared setup.

6.2 Requirements for GRUB 2 PXE boot configuration

- Ensure that the GRUB 2 network boot directory structure is in place, as described in the previous sections.
- Ensure that the installer files are properly organized as described in the previous sections.
- GRUB 2 packages for all target architectures must be installed: `grub2-x86_64-efi`, `grub2-i386-pc`, `grub2-arm64-efi` and `grub2-powerpc-ieee1275`
- The `shim` package for UEFI Secure Boot support (optional).
- Administrative access to `/srv/tftpboot` or equivalent PXE root.

6.3 Creating the GRUB 2 configuration

The GRUB 2 configuration file handles three main tasks: detecting the client's architecture, managing network interfaces and loading other configuration files. This modular approach provides flexibility for different deployment scenarios.

PROCEDURE 6: SETTING UP THE MAIN `grub.cfg` FILE

- Create the main GRUB 2 configuration file at `/srv/tftpboot/boot/grub2/grub.cfg`:

```
> sudo cat > /srv/tftpboot/boot/grub2/grub.cfg << 'EOF'
# Architecture detection and mapping
if [ "$grub_cpu" == "i386" ]; then
  set arch='x86_64'
elif [ "$grub_cpu" == "x86_64" ]; then
  set arch='x86_64'
elif [ "$grub_cpu" == "arm64" ]; then
  set arch='aarch64'
elif [ "$grub_cpu" == "powerpc" ]; then
  set arch='ppc64le'
fi

if [ "X$arch" == "X" ]; then
  echo "ERROR: No architecture found for ${grub_cpu}"
  exit
else
  echo "Running on $arch CPU architecture"
fi
export arch
```

```
# Network interface configuration for PXE-selected NIC
# - dracut based images on SLE-16:
set ipcfg="ifname=pxe0:${net_default_mac} ip=pxe0:dhcp"
export ipcfg
# - linuxrc installer on SLE-15:
set ifcfg="ifcfg=${net_default_mac}=dhcp"
export ifcfg

# Define typical serial console kernel parameter
#set sconsole="console=tty0 console=ttyS0,115200n8"
#export sconsole

# Load machine-specific configuration if available
set config="/boot/config"
export config
if [ -s "${config}/${net_default_mac}/grub.cfg" ]; then
  ## Source a host specific configuration of grub menu:
  source "${config}/${net_default_mac}/grub.cfg"
else
  ## Source default grub boot menu:
  source "${prefix}/menu.cfg"
fi
EOF
```

KEY CONFIGURATION ELEMENTS

Architecture detection

Maps GRUB 2 CPU types to distribution architectures, enabling unified menu entries that work across different hardware platforms.

Network interface management

Defines an `${ipcfg}` variable using the grub2 variable `${net_default_mac}` to activate dhcp only on the PXE boot interface named `pxe0`, avoiding network probing issues on multi-interface systems.



Note: Selecting a specific network interface on a multi-NIC machine

The default configuration above never needs to know an interface's real name: it binds a custom name (`pxe0`) to whichever NIC actually performed the PXE boot, using dracut's `ifname=<name>:<MAC>` syntax. The machine-specific example later in this topic instead hardcodes a literal interface name (`eno1`). If a target machine has multiple NICs and you need to boot from a specific one, for example the third, identify its real interface name or MAC address on that machine first (for example, with **`ip link show`**, or from the hardware

vendor's documentation), then either reference that name directly in `ip=<interface>:dhcp`, or bind a custom name to its MAC address with `ifname=` the same way the default configuration does. dracut also accepts multiple `ip=` parameters at once, but then requires `bootdev=<interface>` to specify which one provides routing and netroot information.

Utility definitions

Defines a typical `${sconsole}` variable with serial console parameters.

Machine-specific configuration

Loads optional per-machine configuration files based on MAC address, enabling customized per-machine boot parameters and automated installation profiles.

6.4 Creating the unified boot menu for multiple PXE installations

The boot menu uses variables from the main configuration to provide architecture-agnostic menu entries that automatically adapt to different hardware platforms and installation types.

PROCEDURE 7: SETTING UP THE MENU.CFG FILE

- Create the unified boot menu at `/srv/tftpboot/boot/grub2/menu.cfg`:

```
> sudo cat > /srv/tftpboot/boot/grub2/menu.cfg << 'EOF'
menuentry 'SLES-16.0 Online Installation' {
  linux  /boot/images/SLES-16.0/${arch}/linux showopts root=live:http://
pxe.example.net/boot/images/SLES-16.0/${arch}/squashfs.img ${ipcfg} ${sconsole}
 ${autoinstall}
  initrd /boot/images/SLES-16.0/${arch}/initrd
}

menuentry 'SLES-16.0 Local Installation' {
  linux  /boot/images/SLES-16.0/${arch}/linux showopts root=live:http://
pxe.example.net/boot/images/SLES-16.0/${arch}/squashfs.img inst.install_url=http://
pxe.example.net/install/SLES-16.0/${arch} ${ipcfg} ${sconsole} ${autoinstall}
  initrd /boot/images/SLES-16.0/${arch}/initrd
}
EOF
```



Note: Menu entry flexibility

The menu entries use variables that are automatically populated based on the client architecture and configuration. The `${arch}` variable ensures the correct files are loaded.

The optional `${ipcfg}` variable causes only the PXE-selected network interface to be set up.

The optional `${sconsole}` variable enables a serial console in the installer system.



Note: Mixing protocols for kernel and initrd loading

GRUB 2 automatically prepends the protocol and server it booted from to every path it loads, including the `linux` and `initrd` lines, so you normally do not need to specify a protocol explicitly. Only specify it if you want GRUB 2 to load these particular files via a *different* protocol than the one it booted with, for example booting via TFTP but fetching `linux` and `initrd` over HTTP:

```
menuentry 'SLES-16.0 Online Installation' {
linux  (http,192.168.1.200)/boot/images/SLES-16.0/${arch}/linux showopts
 root=live:http://pxe.example.net/boot/images/SLES-16.0/${arch}/squashfs.img
 ${ipcfg} ${sconsole} ${autoinstall}
initrd (http,192.168.1.200)/boot/images/SLES-16.0/${arch}/initrd
}
```

Use an IP address rather than a host name in the protocol prefix. GRUB 2 itself resolves the boot server as an IP address, and using a host name here would trigger a separate DNS lookup for every path.

6.5 Machine-specific configurations

For advanced deployments, you can create machine-specific configuration files that override default settings or provide automated installation parameters.

PROCEDURE 8: CREATING MACHINE-SPECIFIC CONFIGURATION

1. Create a directory for machine-specific configurations:

```
> sudo mkdir -p /srv/tftpboot/boot/config
```

2. For a machine with a MAC address `aa:bb:cc:dd:ee:ff`, create a specific configuration:

```
> sudo mkdir -p /srv/tftpboot/boot/config/aa:bb:cc:dd:ee:ff
```

3. Create the machine-specific `grub.cfg`:

```
> sudo cat > /srv/tftpboot/boot/config/aa:bb:cc:dd:ee:ff/grub.cfg << 'EOF'
# Machine-specific configuration for aa:bb:cc:dd:ee:ff
```

```
set default='SLES-16.0 Local Installation'

# Activate the menu-entry after 5sec timeout
set timeout=5

# Use know predictable network interface name
set ipcfg="ip=eno1:dhcp"

# Set the autoinstall variable for this machine
set autoinstall="inst.auto=http://pxe.example.net/install/profiles/
aa:bb:cc:dd:ee:ff/sles16.json"
export autoinstall

# Load the default menu
source "/boot/grub2/menu.cfg"
EOF
```

Alternatively, provide your own menu entry in the host-specific `grub.cfg`, for example generated for a specific boot attempt:

```
> sudo cat > /srv/tftpboot/boot/config/aa:bb:cc:dd:ee:ff/grub.cfg << 'EOF'
set default='SLES-16.0 Auto-Installation'
set timeout=5

menuentry 'SLES-16.0 Auto-Installation' {
  linux  /boot/images/SLES-16.0/${arch}/linux showopts root=live:http://
pxe.example.net/boot/images/SLES-16.0/${arch}/squashfs.img inst.install_url=http://
pxe.example.net/install/SLES-16.0/${arch} inst.auto=http://pxe.example.net/install/
profiles/${net_default_mac}/sles16.json ip=eno1:dhcp
  initrd /boot/images/SLES-16.0/${arch}/initrd
}
EOF
```

EXAMPLE 2: COMMON MACHINE-SPECIFIC PARAMETERS

`default`
: Specifies which menu entry to boot automatically.

`timeout`
: Sets the boot timeout in seconds.

`ipcfg`
: Overrides network interface configuration for specific hardware.

`autoinstall`

Provides machine-specific automated installation profile URLs, using the `inst.auto` kernel parameter. For a full list of `inst.*` and other Agama boot parameters, including which are mandatory, their exact syntax, and defaults, refer to Agama boot options (https://documentation.suse.com/sles/16.1/html/SLES-x86-64-agama-automated-installation/index.html#agama-boot-options).

6.6 Verifying GRUB 2 configuration

After creating the configuration files, verify that the setup is correct, and all required files are in place.

PROCEDURE 9: VERIFICATION STEPS

1. Check the GRUB 2 directory structure:

```
> find /srv/tftpboot/boot/grub2 -type f -name "*.cfg" -o -name "*.efi" -o -name
 "core.*"
```

2. Verify the configuration file syntax by testing with GRUB 2 tools:

```
> grub2-script-check /srv/tftpboot/boot/grub2/grub.cfg
```

```
> grub2-script-check /srv/tftpboot/boot/grub2/menu.cfg
```

3. Ensure proper file permissions:

```
> sudo chmod -R 644 /srv/tftpboot/boot/grub2/*.cfg
```

```
> sudo find /srv/tftpboot/boot/grub2 -type d -exec chmod 0755 {} \;
```

Important: Configuration testing

Test the GRUB 2 configuration with actual PXE clients to ensure proper architecture detection and menu functionality. The `${net_default_mac}` variable is only available during actual network boot scenarios.

6.7 Troubleshooting GRUB 2 configuration

Common issues and their solutions when working with GRUB 2 PXE configurations. Each problem includes diagnostic steps and specific commands to resolve the issue.

6.7.1 Architecture detection fails

When GRUB 2 fails to detect the correct architecture, clients may boot with incorrect binaries or fail to load altogether.

PROCEDURE 10: DEBUGGING ARCHITECTURE DETECTION

1. Add debug output to the GRUB 2 configuration to see detected values:

```
> sudo cat >> /srv/tftpboot/boot/grub2/grub.cfg << 'EOF'
# Debug architecture detection
echo "Detected grub_cpu: ${grub_cpu}"
echo "Mapped arch: ${arch}"
sleep 3
EOF
```

2. Test the configuration syntax:

```
> grub2-script-check /srv/tftpboot/boot/grub2/grub.cfg
```

3. If architecture mapping is incomplete, extend the detection logic:

```
> sudo sed -i '/elif \[ "$grub_cpu" == "powerpc" \]/a\\nelif [ "$grub_cpu" ==
 "riscv64" ]; then\n  set arch='\''riscv64'\''\\' /srv/tftpboot/boot/grub2/grub.cfg
```

4. Verify the architecture-specific directories exist:

```
> ls -la /srv/tftpboot/boot/grub2/
```

6.7.2 Network interface not found

Some firmware implementations may not set the `${net_default_mac}` variable correctly, causing network configuration failures.

PROCEDURE 11: DIAGNOSING NETWORK INTERFACE ISSUES

1. Add debug output to check network variables:

```
> sudo sed -i '/set ipcfg=/i\\necho "Default MAC: ${net_default_mac}"\necho "Network
 variables set"\nsleep 2' /srv/tftpboot/boot/grub2/grub.cfg
```

2. Create a fallback network configuration:

```
> sudo cat >> /srv/tftpboot/boot/grub2/grub.cfg << 'EOF'
```

```
# Fallback network configuration if net_default_mac is empty
if [ "X${net_default_mac}" == "X" ]; then
  set ipcfg="ip=dhcp"
  set ifcfg="ifcfg=*=dhcp"
  echo "WARNING: Using fallback network configuration"
  sleep 2
fi
EOF
```

3. Test network configuration with a specific interface:

```
> sudo echo 'set ipcfg="ip=eno1:dhcp"' > /srv/tftpboot/boot/config/test-network.cfg
```

4. Verify network interface names on the target system:

```
> ip link show
```

6.7.3 File paths not found

Incorrect file paths prevent GRUB 2 from loading kernel and initrd files, causing boot failures.

PROCEDURE 12: VERIFYING FILE PATH ACCESSIBILITY

1. Check if the installer files exist in the expected locations:

```
> find /srv/tftpboot/boot/images -name "linux" -o -name "initrd" -o -name
 "squashfs.img"
```

2. Verify TFTP access to boot files:

```
> tftp localhost -c get /boot/grub2/grub.cfg /tmp/test-grub.cfg
```

3. Test HTTP access to installer files:

```
> curl -I http://localhost/boot/images/SLES-16.0/x86_64/linux
```

4. Check file permissions and ownership:

```
> ls -la /srv/tftpboot/boot/images/SLES-16.0/*/
```

5. Fix permissions if needed:

```
> sudo chmod -R 644 /srv/tftpboot/boot/images/
```

```
> sudo find /srv/tftpboot/boot/images/ -type d -exec chmod 755 {} \;
```

6. Verify symbolic links are not broken:

```
> find /srv/tftpboot/boot/images/ -type l -exec ls -la {} \;
```

6.7.4 EFI boot failures

EFI and Secure Boot issues can prevent proper boot loader initialization or cause authentication failures.

PROCEDURE 13: DIAGNOSING EFI BOOT PROBLEMS

1. Verify Secure Boot files are present:

```
> ls -la /srv/tftpboot/boot/grub2/x86_64-efi/*.efi
```

2. Check that the shim file (either `bootx64.efi` or `shim.efi`), `grub.efi` and `MokManager.efi` are properly copied:

```
> file /srv/tftpboot/boot/grub2/x86_64-efi/bootx64.efi
```

3. Verify EFI file integrity:

```
> sha256sum /srv/tftpboot/boot/grub2/x86_64-efi/*.efi
```

4. Test if files are accessible via TFTP:

```
> tftp localhost -c get /boot/grub2/x86_64-efi/bootx64.efi /tmp/test-shim.efi
```

5. For aarch64 systems, verify ARM64 EFI files:

```
> ls -la /srv/tftpboot/boot/grub2/arm64-efi/*.efi
```

6. Check that the DHCP configuration provides the correct boot loader paths:

```
> grep -n "bootx64.efi\|shim.efi\|bootaa64.efi"
/etc/dnsmasq.d/dhcp.conf /etc/kea/kea-dhcp?.conf /etc/dhcpd?.conf
```

7. If files are missing, recopy them from the ISO mounted at `/mnt` or from the `shim` package files:

```
> sudo cp -v /mnt/EFI/BOOT/*.efi /srv/tftpboot/boot/grub2/x86_64-efi/
```

```
> sudo cp -pL /usr/share/efi/x86_64/*.efi /srv/tftpboot/boot/grub2/x86_64-efi/
```

6.7.5 Menu entries not loading

When GRUB 2 loads but menu entries fail or display errors, the issue is often related to variable expansion or file references.

PROCEDURE 14: DEBUGGING MENU ENTRY PROBLEMS

1. Test menu configuration syntax:

```
> grub2-script-check /srv/tftpboot/boot/grub2/menu.cfg
```

2. Add debug output to menu entries:

```
> sudo sed -i '/linux_kernel.*{images}/i\\necho "Loading: ${images}/SLES-16.0/
${arch}/linux"\necho "Architecture: ${arch}"' /srv/tftpboot/boot/grub2/menu.cfg
```

3. Verify variable expansion works correctly:

```
> sudo cat > /srv/tftpboot/boot/grub2/debug-menu.cfg << 'EOF'
menuentry 'Debug Variables' {
  echo "arch = ${arch}"
  echo "images = ${images}"
  echo "ipcfg = ${ipcfg}"
  sleep 5
}
EOF
```

4. Test with a simplified menu entry:

```
> sudo cat > /srv/tftpboot/boot/grub2/simple-menu.cfg << 'EOF'
menuentry 'Simple Test' {
  linux /boot/images/SLES-16.0/x86_64/linux
  initrd /boot/images/SLES-16.0/x86_64/initrd
}
EOF
```

5. Load the test menu temporarily:

```
> sudo sed -i 's|source "${prefix}/menu.cfg"|source "${prefix}/simple-menu.cfg"|' /
srv/tftpboot/boot/grub2/grub.cfg
```

6. Restore the original menu after testing:

```
> sudo sed -i 's|source "${prefix}/simple-menu.cfg"|source "${prefix}/menu.cfg"|' /
srv/tftpboot/boot/grub2/grub.cfg
```

6.7.6 Enabling detailed logging

For persistent issues, enable logging to capture detailed information about the boot process.

PROCEDURE 15: SETTING UP GRUB 2 DEBUG LOGGING

1. Create a debug version of the main configuration:

```
> sudo cp /srv/tftpboot/boot/grub2/grub.cfg /srv/tftpboot/boot/grub2/grub.cfg.backup
```

2. Add debug output:

```
> sudo cat > /srv/tftpboot/boot/grub2/debug.cfg << 'EOF'
# Debug configuration for GRUB troubleshooting
set debug=all
set pager=1

echo "=== GRUB Debug Information ==="
echo "grub_cpu: ${grub_cpu}"
echo "grub_platform: ${grub_platform}"
echo "net_default_mac: ${net_default_mac}"
echo "net_default_server: ${net_default_server}"
echo "============================="
sleep 5
EOF
```

3. Include debug configuration in main file:

```
> sudo sed -i '1i\source "${prefix}/debug.cfg"' /srv/tftpboot/boot/grub2/grub.cfg
```

4. Monitor TFTP logs during boot attempts:

```
> sudo journalctl -f -u tftp.socket
```

5. Monitor DHCP logs for PXE requests:

```
> sudo journalctl -f -u dhcpd
```

6. Disable debug mode after troubleshooting:

```
> sudo sed -i '/source "${prefix}\/debug.cfg"/d' /srv/tftpboot/boot/grub2/grub.cfg
```

6.8 Multi-environment subdirectory setups

If you want to serve multiple independent PXE environments from subdirectories under one TFTP root (for example, `/srv/tftpboot/ENV1/boot/...` and `/srv/tftpboot/ENV2/boot/...`), the standard `/boot` prefix used throughout this guide must be replaced with your custom prefix consistently in all of the following places. Missing even one will cause boot failures.

- **DHCP boot file name:** the boot filename delivered to the client must include the custom prefix. For example, `/ENV1/boot/grub2/x86_64-efi/bootx64.efi`.
- **nginx location alias:** update the `location /boot` block in `/etc/nginx/nginx.conf` to expose the custom prefix path.
- `grub.cfg` **source paths:** update the hardcoded absolute paths inside `/srv/tftpboot/boot/grub2/grub.cfg`:

  ```
  source "${config}/${net_default_mac}/grub.cfg"
  ```

  ```
  source "${prefix}/menu.cfg"
  ```

  Both must use the custom prefix.
- `menu.cfg` **paths:** update all `/boot/images/...` paths inside `menu.cfg` to use the custom prefix.
- **Per-MAC host configurations:** if you use machine-specific configurations, also update the `source "/boot/grub2/menu.cfg"` line inside each per-MAC `grub.cfg` to use the custom prefix.
- **SELinux file context:** run **`restorecon`** on your custom path:

  ```
  > sudo restorecon -Rv /srv/tftpboot/ENV1
  ```
- **SELinux policy:** running **`setsebool -P httpd_serve_cobbler_files=1`** only covers the standard `/boot` path. A custom SELinux policy is required for a non-standard prefix. Use **`ausearch -m avc -ts recent`** to identify denials and create the appropriate policy module.

6.9 Next steps: Completing the PXE setup

GRUB 2 is now configured. The remaining tasks are:

- Configure HTTP and TFTP services to serve the boot files and installer content.
- Set up DHCP services to direct PXE clients to the appropriate boot loaders.
- Test the complete PXE boot process on target hardware.

7 Configuring TFTP for PXE booting

This section explains how to configure TFTP services to serve GRUB 2 boot loaders and PXE boot content for SUSE Linux Enterprise Server 16.1 installations. It covers the traditional `in.tftpd` server and the integrated TFTP functionality provided by `dnsmasq`.

7.1 The role of TFTP in PXE boot

TFTP serves boot loader files to PXE clients during the network boot process. SUSE Linux Enterprise Server 16.1 supports two TFTP server implementations: the traditional `in.tftpd` server from the `tftp` package and the integrated TFTP functionality within `dnsmasq`.

7.2 Requirements for configuring TFTP for PXE boot

- Either the `tftp` package or the `dnsmasq` package installed.
- PXE boot files organized under `/srv/tftpboot`.
- Administrative privileges to configure services.

7.3 Configuring the in.tftpd server

The `in.tftpd` server uses the configuration file `/etc/sysconfig/tftp` to define the TFTP root directory and server options.

PROCEDURE 16: SETTING UP IN.TFTPD TFTP SERVER

1. Optionally, enable verbose logging by setting TFTP options:

```
> sudo sed -i 's/^TFTP_OPTIONS=.*/TFTP_OPTIONS="-v"/' /etc/sysconfig/tftp
```

   The `-v` option enables verbose logging to see the file names fetched via TFTP.

2. Enable and start the TFTP service:

```
> sudo systemctl enable --now tftp.service
```

7.4 Configuring dnsmasq as a TFTP server

`dnsmasq` provides a built-in TFTP server that can be enabled and configured to use the `/srv/tftpboot` directory.

PROCEDURE 17: SETTING UP DNSMASQ TFTP FUNCTIONALITY

1. Create the TFTP configuration file:

```
> sudo cat > /etc/dnsmasq.d/tftp.conf << 'EOF'
enable-tftp
tftp-root=/srv/tftpboot
EOF
```

2. Enable and start the dnsmasq service:

```
> sudo systemctl enable --now dnsmasq
```

7.5 Verifying TFTP server configuration

Test the TFTP server functionality to ensure it can serve files to PXE clients.

PROCEDURE 18: TESTING TFTP SERVER FUNCTIONALITY

1. Create a test file:

```
> echo "test file" | sudo tee /srv/tftpboot/test.txt
```

2. Retrieve the test file via TFTP:

```
> tftp localhost -c get test.txt /tmp/tftp-test.txt
```

3. Verify the file was retrieved successfully:

```
> cat /tmp/tftp-test.txt
```

4. Clean up test files:

```
> sudo rm /srv/tftpboot/test.txt /tmp/tftp-test.txt
```

7.6 Troubleshooting TFTP configuration

Common issues when configuring TFTP services for PXE boot environments.

7.6.1 Resolving TFTP service conflicts on port 69

Both `in.tftpd` and `dnsmasq` use UDP port 69 for TFTP services and cannot run simultaneously.

PROCEDURE 19: RESOLVING TFTP SERVICE CONFLICTS

1. Check which services are running:

```
> systemctl status tftp.service dnsmasq
```

2. Check what is using port 69:

```
> ss -ulnp | grep :69
```

3. Stop the conflicting service (example for dnsmasq):

```
> sudo systemctl stop dnsmasq
```

4. Start your preferred TFTP service:

```
> sudo systemctl start tftp.service
```

7.6.2 Troubleshooting TFTP directory access issues

Problems accessing the TFTP root directory can prevent file serving.

PROCEDURE 20: VERIFYING TFTP DIRECTORY CONFIGURATION

1. Verify the TFTP directory setting for in.tftpd:

```
> grep TFTP_DIRECTORY /etc/sysconfig/tftp
```

2. Verify the TFTP directory setting for dnsmasq:

```
> grep tftp-root /etc/dnsmasq.d/tftp.conf
```

3. Check if the directory exists:

```
> ls -la /srv/tftpboot/
```

4. Create the directory if missing:

```
> sudo mkdir -p /srv/tftpboot
```

7.6.3 Enabling TFTP logging

Verbose logging helps identify file access issues with TFTP transfers.

PROCEDURE 21: ENABLING VERBOSE TFTP LOGGING

1. Check the current TFTP options:

```
> grep TFTP_OPTIONS /etc/sysconfig/tftp
```

2. Enable verbose logging:

```
> sudo sed -i 's/^TFTP_OPTIONS=.*/TFTP_OPTIONS="-v"/' /etc/sysconfig/tftp
```

3. Restart the TFTP service:

```
> sudo systemctl restart tftp.service
```

4. Monitor TFTP logs:

```
> journalctl -u tftp.service -f
```

7.7 Next steps

TFTP is now configured. Next, set up HTTP services to serve installer files and DHCP services to direct PXE clients to the appropriate boot loaders.

8 Configuring nginx for HTTP delivery

This section explains how to configure nginx to serve PXE boot content over HTTP, allowing clients to load installer files like kernel, initrd and squashfs images from a central location. HTTP delivery provides better performance than TFTP for large files and is required for SUSE Linux Enterprise Server 16.1 Agama installations.

8.1 The role of nginx in PXE boot HTTP delivery

nginx serves as the HTTP server for PXE boot environments, providing access to installer files through Web-based delivery. The HTTP server exposes the TFTP boot directory and installation repositories, enabling PXE clients to download kernel images, initrd files and the Agama installer components over HTTP rather than the slower TFTP protocol.

8.2 Requirements for configuring nginx for PXE boot

- The `nginx` package installed.
- PXE boot files organized under `/srv/tftpboot/boot`.
- Installation repositories available under `/srv/www/htdocs/install`.
- Administrative privileges to modify nginx configuration.

8.3 Configuring nginx for PXE boot

The nginx configuration exposes the TFTP boot directory through a dedicated location alias and serves installation repositories directly from the document root.

PROCEDURE 22: SETTING UP NGINX HTTP SERVER

1. Edit the nginx configuration file:

```
> sudo vim /etc/nginx/nginx.conf
```

2. Configure the HTTP server block in the `http` section:

```
> sudo cat > /etc/nginx/nginx.conf << 'EOF'
```

```
http {

    include             mime.types;
    default_type        application/octet-stream;

    charset             utf-8;
    sendfile            on;
    keepalive_timeout   65;

    server {
        listen          80  default_server;
        listen     [::]:80  default_server;

        location / {
            root   /srv/www/htdocs/;
            index  index.html index.htm;
        }

        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   /srv/www/htdocs/;
        }

        # Expose TFTP boot directory for HTTP boot
        location /boot {
            alias      /srv/tftpboot/boot;
            autoindex  on;
        }
    }
}

events {
    worker_connections  1024;
}
EOF
```

3. Test the nginx configuration syntax:

```
> sudo nginx -t
```

4. Allow nginx to serve files from the directory:

```
> sudo setsebool -P httpd_serve_cobbler_files=1
```

5. Enable and start the nginx service:

```
> sudo systemctl enable --now nginx.service
```

8.4 Verifying nginx HTTP access

Test the HTTP server functionality to ensure it can serve PXE boot files and installation content to clients.

PROCEDURE 23: TESTING NGINX HTTP SERVER

1. Test HTTP access to boot files:

```
> curl -I http://localhost/boot/
```

2. Test access to the installation directory:

```
> curl -I http://localhost/install/
```

3. Verify a specific installer file is accessible:

   - For AMD64/Intel 64:

```
> curl -I http://localhost/boot/images/SLES-16.0/x86_64/squashfs.img
```

   - For AArch64:

```
> curl -I http://localhost/boot/images/SLES-16.0/aarch64/squashfs.img
```

   - For ppc64le:

```
> curl -I http://localhost/boot/images/SLES-16.0/ppc64le/squashfs.img
```

8.5 Troubleshooting nginx HTTP delivery issues for PXE boot

This section describes common issues when configuring nginx for PXE boot HTTP delivery.

8.5.1 Configuration syntax errors

Incorrect nginx configuration syntax prevents the service from starting or reloading properly.

PROCEDURE 24: RESOLVING NGINX CONFIGURATION ISSUES

1. Test the configuration syntax:

```
> sudo nginx -t
```

2. Check nginx service status if startup fails:

```
> systemctl status nginx.service
```

3. View detailed error logs:

```
> journalctl -u nginx.service -f
```

4. Check nginx error log file:

```
> tail -f /var/log/nginx/error.log
```

8.5.2 File access and permission issues

nginx may fail to serve files due to incorrect permissions or missing directories.

PROCEDURE 25: RESOLVING FILE ACCESS PROBLEMS

1. Check if the boot directory exists and is accessible:

```
> ls -la /srv/tftpboot/boot/
```

2. Check if the install directory exists:

```
> ls -la /srv/www/htdocs/install/
```

3. Verify nginx can read the directories:

```
> sudo -u nginx ls /srv/tftpboot/boot/
```

4. Create missing directories if needed:

```
> sudo mkdir -p /srv/www/htdocs/install
```

5. Set appropriate permissions:

```
> sudo chmod -R 755 /srv/tftpboot/boot /srv/www/htdocs/install
```

6. If nginx still cannot serve files from `/srv/tftpboot/boot`, restore the SELinux file context after placing or moving files into these directories:

```
> sudo restorecon -Rv /srv/www/htdocs/install
```

```
> sudo restorecon -Rv /srv/tftpboot/boot
```



Note: Verifying SELinux access

This step is needed in addition to **`setsebool -P httpd_serve_cobbler_files=1`**. Run **`ausearch -m avc -ts recent`** to check for SELinux denials; the command should produce no output once access is set up correctly. Do not work around remaining denials with **`audit2allow`**. Instead, check the file context with **`ls -lZ`** or **`matchpathcon`** and confirm the required booleans are enabled.

8.5.3 Port binding conflicts

nginx may fail to start if another service is using port 80.

PROCEDURE 26: RESOLVING PORT CONFLICTS

1. Check what is using port 80:

   ```
   > ss -tlnp | grep :80
   ```

2. Stop conflicting services if needed:

   ```
   > sudo systemctl stop apache2
   ```

3. Start nginx service:

   ```
   > sudo systemctl start nginx.service
   ```

4. Verify nginx is listening on port 80:

   ```
   > ss -tlnp | grep :80
   ```

8.6 Next steps

nginx is now ready to deliver boot files and installer content over HTTP. Next, configure DHCP services to direct PXE clients to the appropriate boot loaders and HTTP resources.

9 Configuring a DNS server using dnsmasq

This section explains how to configure DNS services using dnsmasq to provide host name resolution for PXE clients accessing SUSE Linux Enterprise Server 16.1 installation resources. DNS configuration enables clients to use host names instead of IP addresses in boot URLs and DHCP configurations.

9.1 The role of DNS in PXE boot environments

DNS services enable PXE clients to resolve host names in boot URLs and installation sources. While full DNS server configuration is beyond the scope of this document, this section provides a basic DNS configuration using dnsmasq that allows clients to resolve the PXE server host name (pxe.example.net) to its IP address.

Without DNS configuration, boot URLs must use IP addresses directly, such as `http://192.168.1.200/` or `http://[2001:db8:0:1::200]/`. Some BIOS/UEFI firmware implementations do not support host names in DHCP TFTP URLs and require IP addresses like `tftp://[2001:db8:0:1::200]/`.

9.2 Requirements for configuring dnsmasq DNS services

- The `dnsmasq` package installed.
- Static IP addresses configured for the PXE server.
- Administrative privileges to configure DNS services.

9.3 Configuring dnsmasq for PXE client host name resolution

The dnsmasq DNS configuration provides local host name resolution and uses upstream name servers for external queries.

PROCEDURE 27: SETTING UP DNSMASQ DNS SERVER

1. Create the DNS configuration file for dnsmasq:

```
> sudo cat > /etc/dnsmasq.d/dns.conf << 'EOF'
# DNS configuration file for dnsmasq
```

```
# Log DNS queries
log-queries

# DNS cache behavior
cache-size=10000
local-ttl=60
neg-ttl=10

# Never forward A or AAAA queries for plain names to upstream name servers
domain-needed

# Add local domain to simple names in /etc/hosts and DHCP
expand-hosts

# Specifies DNS domain and networks including local forward and reverse declarations
domain=example.net,192.168.1.0/24,local
domain=example.net,2001:db8:0:1::/64,local
EOF
```

2. Add host name entries to the system hosts file:

```
> sudo cat >> /etc/hosts << 'EOF'
192.168.1.200 pxe.example.net
2001:db8:0:1::200 pxe.example.net
EOF
```

3. Test the dnsmasq configuration:

```
> sudo dnsmasq --test
```

4. Enable and start the dnsmasq service:

```
> sudo systemctl enable --now dnsmasq
```



Note: DNS forwarding behavior

By default, dnsmasq uses the name servers in `/etc/resolv.conf` as forwarders and provides records from `/etc/hosts`. This allows the PXE server to resolve external host names while providing local resolution for PXE-related services.

9.4 Verifying DNS configuration

Test the DNS server functionality to ensure host name resolution works for PXE clients.

PROCEDURE 28: TESTING DNS SERVER FUNCTIONALITY

1. Test IPv4 host name resolution:

```
> nslookup pxe.example.net localhost
```

2. Test IPv6 host name resolution:

```
> nslookup pxe.example.net localhost | grep 2001:db8
```

3. Test reverse DNS lookup for IPv4:

```
> nslookup 192.168.1.200 localhost
```

4. Verify external DNS forwarding still works:

```
> nslookup google.com localhost
```

9.5 Troubleshooting dnsmasq DNS issues

This section describes common issues when configuring dnsmasq for DNS services in PXE environments.

9.5.1 dnsmasq configuration and service issues

dnsmasq may fail to start due to configuration errors or port conflicts.

PROCEDURE 29: RESOLVING DNS CONFIGURATION PROBLEMS

1. Test the dnsmasq configuration syntax:

```
> sudo dnsmasq --test
```

2. Check dnsmasq service status:

```
> systemctl status dnsmasq
```

3. Check what is using DNS port 53:

```
> ss -ulnp | grep :53
```

4. View dnsmasq logs for errors:

```
> journalctl -u dnsmasq -f
```

5. Stop conflicting DNS services if needed:

```
> sudo systemctl stop systemd-resolved
```

9.5.2 PXE client host name resolution failures

DNS queries may fail due to incorrect configuration or missing host name entries.

PROCEDURE 30: DIAGNOSING DNS RESOLUTION ISSUES

1. Check if host name entries exist in the hosts file:

```
> grep pxe.example.net /etc/hosts
```

2. Verify domain configuration in dnsmasq:

```
> grep domain= /etc/dnsmasq.d/dns.conf
```

3. Test DNS query with verbose output:

```
> dig @localhost pxe.example.net
```

4. Monitor dnsmasq query logs:

```
> journalctl -u dnsmasq | grep "query"
```

5. Restart dnsmasq to reload the configuration:

```
> sudo systemctl restart dnsmasq
```

9.5.3 dnsmasq DNS forwarding problems

External DNS queries may fail if the upstream name server configuration is incorrect.

PROCEDURE 31: TROUBLESHOOTING DNS FORWARDING ISSUES

1. Check the upstream name server configuration:

```
> cat /etc/resolv.conf
```

2. Test direct query to the upstream name server:

```
> nslookup google.com 8.8.8.8
```

3. Check dnsmasq forwarding configuration:

```
> grep -E "server=|no-resolv" /etc/dnsmasq.d/dns.conf
```

4. Add a specific upstream name server if needed:

```
> sudo echo "server=8.8.8.8" >> /etc/dnsmasq.d/dns.conf
```

5. Restart the dnsmasq service:

```
> sudo systemctl restart dnsmasq
```

9.6 Next steps

DNS is now configured, and PXE clients can resolve host names in boot URLs and installation sources. Next, configure DHCP services that reference this DNS server for client configuration.

10 Configuring an NTP server using chrony

This section explains how to configure NTP services using `chrony` to keep the PXE server's clock accurate and, where needed, provide time synchronization to PXE clients during SUSE Linux Enterprise Server 16.1 installations. Proper time synchronization avoids certificate validation errors and keeps system logs accurately timestamped during network-based installations.

10.1 The role of NTP time synchronization in PXE boot

Synchronized time avoids certificate validation errors during HTTPS connections and keeps log timestamps accurate, though NTP is not strictly required for a PXE deployment to work. Whether the PXE server itself needs to serve time to PXE clients depends on your network:

- If PXE clients can already reach an NTP source other than the PXE server, such as a gateway or router operated separately from it (the setup used throughout the rest of this article, where the sample network's NTP server is the gateway at `192.168.1.1`), you only need `chrony` to keep the PXE server's own clock accurate. PXE clients get their NTP server address from that external source via DHCP, not from the PXE server.
- If the PXE server also acts as the network's gateway, for example in an air-gapped or isolated test network with no other NTP source reachable by clients, configure `chrony` on the PXE server to also serve time to those clients, as shown below.

10.2 Requirements for configuring chrony NTP services

- The `chrony` package installed.

```
> sudo zypper install chrony
```

- Network connectivity to upstream NTP servers.
- Administrative privileges to configure NTP services.

10.3 Configuring chrony NTP service

The `chrony` service provides NTP functionality with automatic time synchronization to upstream servers and local time-serving capabilities for network clients.

PROCEDURE 32: SETTING UP `chrony` NTP SERVER

1. Enable and start the `chrony` service:

```
> sudo systemctl enable --now chronyd.service
```

2. Optionally, allow PXE clients to sync time from this server if they cannot reach an NTP source other than the PXE server itself, for example because the PXE server also acts as the network's gateway:

```
> sudo cat >> /etc/chrony.conf << 'EOF'
```

```
# Allow NTP clients on the PXE network to sync from this server
allow 192.168.1.0/24
allow 2001:db8:0:1::/64
EOF
```

Restart `chrony` to apply the change:

```
> sudo systemctl restart chronyd.service
```

10.4 Verifying NTP configuration

Test the NTP service functionality to ensure time synchronization works correctly.

PROCEDURE 33: TESTING NTP SERVER FUNCTIONALITY

1. Check `chrony` service status:

   ```
   > systemctl status chronyd.service
   ```

2. View the current time synchronization status:

   ```
   > chronyc tracking
   ```

3. List configured NTP sources:

   ```
   > chronyc sources
   ```

4. Check NTP server statistics:

   ```
   > chronyc sourcestats
   ```

5. Optionally, if you configured `chrony` to serve PXE clients, verify the `allow` directive is in place:

   ```
   > grep allow /etc/chrony.conf
   ```

10.5 Troubleshooting NTP configuration

This section describes common issues when configuring `chrony` for NTP services in PXE environments.

10.5.1 chrony service startup issues

`chrony` service may fail to start due to configuration errors or network connectivity problems.

PROCEDURE 34: RESOLVING NTP SERVICE PROBLEMS

1. Check `chrony` service status and logs:

```
> systemctl status chronyd.service
```

2. View detailed service logs:

```
> journalctl -u chronyd.service -f
```

3. Test `chrony` configuration:

```
> sudo chronyd -Q
```

4. Restart the service if needed:

```
> sudo systemctl restart chronyd.service
```

10.5.2 Time synchronization failures

Time synchronization may fail due to network issues or incorrect server configuration.

PROCEDURE 35: DIAGNOSING TIME SYNCHRONIZATION ISSUES

1. Check the current synchronization status:

```
> chronyc tracking
```

2. View NTP source connectivity:

```
> chronyc sources -v
```

3. Force immediate synchronization:

```
> sudo chronyc makestep
```

4. Check system time against hardware clock:

```
> timedatectl status
```

5. Verify network connectivity to NTP servers:

```
> chronyc activity
```

10.5.3 Firewall and network issues

NTP traffic may be blocked by firewall rules, preventing time synchronization.

PROCEDURE 36: RESOLVING NTP NETWORK CONNECTIVITY

1. Check if the NTP port is open in the firewall:

```
> firewall-cmd --list-services | grep ntp
```

2. Add NTP service to firewall if needed:

```
> sudo firewall-cmd --permanent --add-service=ntp
```

3. Reload firewall configuration:

```
> sudo firewall-cmd --reload
```

4. Test NTP connectivity manually:

```
> ntpdate -q pool.ntp.org
```

5. Check `chrony` port usage:

```
> ss -ulnp | grep :123
```

10.6 Next steps

NTP is now configured. Continue with the remaining PXE services needed for your deployment, such as DHCP.

11 Configuring IPv6 router advertisement

This section describes how to configure IPv6 router advertisement functionality to provide adequate router advertisements for PXE clients. IPv6 RA enables IPv6 routing configuration and stateful DHCPv6 address automatic configuration for SUSE Linux Enterprise Server 16.1 installations.

11.1 The role of IPv6 router advertisement in PXE boot

IPv6 router advertisement (RA) provides network configuration information to PXE clients, including IPv6 routing and DHCPv6 address automatic configuration settings. If your network does not already have an IPv6 router providing adequate router advertisements, this section shows how to configure the PXE server to provide them: IPv6 routing to the network and default route, and stateful DHCPv6 address automatic configuration using `AdvManagedFlag on`.

11.2 Requirements for configuring IPv6 router advertisement

- The `radvd` package installed.
- IPv6 network configuration on the server interface.
- Administrative privileges to configure router advertisement services.

11.3 Configuring `radvd` for IPv6 router advertisement

The `radvd` service provides IPv6 router advertisement functionality using configuration defined in `/etc/radvd.conf`.

PROCEDURE 37: SETTING UP `radvd` IPV6 ROUTER ADVERTISEMENT

1. Configure the `radvd` service:

```
> sudo cat > /etc/radvd.conf << 'EOF'
interface eno1
{
    # radvd options
    IgnoreIfMissing on;                # Do not fail and exit when interface is
 missed
    AdvSendAdvert on;                  # Sending RAs on the interface is not
 disabled

    # Configuration settings

    AdvManagedFlag on;                 # Request IPv6 address and dns options via
 DHCPv6
    AdvOtherConfigFlag off;            # Request only dns info via DHCPv6, IP via
 SLAAC

    AdvDefaultLifetime 1800;           # Add default route via this router for
 1800sec
```

```
    prefix 2001:db8:0:1::/64            # Add direct route for this local network/
prefix
    {
        AdvAutonomous           off;    # Assign IPv6 address via SLAAC
        AdvValidLifetime        7200;
        AdvPreferredLifetime    3600;
    };
};
EOF
```

2. Enable and start the `radvd` service:

```
> sudo systemctl enable --now radvd
```

11.4 Verifying IPv6 router advertisement

Test the IPv6 RA functionality to ensure proper configuration and operation.

PROCEDURE 38: TESTING IPV6 ROUTER ADVERTISEMENT

1. Check `radvd` service status:

```
> systemctl status radvd
```

2. Review and verify IPv6 RA settings using **`radvdump`**:

```
> radvdump
```

The **`radvdump`** utility displays IPv6 RA settings sent by the IPv6 router every few minutes.

11.5 Configuring IP forwarding for router functionality

If the PXE server also acts as a router, IP forwarding must be enabled to allow the system to function in a router role.

PROCEDURE 39: ENABLING IP FORWARDING ON THE PXE SERVER

1. Create the network configuration file:

```
> sudo cat > /etc/sysctl.d/90-network.conf << 'EOF'
# This machine is a router
net.ipv4.conf.all.forwarding = 1
net.ipv6.conf.all.forwarding = 1
```

```
# Accept host autoconf on the router's uplink interface
net.ipv6.conf.UPLINK.accept_ra = 2
EOF
```

2. Apply the network configuration settings:

```
> sudo sysctl -p /etc/sysctl.d/90-network.conf
```



Note: Router configuration considerations

A router does not process IPv6 RAs for host automatic configuration by default. To accept IPv6 RA on the router's uplink interface, the `accept_ra = 2` sysctl setting is required; replace *`UPLINK`* with the name of that interface. Consult the Network Configuration section in the Administration Guide for further details on router configuration, including firewall adjustments and other required steps.

11.6 Troubleshooting IPv6 router advertisement issues

This section describes common issues when configuring IPv6 router advertisement for PXE environments.

11.6.1 `radvd` service issues

The `radvd` service may fail to start due to configuration errors or interface problems.

PROCEDURE 40: RESOLVING `radvd` SERVICE PROBLEMS

1. Check `radvd` service status and logs:

```
> systemctl status radvd
```

2. View detailed service logs:

```
> journalctl -u radvd -f
```

3. Test `radvd` configuration syntax:

```
> sudo radvd -C /etc/radvd.conf
```

4. Check if the specified interface exists:

```
> ip link show eno1
```

5. Restart the service after fixing the configuration:

```
> sudo systemctl restart radvd
```

11.6.2 IP forwarding configuration issues

Incorrect IP forwarding settings can prevent proper router functionality.

PROCEDURE 41: DIAGNOSING IP FORWARDING PROBLEMS

1. Check current IP forwarding status:

```
> sysctl net.ipv4.conf.all.forwarding
```

2. Check IPv6 forwarding status:

```
> sysctl net.ipv6.conf.all.forwarding
```

3. Verify sysctl configuration file:

```
> cat /etc/sysctl.d/90-network.conf
```

4. Apply configuration if values are incorrect:

```
> sudo sysctl -p /etc/sysctl.d/90-network.conf
```

5. Check the `accept_ra` setting on the uplink interface (replace *`UPLINK`* with its name):

```
> sysctl net.ipv6.conf.UPLINK.accept_ra
```

11.6.3 Router advertisement reception issues

Clients may not receive or process IPv6 router advertisements correctly.

PROCEDURE 42: TROUBLESHOOTING RA RECEPTION PROBLEMS

1. Monitor router advertisements using **`radvdump`**:

```
> radvdump -d
```

2. Check IPv6 interface configuration on clients:

```
> ip -6 addr show
```

3. Verify IPv6 routing table on clients:

```
> ip -6 route show
```

4. Test IPv6 connectivity to the router:

```
> ping6 2001:db8:0:1::1
```

5. Check firewall rules for ICMPv6:

```
> firewall-cmd --list-protocols | grep ipv6-icmp
```

11.7 Next steps

IPv6 router advertisement is now configured, giving PXE clients the network information they need for DHCPv6 and IPv6 connectivity during network-based installations.

12 Configuring a DHCP server using dnsmasq

This section explains how to configure DHCP services using dnsmasq to provide network configuration and PXE boot information for SUSE Linux Enterprise Server 16.1 installations. The dnsmasq DHCP server uses tag-based configuration to support both IPv4 and IPv6 PXE clients with UEFI and BIOS boot capabilities.

12.1 Introduction

The dnsmasq DHCP server provides network configuration and boot file information to PXE clients using a tag-based system to match client types and provide appropriate boot loaders. This configuration supports both PXEClient and HTTPClient matches that work for DHCPv4 and DHCPv6, enabling boot via UEFI and BIOS systems across multiple architectures.

Important: HTTPClient limitations in dnsmasq

dnsmasq version 2.90 and earlier does not support sending the vendor-class option6:16 back to DHCPv6 clients for HTTPClient configurations. For full HTTPClient support, consider using Kea or ISC DHCP servers.

12.2 Requirements

- The `dnsmasq` package installed.
- PXE boot files properly organized under `/srv/tftpboot`.
- Network interface configured for DHCP service.
- Administrative privileges to configure DHCP services.

12.3 Configuring dnsmasq DHCP services

The dnsmasq DHCP configuration includes client type matching, network ranges, and boot file assignments for both IPv4 and IPv6 networks.

PROCEDURE 43: SETTING UP DNSMASQ DHCP SERVER

1. Create the DHCP configuration file for dnsmasq:

```
> sudo cat > /etc/dnsmasq.d/dhcp.conf << 'EOF'
# DHCP configuration file for dnsmasq

# Log DHCP processing
log-dhcp

# This is the only DHCP server, don't ignore unknown clients/send NAK
dhcp-authoritative

# Disable re-use of the DHCPv4 servername and filename fields as extra
# option space, which may confuse old or broken clients
dhcp-no-override

# IPv4 PXE/HTTP boot client matches (no enterprise number)
# Match client type in PXEClient:Arch and map to a tag
dhcp-vendorclass=set:tftp_bios_x86_pc,PXEClient:Arch:00000
dhcp-vendorclass=set:tftp_uefi_x86_64,PXEClient:Arch:00007
dhcp-vendorclass=set:tftp_ieee_ppc_64,PXEClient:Arch:0000e
dhcp-vendorclass=set:tftp_uefi_arm_64,PXEClient:Arch:00011
# Match client type in HTTPClient:Arch and map to a tag
dhcp-vendorclass=set:http_uefi_x86_64,HTTPClient:Arch:00016
dhcp-vendorclass=set:http_uefi_arm_64,HTTPClient:Arch:00019

# IPv6 PXE/HTTP boot client matches (enterprise:343 intel)
# Match client type in PXEClient:Arch and map to a tag
dhcp-vendorclass=set:tftp_bios_x86_pc,enterprise:343,PXEClient:Arch:00000
dhcp-vendorclass=set:tftp_uefi_x86_64,enterprise:343,PXEClient:Arch:00007
```

```
dhcp-vendorclass=set:tftp_ieee_ppc_64,enterprise:343,PXEClient:Arch:0000e
dhcp-vendorclass=set:tftp_uefi_arm_64,enterprise:343,PXEClient:Arch:00011
# Match client type in HTTPClient:Arch and map to a tag
dhcp-vendorclass=set:http_uefi_x86_64,enterprise:343,HTTPClient:Arch:00016
dhcp-vendorclass=set:http_uefi_arm_64,enterprise:343,HTTPClient:Arch:00019
EOF
```

2. Configure IPv4 DHCP range and options:

```
> sudo cat >> /etc/dnsmasq.d/dhcp.conf << 'EOF'

# IPv4 range and options
dhcp-range=set:net0v4,192.168.1.100,192.168.1.199,255.255.255.0,1h
dhcp-option=tag:net0v4,option:domain-search,example.net
dhcp-option=tag:net0v4,option:dns-server,192.168.1.200
dhcp-option=tag:net0v4,option:ntp-server,192.168.1.1
dhcp-option=tag:net0v4,option:router,192.168.1.1
EOF
```

3. Configure IPv4 PXE boot options:

```
> sudo cat >> /etc/dnsmasq.d/dhcp.conf << 'EOF'

# IPv4 PXEClient boot
dhcp-boot=tag:net0v4,tag:tftp_bios_x86_pc,/boot/grub2/i386-pc/core.0,192.168.1.200
dhcp-boot=tag:net0v4,tag:tftp_uefi_x86_64,/boot/grub2/x86_64-efi/
bootx64.efi,192.168.1.200
dhcp-boot=tag:net0v4,tag:tftp_ieee_ppc_64,/boot/grub2/powerpc-ieee1275/
core.elf,192.168.1.200
dhcp-boot=tag:net0v4,tag:tftp_uefi_arm_64,/boot/grub2/arm64-efi/
bootaa64.efi,192.168.1.200

# IPv4 HTTPClient boot
dhcp-option-force=tag:net0v4,tag:http_uefi_x86_64,option:vendor-class,HTTPClient
dhcp-boot=tag:net0v4,tag:http_uefi_x86_64,http://192.168.1.200/boot/grub2/x86_64-
efi/bootx64.efi
dhcp-option-force=tag:net0v4,tag:http_uefi_arm_64,option:vendor-class,HTTPClient
dhcp-boot=tag:net0v4,tag:http_uefi_arm_64,http://192.168.1.200/boot/grub2/arm64-efi/
bootaa64.efi
EOF
```

4. Configure IPv6 DHCP range and options:

```
> sudo cat >> /etc/dnsmasq.d/dhcp.conf << 'EOF'

# IPv6 range and options
dhcp-range=set:net0v6,2001:db8:0:1:d::,2001:db8:0:1:d::ffff,64,1h
```

```
dhcp-option=tag:net0v6,option6:domain-search,example.net
dhcp-option=tag:net0v6,option6:dns-server,[2001:db8:0:1::200]
dhcp-option=tag:net0v6,option6:sntp-server,[2001:db8:0:1::1]
EOF
```

5. Configure IPv6 PXE boot options:

```
> sudo cat >> /etc/dnsmasq.d/dhcp.conf << 'EOF'

# IPv6 PXEClient boot
dhcp-option=tag:net0v6,tag:tftp_bios_x86_pc,option6:bootfile-url,tftp://
[2001:db8:0:1::200]/boot/grub2/i386-pc/core.0
dhcp-option=tag:net0v6,tag:tftp_uefi_x86_64,option6:bootfile-url,tftp://
[2001:db8:0:1::200]/boot/grub2/x86_64-efi/bootx64.efi
dhcp-option=tag:net0v6,tag:tftp_ieee_ppc_64,option6:bootfile-url,tftp://
[2001:db8:0:1::200]/boot/grub2/powerpc-ieee1275/core.elf
dhcp-option=tag:net0v6,tag:tftp_uefi_arm_64,option6:bootfile-url,tftp://
[2001:db8:0:1::200]/boot/grub2/arm64-efi/bootaa64.efi

# IPv6 HTTPClient boot
# Note: dnsmasq <= 2.90 does not support sending vendor-class option6:16 back to
 client
EOF
```

6. Test the dnsmasq configuration:

```
> sudo dnsmasq --test
```

7. Enable and start the dnsmasq service:

```
> sudo systemctl enable --now dnsmasq
```



Note: Boot file paths require the optional Secure Boot step

The `bootx64.efi` and `bootaa64.efi` boot file paths above are the signed files from the optional Secure Boot step in *Step 2*. If you did not perform that step, replace them with `core.efi` (still under the respective `x86_64-efi`/`arm64-efi` directory), which works as long as UEFI Secure Boot is disabled on the client.

12.4 Verifying DHCP configuration

Test the DHCP server functionality to ensure proper network configuration and boot file delivery to PXE clients.

PROCEDURE 44: TESTING DNSMASQ DHCP SERVER

1. Check dnsmasq service status:

```
> systemctl status dnsmasq
```

2. Verify DHCP port binding:

```
> ss -ulnp | grep :67
```

3. Monitor DHCP lease assignments:

```
> journalctl -u dnsmasq -f
```

4. Check active DHCP leases:

```
> cat /var/lib/misc/dnsmasq.leases
```

12.5 Troubleshooting dnsmasq DHCP configuration

Common issues when configuring dnsmasq for DHCP services in PXE environments.

12.5.1 Service startup and configuration issues

dnsmasq may fail to start due to configuration errors or port conflicts with other DHCP services.

PROCEDURE 45: RESOLVING DNSMASQ DHCP SERVICE PROBLEMS

1. Test dnsmasq configuration syntax:

```
> sudo dnsmasq --test
```

2. Check for DHCP port conflicts:

```
> ss -ulnp | grep :67
```

3. Stop conflicting DHCP services:

```
> sudo systemctl stop dhcpd
```

4. View detailed service logs:

```
> journalctl -u dnsmasq -f
```

5. Restart dnsmasq after resolving conflicts:

```
> sudo systemctl restart dnsmasq
```

12.5.2 DHCP lease assignment problems

Clients may fail to receive IP addresses due to range configuration or network connectivity issues.

PROCEDURE 46: DIAGNOSING DHCP LEASE ISSUES

1. Check DHCP range configuration:

```
> grep dhcp-range /etc/dnsmasq.d/dhcp.conf
```

2. Monitor DHCP requests in real-time:

```
> journalctl -u dnsmasq -f | grep DHCP
```

3. Check network interface status:

```
> ip addr show
```

4. Verify DHCP authoritative setting:

```
> grep dhcp-authoritative /etc/dnsmasq.d/dhcp.conf
```

5. Test DHCP response with dhcping:

```
> dhcping -s 192.168.1.200
```

12.5.3 PXE boot file delivery issues

PXE clients may receive IP addresses but fail to boot due to incorrect boot file configuration or client type matching problems.

PROCEDURE 47: TROUBLESHOOTING PXE BOOT CONFIGURATION

1. Check client vendor class matching:

```
> grep dhcp-vendorclass /etc/dnsmasq.d/dhcp.conf
```

2. Verify boot file paths:

```
> grep dhcp-boot /etc/dnsmasq.d/dhcp.conf
```

3. Test TFTP access to boot files:

```
> tftp 192.168.1.200 -c get /boot/grub2/x86_64-efi/bootx64.efi
```

4. Monitor PXE-specific DHCP logs:

```
> journalctl -u dnsmasq | grep -E "PXE|HTTP"
```

5. Check tag assignment in logs:

```
> journalctl -u dnsmasq | grep "tags:"
```

12.5.4 IPv6 DHCP configuration issues

IPv6 DHCP clients require proper router advertisement configuration and may have different addressing requirements than IPv4.

PROCEDURE 48: RESOLVING IPV6 DHCP PROBLEMS

1. Verify IPv6 DHCP range configuration:

```
> grep "2001:db8" /etc/dnsmasq.d/dhcp.conf
```

2. Check IPv6 router advertisement status:

```
> systemctl status radvd
```

3. Monitor DHCPv6 requests:

```
> journalctl -u dnsmasq | grep "DHCPv6"
```

4. Test IPv6 connectivity:

```
> ping6 2001:db8:0:1::200
```

5. Check IPv6 option configuration:

```
> grep option6 /etc/dnsmasq.d/dhcp.conf
```

12.6 Next steps

dnsmasq now provides network configuration and boot file information to PXE clients over both IPv4 and IPv6. Its tag-based system assigns the correct boot file for each client architecture and boot method.

13 Configuring a DHCP server using Kea

This section explains how to configure DHCP services using Kea to provide network configuration and PXE boot information for SUSE Linux Enterprise Server 16.1 installations. Kea is a modern DHCP server that supports both IPv4 and IPv6 with client class matching for PXE and HTTP boot scenarios.

13.1 The role of Kea DHCP in PXE boot environments

Kea is the modern DHCP server developed by ISC as the successor to the legacy ISC DHCP server. It supports both DHCPv4 and DHCPv6 with client classification capabilities that enable proper boot file delivery based on client architecture and boot method. Kea uses JSON-based configuration files and supports advanced features like vendor class identification for HTTP boot.

13.2 Requirements for configuring Kea DHCP services

- Kea DHCP packages installed: `kea-dhcp4` and `kea-dhcp6`.
- PXE boot files properly organized under `/srv/tftpboot`.
- Network interface configured for DHCP service.
- Administrative privileges to configure DHCP services.

13.3 Configuring the Kea DHCPv4 server

The Kea DHCPv4 configuration uses client classes to match PXE and HTTP client types and provide appropriate boot files for different architectures.

PROCEDURE 49: SETTING UP KEA DHCPV4 SERVER

1. Configure the Kea DHCPv4 server:

```
> sudo cat > /etc/kea/kea-dhcp4.conf << 'EOF'
{
  "Dhcp4": {
    "interfaces-config": {
      "interfaces": [
        "eno1"
      ]
```

```
    },
    "control-socket": {
      "socket-type": "unix",
      "socket-name": "/tmp/kea4-ctrl-socket"
    },
    "lease-database": {
      "type": "memfile",
      "persist": true,
      "name": "/var/lib/kea/dhcp4.leases",
      "lfc-interval": 3600
    },
    "expired-leases-processing": {
      "reclaim-timer-wait-time": 10,
      "flush-reclaimed-timer-wait-time": 25,
      "hold-reclaimed-time": 3600,
      "max-reclaim-leases": 100,
      "max-reclaim-time": 250,
      "unwarned-reclaim-cycles": 5
    },
    "renew-timer": 1800,
    "rebind-timer": 3150,
    "valid-lifetime": 3600,
    "option-data": [],
    "client-classes": [
      {
        "name": "pxeclients#00000",
        "test": "substring(option[60].hex,0,20) == 'PXEClient:Arch:00000'",
        "next-server": "192.168.1.200",
        "boot-file-name": "/boot/grub2/i386-pc/core.0"
      },
      {
        "name": "pxeclients#00007",
        "test": "substring(option[60].hex,0,20) == 'PXEClient:Arch:00007'",
        "next-server": "192.168.1.200",
        "boot-file-name": "/boot/grub2/x86_64-efi/bootx64.efi"
      },
      {
        "name": "pxeclients#0000e",
        "test": "substring(option[60].hex,0,20) == 'PXEClient:Arch:0000e'",
        "next-server": "192.168.1.200",
        "boot-file-name": "/boot/grub2/powerpc-ieee1275/core.elf"
      },
      {
        "name": "pxeclients#00011",
        "test": "substring(option[60].hex,0,20) == 'PXEClient:Arch:00011'",
        "next-server": "192.168.1.200",
        "boot-file-name": "/boot/grub2/arm64-efi/bootaa64.efi"
```

```
      },
      {
        "name": "httpclients#00016",
        "test": "substring(option[60].hex,0,21) == 'HTTPClient:Arch:00016'",
        "boot-file-name": "http://192.168.1.200/boot/grub2/x86_64-efi/bootx64.efi",
        "option-data": [
          {
            "name": "vendor-class-identifier",
            "data": "HTTPClient"
          }
        ]
      },
      {
        "name": "httpclients#00019",
        "test": "substring(option[60].hex,0,21) == 'HTTPClient:Arch:00019'",
        "boot-file-name": "http://192.168.1.200/boot/grub2/arm64-efi/bootaa64.efi",
        "option-data": [
          {
            "name": "vendor-class-identifier",
            "data": "HTTPClient"
          }
        ]
      }
    ],
    "subnet4": [
      {
        "id": 1,
        "subnet": "192.168.1.0/24",
        "pools": [
          {
            "pool": "192.168.1.100 - 192.168.1.199"
          }
        ],
        "option-data": [
          {
            "name": "routers",
            "data": "192.168.1.1"
          },
          {
            "name": "ntp-servers",
            "data": "192.168.1.1"
          },
          {
            "name": "domain-name-servers",
            "data": "192.168.1.200"
          },
          {
```

```
            "name": "domain-search",
            "data": "example.net"
          }
        ],
        "reservations": []
      }
    ],
    "loggers": [
      {
        "name": "kea-dhcp4",
        "output-options": [
          {
            "output": "/var/log/kea/dhcp4.log"
          }
        ],
        "severity": "INFO",
        "debuglevel": 0
      }
    ]
  }
}
EOF
```

2. Create the Kea log directory:

```
> sudo mkdir -p /var/log/kea
```

3. Test the Kea DHCPv4 configuration:

```
> sudo kea-dhcp4 -t /etc/kea/kea-dhcp4.conf
```

4. Enable and start the Kea DHCPv4 service:

```
> sudo systemctl enable --now kea-dhcp4
```

13.4 Configuring Kea DHCPv6 server

The Kea DHCPv6 configuration provides IPv6 address assignment and boot file information using vendor class matching for different client architectures.

PROCEDURE 50: SETTING UP KEA DHCPV6 SERVER

1. Configure the Kea DHCPv6 server:

```
> sudo cat > /etc/kea/kea-dhcp6.conf << 'EOF'
```

```
{
  "Dhcp6": {
    "interfaces-config": {
      "interfaces": [
        "eno1"
      ]
    },
    "control-socket": {
      "socket-type": "unix",
      "socket-name": "/tmp/kea6-ctrl-socket"
    },
    "lease-database": {
      "type": "memfile",
      "persist": true,
      "name": "/var/lib/kea/dhcp6.leases",
      "lfc-interval": 3600
    },
    "expired-leases-processing": {
      "reclaim-timer-wait-time": 10,
      "flush-reclaimed-timer-wait-time": 25,
      "hold-reclaimed-time": 3600,
      "max-reclaim-leases": 100,
      "max-reclaim-time": 250,
      "unwarned-reclaim-cycles": 5
    },
    "renew-timer": 1800,
    "rebind-timer": 2880,
    "preferred-lifetime": 3600,
    "valid-lifetime": 7200,
    "option-data": [],
    "option-def": [],
    "client-classes": [
      {
        "name": "pxeclients#00000",
        "test": "substring(option[16].hex,6,20) == 'PXEClient:Arch:00000'",
        "option-data": [
          {
            "name": "bootfile-url",
            "data": "tftp://[2001:db8:0:1::200]/boot/grub2/i386-pc/core.0"
          }
        ]
      },
      {
        "name": "pxeclients#00007",
        "test": "substring(option[16].hex,6,20) == 'PXEClient:Arch:00007'",
        "option-data": [
          {
```

```
            "name": "bootfile-url",
            "data": "tftp://[2001:db8:0:1::200]/boot/grub2/x86_64-efi/bootx64.efi"
          }
        ]
      },
      {
        "name": "pxeclients#0000e",
        "test": "substring(option[16].hex,6,20) == 'PXEClient:Arch:0000e'",
        "option-data": [
          {
            "name": "bootfile-url",
            "data": "tftp://[2001:db8:0:1::200]/boot/grub2/powerpc-ieee1275/
core.elf"
          }
        ]
      },
      {
        "name": "pxeclients#00011",
        "test": "substring(option[16].hex,6,20) == 'PXEClient:Arch:00011'",
        "option-data": [
          {
            "name": "bootfile-url",
            "data": "tftp://[2001:db8:0:1::200]/boot/grub2/arm64-efi/bootaa64.efi"
          }
        ]
      }
    ],
    "subnet6": [
      {
        "id": 1,
        "subnet": "2001:db8:0:1::/64",
        "interface": "eno1",
        "pools": [
          {
            "pool": "2001:db8:0:1:d::/112"
          }
        ],
        "option-data": [
          {
            "name": "sntp-servers",
            "data": "2001:db8:0:1::1"
          },
          {
            "name": "dns-servers",
            "data": "2001:db8:0:1::200"
          },
          {
```

```
            "name": "domain-search",
            "data": "example.net"
          }
        ],
        "reservations": []
      }
    ],
    "loggers": [
      {
        "name": "kea-dhcp6",
        "output-options": [
          {
            "output": "/var/log/kea/dhcp6.log"
          }
        ],
        "severity": "INFO",
        "debuglevel": 0
      }
    ]
  }
}
EOF
```

2. Test the Kea DHCPv6 configuration:

```
> sudo kea-dhcp6 -t /etc/kea/kea-dhcp6.conf
```

3. Enable and start the Kea DHCPv6 service:

```
> sudo systemctl enable --now kea-dhcp6
```



Note: Boot file paths require the optional Secure Boot step

The `bootx64.efi` and `bootaa64.efi` boot file paths used above (in both the DHCPv4 and DHCPv6 configuration) are the signed files from the optional Secure Boot step in *Step 2*. If you did not perform that step, replace them with `core.efi` (still under the respective `x86_64-efi`/`arm64-efi` directory), which works as long as UEFI Secure Boot is disabled on the client.

13.5 Verifying Kea DHCP configuration

Test the Kea DHCP server functionality to ensure proper network configuration and boot file delivery to PXE clients.

PROCEDURE 51: TESTING KEA DHCP SERVERS

1. Check Kea DHCPv4 service status:

```
> systemctl status kea-dhcp4
```

2. Check Kea DHCPv6 service status:

```
> systemctl status kea-dhcp6
```

3. Verify DHCP port binding:

```
> ss -ulnp | grep -E ":67|:547"
```

4. Monitor DHCPv4 logs:

```
> tail -f /var/log/kea/dhcp4.log
```

5. Monitor DHCPv6 logs:

```
> tail -f /var/log/kea/dhcp6.log
```

6. Check active DHCP leases:

```
> cat /var/lib/kea/dhcp4.leases
```

13.6 Troubleshooting Kea DHCP configuration

Common issues when configuring Kea DHCP servers for PXE boot environments.

13.6.1 Kea configuration and service issues

Kea services may fail to start due to JSON configuration errors or network interface problems.

PROCEDURE 52: RESOLVING KEA CONFIGURATION PROBLEMS

1. Test DHCPv4 configuration syntax:

```
> sudo kea-dhcp4 -t /etc/kea/kea-dhcp4.conf
```

2. Test DHCPv6 configuration syntax:

```
> sudo kea-dhcp6 -t /etc/kea/kea-dhcp6.conf
```

3. Check for JSON syntax errors:

```
> python3 -m json.tool /etc/kea/kea-dhcp4.conf
```

4. Verify network interface configuration:

```
> ip addr show eno1
```

5. Check Kea service logs:

```
> journalctl -u kea-dhcp4 -f
```

13.6.2 Kea DHCP lease assignment problems

Clients may fail to receive IP addresses due to subnet configuration or pool exhaustion issues.

PROCEDURE 53: DIAGNOSING KEA LEASE ISSUES

1. Check subnet and pool configuration:

```
> grep -A 10 "subnet4\|pools" /etc/kea/kea-dhcp4.conf
```

2. Monitor lease assignments in real-time:

```
> tail -f /var/log/kea/dhcp4.log | grep -E "ALLOC|DISCOVER"
```

3. Check lease database for conflicts:

```
> cat /var/lib/kea/dhcp4.leases | tail -20
```

4. Verify interface binding:

```
> grep interfaces /etc/kea/kea-dhcp4.conf
```

5. Clear lease database if needed:

```
> sudo systemctl stop kea-dhcp4
```

```
> sudo mv /var/lib/kea/dhcp4.leases /var/lib/kea/dhcp4.leases.backup
```

```
> sudo systemctl start kea-dhcp4
```

13.6.3 PXE client class matching issues

PXE clients may receive IP addresses but fail to get correct boot files due to client class configuration problems.

PROCEDURE 54: TROUBLESHOOTING KEA CLIENT CLASSIFICATION

1. Check client class definitions:

```
> grep -A 5 "client-classes" /etc/kea/kea-dhcp4.conf
```

2. Monitor client class matching in logs:

```
> tail -f /var/log/kea/dhcp4.log | grep -i class
```

3. Verify vendor class identifier patterns:

```
> grep "PXEClient\|HTTPClient" /etc/kea/kea-dhcp4.conf
```

4. Test boot file accessibility:

```
> curl -I http://192.168.1.200/boot/grub2/x86_64-efi/bootx64.efi
```

5. Enable debug logging for detailed client analysis:

```
> sudo sed -i 's/"debuglevel": 0/"debuglevel": 99/' /etc/kea/kea-dhcp4.conf
```

```
> sudo systemctl restart kea-dhcp4
```

13.6.4 DHCPv6-specific issues

IPv6 DHCP clients require proper router advertisement configuration and have different vendor class option handling than IPv4.

PROCEDURE 55: RESOLVING KEA DHCPV6 PROBLEMS

1. Check DHCPv6 subnet configuration:

```
> grep -A 10 "subnet6" /etc/kea/kea-dhcp6.conf
```

2. Verify IPv6 router advertisement status:

```
> systemctl status radvd
```

3. Monitor DHCPv6 vendor class matching:

```
> tail -f /var/log/kea/dhcp6.log | grep "option\[16\]"
```

4. Check IPv6 bootfile-url option format:

```
> grep "bootfile-url" /etc/kea/kea-dhcp6.conf
```

5. Test IPv6 connectivity to boot server:

```
> ping6 2001:db8:0:1::200
```

13.7 Next steps

Kea now provides network configuration and boot file information to PXE clients over both IPv4 and IPv6. Its client classification system assigns the correct boot file for each client architecture and supports both traditional PXE and modern HTTP boot methods.

14 Configuring a DHCP server using ISC DHCP

This section explains how to configure the ISC DHCP server to provide network configuration and PXE boot information. ISC DHCP remains fully supported on SUSE Linux Enterprise Server 15 (for example, SP7). On SUSE Linux Enterprise Server 16.1, the `dhcp-server` package is unsupported and is only available through the PackageHub/Backports-16.0 repository, not the standard module repositories. ISC DHCP uses class and subclass matching to support PXE and HTTP boot scenarios across different client architectures.

14.1 ISC DHCP support for PXE and HTTP boot environments

ISC DHCP is the traditional DHCP server that provides network configuration and boot file information to PXE clients using a class and subclass system. While ISC has declared this server end-of-life as of 2022, it remains widely used in existing deployments and supports PXE and HTTP boot scenarios with vendor class identification.

Important: ISC DHCP end-of-life status

ISC DHCP has been declared end-of-life by ISC in 2022. For new deployments, consider using Kea or dnsmasq instead. This configuration is provided for compatibility with existing ISC DHCP installations.

14.2 Requirements for configuring ISC DHCP services

- ISC DHCP packages installed: `dhcp-server`. On SUSE Linux Enterprise Server 16.1, this package requires the PackageHub/Backports-16.0 repository, since it is not part of the standard module repositories.
- PXE boot files properly organized under `/srv/tftpboot`.
- Network interface configured for DHCP service.
- Administrative privileges to configure DHCP services.

14.3 Configuring the ISC DHCPv4 server

The ISC DHCPv4 configuration uses class and subclass declarations to match PXE and HTTP client types and provide appropriate boot files for different architectures.

PROCEDURE 56: SETTING UP ISC DHCPV4 SERVER

1. Configure the ISC DHCPv4 server:

```
> sudo cat > /etc/dhcpd.conf << 'EOF'
# /etc/dhcpd.conf
#
# Sample configuration file for ISC dhcpd
#
# *** PLEASE CONFIGURE IT FIRST ***
#
# Don't forget to set the DHCPD_INTERFACE in the
# /etc/sysconfig/dhcpd file.
#

# if you want to use dynamical DNS updates, you should first read
# read /usr/share/doc/packages/dhcp-server/DDNS-howto.txt
#
ddns-updates off;
```

```
# Use this to enable / disable dynamic dns updates globally.
ddns-update-style none;

# default lease time
default-lease-time              3600;
max-lease-time                  7200;

##
## PXE / HTTP boot option declarations
##
class "pxeclients" {
        # PXEClient:Arch:00000:UNDI:002001
        match substring (option vendor-class-identifier, 0, 20);
}
class "httpclients" {
        # HTTPClient:Arch:00016:UNDI:003001
        match substring (option vendor-class-identifier, 0, 21);
}

##
## PXE / HTTP boot subclass request matches
##
subclass "pxeclients"   "PXEClient:Arch:00000" {
        next-server     192.168.1.200;
        filename        "/boot/grub2/i386-pc/core.0";
}
subclass "pxeclients"   "PXEClient:Arch:00007" {
        next-server     192.168.1.200;
        filename        "/boot/grub2/x86_64-efi/bootx64.efi";
}
subclass "pxeclients"   "PXEClient:Arch:0000e" {
        next-server     192.168.1.200;
        filename        "/boot/grub2/powerpc-ieee1275/core.elf";
}
subclass "pxeclients"   "PXEClient:Arch:00011" {
        next-server     192.168.1.200;
        filename        "/boot/grub2/arm64-efi/bootaa64.efi";
}

subclass "httpclients"  "HTTPClient:Arch:00016" {
        option vendor-class-identifier "HTTPClient";
        filename        "http://192.168.1.200/boot/grub2/x86_64-efi/bootx64.efi";
}
subclass "httpclients"  "HTTPClient:Arch:00019" {
        option vendor-class-identifier "HTTPClient";
        filename        "http://192.168.1.200/boot/grub2/arm64-efi/bootaa64.efi";
```

```
}

##
## Subnet declaration for the pxe network
##
subnet 192.168.1.0 netmask 255.255.255.0 {
        authoritative;

        range  dynamic-bootp            192.168.1.100 192.168.1.199;

        option subnet-mask              255.255.255.0;

        option routers                  192.168.1.1;
        option ntp-servers              192.168.1.1;
        option domain-name-servers      192.168.1.200;
        option domain-name              "example.net";
        option domain-search            "example.net";
}
EOF
```

2. Configure the DHCP interface in sysconfig:

```
> sudo echo 'DHCPD_INTERFACE="eno1"' > /etc/sysconfig/dhcpd
```

3. Test the DHCPv4 configuration:

```
> sudo dhcpd -t -cf /etc/dhcpd.conf
```

4. Enable and start the ISC DHCPv4 service:

```
> sudo systemctl enable --now dhcpd
```

14.4 Configuring the ISC DHCPv6 server

The ISC DHCPv6 configuration provides IPv6 address assignment and boot file information using vendor class matching with proper DHCPv6 option handling.

PROCEDURE 57: SETTING UP ISC DHCPV6 SERVER

1. Configure the ISC DHCPv6 server:

```
> sudo cat > /etc/dhcpd6.conf << 'EOF'
# /etc/dhcpd6.conf
#
# Sample DHCPv6 configuration file for ISC dhcpd
```

```
#
# *** PLEASE CONFIGURE IT FIRST ***
#
# Don't forget to set the DHCPD6_INTERFACE in the
# /etc/sysconfig/dhcpd file.
#

# if you want to use dynamical DNS updates, you should first
# read /usr/share/doc/packages/dhcp-server/DDNS-howto.txt
ddns-updates off;

# Use this to enable / disable dynamic dns updates globally.
ddns-update-style none;

# IPv6 address valid lifetime
#  (at the end the address is no longer usable by the client)
#  (set to 30 days, the usual IPv6 default)
default-lease-time 7200;

# IPv6 address preferred lifetime
#  (at the end the address is deprecated, i.e., the client should use
#   other addresses for new connections)
#  (set to 7 days, the usual IPv6 default)
preferred-lifetime 3600;

##
## PXE / HTTP boot option declarations
##

# The dhcp6 option 16 is in fact an:
#   { uint32 enterprise-number, array of { uint16 len, string tag} vendor-class-
data }
# this declaration is using the whole option data as string for substring match:
option dhcp6.vendor-class-as-string code 16 = string;

# this declaration is using the enterprise-number with 1st tag length and string:
option dhcp6.vendor-class-en-len-tag code 16 = {integer 32, integer 16, string};

class "pxeclients" {
        # PXEClient:Arch:00000:UNDI:002001
        # note: +6 to skip the enterprise-number+len until the PXEClient string
        match substring (option dhcp6.vendor-class-as-string, 6, 20);
}
class "httpclients" {
        # HTTPClient:Arch:00016:UNDI:003001
        # note: +6 to skip the enterprise-number+len until the HTTPClient string
        match substring (option dhcp6.vendor-class-as-string, 6, 21);
```

```
}

##
## PXE / HTTP boot subclass request matches
##
subclass "pxeclients"   "PXEClient:Arch:00000" {
        option dhcp6.bootfile-url "tftp://[2001:db8:0:1::200]/boot/grub2/i386-pc/
core.0";
}
subclass "pxeclients"   "PXEClient:Arch:00007" {
        option dhcp6.bootfile-url "tftp://[2001:db8:0:1::200]/boot/grub2/x86_64-efi/
bootx64.efi";
}
subclass "pxeclients"   "PXEClient:Arch:0000e" {
        option dhcp6.bootfile-url "tftp://[2001:db8:0:1::200]/boot/grub2/powerpc-
ieee1275/core.elf";
}
subclass "pxeclients"   "PXEClient:Arch:00011" {
        option dhcp6.bootfile-url "tftp://[2001:db8:0:1::200]/boot/grub2/arm64-efi/
bootaa64.efi";
}

subclass "httpclients"  "HTTPClient:Arch:00016" {
        option dhcp6.vendor-class-en-len-tag 343 10 "HTTPClient";
        option dhcp6.bootfile-url "http://[2001:db8:0:1::200]/boot/grub2/x86_64-efi/
bootx64.efi";
}
subclass "httpclients"  "HTTPClient:Arch:00019" {
        option dhcp6.vendor-class-en-len-tag 343 10 "HTTPClient";
        option dhcp6.bootfile-url "http://[2001:db8:0:1::200]/boot/grub2/arm64-efi/
bootaa64.efi";
}

##
## Subnet declaration for the pxe network
##
subnet6 2001:db8:0:1::/64 {
       authoritative;

       range6  2001:db8:0:1:d:: 2001:db8:0:1:d::ffff;

       option dhcp6.sntp-servers       2001:db8:0:1::1;
       option dhcp6.name-servers       2001:db8:0:1::200;
       option dhcp6.domain-search      "example.net";
}
EOF
```

2. Configure the DHCPv6 interface in sysconfig:

```
> sudo echo 'DHCPD6_INTERFACE="eno1"' >> /etc/sysconfig/dhcpd
```

3. Test the DHCPv6 configuration:

```
> sudo dhcpd -6 -t -cf /etc/dhcpd6.conf
```

4. Enable and start the ISC DHCPv6 service:

```
> sudo systemctl enable --now dhcpd6
```



Note: Boot file paths require the optional Secure Boot step

The `bootx64.efi` and `bootaa64.efi` boot file paths used above (in both the DHCPv4 and DHCPv6 configuration) are the signed files from the optional Secure Boot step in *Step 2*. If you did not perform that step, replace them with `core.efi` (still under the respective `x86_64-efi`/`arm64-efi` directory), which works as long as UEFI Secure Boot is disabled on the client.

14.5 Verifying the ISC DHCP configuration

Test the ISC DHCP server functionality to ensure proper network configuration and boot file delivery to PXE clients.

PROCEDURE 58: TESTING ISC DHCP SERVERS

1. Check ISC DHCPv4 service status:

```
> systemctl status dhcpd
```

2. Check ISC DHCPv6 service status:

```
> systemctl status dhcpd6
```

3. Verify DHCP port binding:

```
> ss -ulnp | grep -E ":67|:547"
```

4. Monitor DHCP logs:

```
> journalctl -u dhcpd -f
```

5. Check active DHCP leases:

```
> cat /var/lib/dhcp/dhcpd.leases
```

6. Monitor DHCPv6 activity:

```
> journalctl -u dhcpd6 -f
```

14.6 Troubleshooting ISC DHCP configuration

This section describes common issues when configuring ISC DHCP servers for PXE boot environments.

14.6.1 ISC DHCP configuration and service issues

ISC DHCP services may fail to start due to configuration syntax errors or interface binding problems.

PROCEDURE 59: RESOLVING ISC DHCP CONFIGURATION PROBLEMS

1. Test DHCPv4 configuration syntax:

```
> sudo dhcpd -t -cf /etc/dhcpd.conf
```

2. Test DHCPv6 configuration syntax:

```
> sudo dhcpd -6 -t -cf /etc/dhcpd6.conf
```

3. Check interface configuration:

```
> cat /etc/sysconfig/dhcpd
```

4. Verify network interface status:

```
> ip addr show eno1
```

5. Check for port conflicts:

```
> ss -ulnp | grep :67
```

6. View detailed service logs:

```
> journalctl -u dhcpd -xe
```

14.6.2 DHCP lease assignment problems

Clients may fail to receive IP addresses due to subnet configuration or authorization issues.

PROCEDURE 60: DIAGNOSING ISC DHCP LEASE ISSUES

1. Check subnet and range configuration:

```
> grep -A 10 "subnet\|range" /etc/dhcpd.conf
```

2. Verify authoritative setting:

```
> grep authoritative /etc/dhcpd.conf
```

3. Monitor lease assignments in real time:

```
> tail -f /var/log/messages | grep dhcpd
```

4. Check lease database for errors:

```
> tail -20 /var/lib/dhcp/dhcpd.leases
```

5. Test DHCP response manually:

```
> dhcping -s 192.168.1.200 -h aa:bb:cc:dd:ee:ff
```

14.6.3 ISC DHCP class and subclass matching issues

PXE clients may receive IP addresses but fail to get correct boot files due to class matching configuration problems.

PROCEDURE 61: TROUBLESHOOTING ISC DHCP CLASS MATCHING

1. Check class definitions:

```
> grep -A 3 "class.*clients" /etc/dhcpd.conf
```

2. Verify subclass entries:

```
> grep -A 5 "subclass" /etc/dhcpd.conf
```

3. Monitor vendor class identification:

```
> tail -f /var/log/messages | grep -E "PXEClient|HTTPClient"
```

4. Test boot file accessibility:

```
> tftp 192.168.1.200 -c get /boot/grub2/x86_64-efi/bootx64.efi
```

5. Enable detailed logging:

```
> sudo sed -i '1i\log-facility local7;' /etc/dhcpd.conf
```

```
> sudo systemctl restart dhcpd
```

14.6.4 ISC DHCPv6 vendor class option issues

IPv6 DHCP clients have complex vendor class option handling that may require specific configuration for proper PXE boot support.

PROCEDURE 62: RESOLVING ISC DHCPV6 PROBLEMS

1. Check DHCPv6 option definitions:

```
> grep -A 3 "option dhcp6" /etc/dhcpd6.conf
```

2. Verify vendor class string parsing:

```
> grep "substring.*6.*20\|21" /etc/dhcpd6.conf
```

3. Monitor DHCPv6 vendor class matching:

```
> journalctl -u dhcpd6 | grep -i vendor
```

4. Check IPv6 bootfile-url format:

```
> grep "bootfile-url" /etc/dhcpd6.conf
```

5. Verify router advertisement dependency:

```
> systemctl status radvd
```

6. Test IPv6 connectivity:

```
> ping6 2001:db8:0:1::200
```

14.7 Next steps

ISC DHCP now provides network configuration and boot file information to PXE clients using the traditional class and subclass system. While ISC DHCP is end-of-life, this configuration keeps existing deployments running with PXE and HTTP boot support across multiple client architectures.

15 Validating PXE server setup

This section describes how to validate and test the complete PXE server setup to ensure all components are working correctly for SUSE Linux Enterprise Server 16.1 network installations. It covers service verification, network connectivity testing and end-to-end PXE boot validation.

15.1 PXE server validation process

After configuring all PXE server components, including TFTP, HTTP, DNS, DHCP and GRUB 2 boot loader services, validate that the complete system functions correctly. This validation ensures that PXE clients can successfully boot into the Agama installer and perform network-based installations of SUSE Linux Enterprise Server 16.1.



Note: Boot file paths require the optional Secure Boot step

This section refers to `bootx64.efi` and `bootaa64.efi`, the signed files from the optional Secure Boot step in *Step 2*. If you did not perform that step, substitute `core.efi` (still under the respective `x86_64-efi`/`arm64-efi` directory) wherever this section references those files.

15.2 Requirements for PXE server validation

- All PXE server components configured and running.
- Test client systems capable of PXE booting.
- Network connectivity between the PXE server and clients.
- Administrative access to monitor server services.

15.3 Verifying PXE server services and component status

Verify that all required PXE server services are running and properly configured before testing with PXE clients.

PROCEDURE 63: CHECKING PXE SERVER SERVICE STATUS

1. Verify TFTP service status:

```
> systemctl status tftp.socket
```

   Expected result: Service should be active and listening on port 69.

2. Check nginx HTTP service:

```
> systemctl status nginx
```

   Expected result: Service should be active and listening on port 80.

3. Verify DNS service (if using dnsmasq):

```
> systemctl status dnsmasq
```

   Expected result: Service should be active and listening on port 53.

4. Check the DHCP service status for your chosen DHCP server:

   - For ISC DHCP:

```
> systemctl status dhcpd
```

   - For dnsmasq DHCP:

```
> systemctl status dnsmasq
```

   - For Kea DHCP:

```
> systemctl status kea-dhcp4 kea-dhcp6
```

   Expected result: DHCP service should be active and listening on appropriate ports.

5. Verify IPv6 router advertisement (if configured):

```
> systemctl status radvd
```

   Expected result: Service should be active for IPv6 environments.

6. Check NTP service:

```
> systemctl status chronyd
```

Expected result: Service should be active and synchronized.

15.4 Testing network connectivity and file access

Validate that PXE clients can access boot files and installation content over the network using both TFTP and HTTP protocols.

PROCEDURE 64: TESTING NETWORK FILE ACCESS

1. Test TFTP access to boot loader files:

```
> tftp localhost -c get /boot/grub2/x86_64-efi/bootx64.efi /tmp/test-bootx64.efi
```

Verify the file was retrieved:

```
> file /tmp/test-bootx64.efi
```

Clean up test file:

```
> rm /tmp/test-bootx64.efi
```

2. Test HTTP access to GRUB 2 configuration:

```
> curl -I http://localhost/boot/grub2/grub.cfg
```

Expected result: HTTP 200 OK response.

3. Verify HTTP access to installer files:

```
> curl -I http://localhost/boot/images/SLES-16.0/x86_64/liveiso/LiveOS/squashfs.img
```

Expected result: HTTP 200 OK response with appropriate content length.

4. Test DNS resolution (if local DNS configured):

```
> nslookup pxe.example.net localhost
```

Expected result: Proper A and AAAA record resolution.

5. Verify directory browsing for autoindex locations:

```
> curl http://localhost/boot/
```

Expected result: Directory listing showing boot files.

15.5 Validating DHCP functionality

Test DHCP server responses and verify that proper boot information is provided to different client types.

PROCEDURE 65: TESTING DHCP SERVER RESPONSES

1. Check DHCP port binding:

```
> ss -ulnp | grep -E ":67|:547"
```

Expected result: DHCP services listening on ports 67 (IPv4) and 547 (IPv6).

2. Monitor DHCP requests in real time for your chosen DHCP server:

   - For ISC DHCP:

     ```
     > journalctl -u dhcpd -f
     ```

   - For dnsmasq:

     ```
     > journalctl -u dnsmasq -f
     ```

   Leave this running to observe DHCP activity during testing.

3. Test DHCP response using dhcping (if available):

```
> dhcping -s 192.168.1.200
```

Expected result: Successful DHCP response from server.

4. Check active DHCP leases for your chosen DHCP server:

   - For ISC DHCP:

     ```
     > cat /var/lib/dhcp/dhcpd.leases
     ```

   - For Kea:

     ```
     > cat /var/lib/kea/dhcp4.leases
     ```

Expected result: Lease entries for test clients.

15.6 End-to-end PXE boot testing

Perform complete PXE boot tests with actual client systems to validate the entire boot process from DHCP to Agama installer startup.

PROCEDURE 66: TESTING COMPLETE PXE BOOT PROCESS

1. Prepare a test client system:
   - Configure BIOS/UEFI to enable network boot.
   - Set network boot as the first boot priority.
   - Connect the client to the same network as the PXE server.
2. Monitor PXE server logs during client boot:

```
> journalctl -f | grep -E "dhcp|tftp|nginx"
```

3. Boot the test client and observe the following sequence:
   1. Client should receive an IP address via DHCP.
   2. Client should download the boot loader via TFTP.
   3. The GRUB 2 menu should appear with installation options.
   4. The kernel and initrd should load via HTTP.
   5. The Agama installer should start successfully.
4. Verify client architecture detection by testing different client types:
   - Legacy BIOS x86_64 systems (should get `core.0`).
   - UEFI x86_64 systems (should get `bootx64.efi`).
   - UEFI aarch64 systems (should get `bootaa64.efi`).

5. Test IPv6 PXE boot (if IPv6 configured):

    - Enable IPv6-only network configuration on the test client.
    - Verify DHCPv6 address assignment.
    - Confirm IPv6 bootfile-url delivery.

15.7 Validating Agama installer functionality

Verify that the Agama installer starts correctly and can access installation sources for completing SUSE Linux Enterprise Server 16.1 installations.

PROCEDURE 67: TESTING AGAMA INSTALLER STARTUP

1. Verify Agama Web interface accessibility:
   During client boot, note the IP address assigned and access:

   ```
   http://CLIENT_IP_ADDRESS
   ```

   Expected result: Agama Web interface should load successfully.

2. Check Agama installer logs on the client:
   Switch to the console (Alt+F2) and run:

   ```
   # journalctl -u agama-web-server -f
   ```

   Expected result: No critical errors in Agama startup.

3. Verify installation source accessibility:
   For Full ISO installations, check repository access:

   ```
   # curl -I http://192.168.1.200/install/SLES-16.0/x86_64/
   ```

   Expected result: HTTP 403 Forbidden. Directory listing is intentionally not enabled for this location; a 403 response confirms nginx is serving requests from this path. Agama and `zypper` access specific files in the repository directly and never need a directory listing.

4. Test package installation capability:
   In the Agama interface, verify that:

   - The system can detect available disks.
   - Network configuration is preserved.
   - Package repository is accessible.
   - Installation can proceed to completion.

15.8 Troubleshooting PXE server validation failures

This section describes common issues during PXE server validation and their resolution steps.

15.8.1 DHCP assignment failures

Clients fail to receive IP addresses during PXE boot.

PROCEDURE 68: RESOLVING DHCP VALIDATION ISSUES

1. Check DHCP service conflicts:

   ```
   > ss -ulnp | grep :67
   ```

2. Verify network interface is up:

   ```
   > ip addr show eno1
   ```

3. Check DHCP range availability:

   ```
   > nmap -sn 192.168.1.100-199
   ```

4. Monitor DHCP logs for errors:

   ```
   > journalctl -u dhcpd | tail -50
   ```

15.8.2 Boot file delivery failures

Clients receive IP addresses but fail to download boot files.

PROCEDURE 69: RESOLVING BOOT FILE ISSUES

1. Verify TFTP service accessibility:

```
> tftp 192.168.1.200 -c get /boot/grub2/x86_64-efi/bootx64.efi
```

2. Check file permissions:

```
> ls -la /srv/tftpboot/boot/grub2/x86_64-efi/
```

3. Monitor TFTP access logs:

```
> journalctl -u tftp.socket -f
```

4. Verify client architecture detection:

```
> grep -E "PXEClient|HTTPClient" /var/log/messages
```

15.8.3 Agama installer startup failures

Boot files load successfully but Agama installer fails to start.

PROCEDURE 70: RESOLVING AGAMA STARTUP ISSUES

1. Check HTTP access to installer files:

```
> curl -I http://192.168.1.200/boot/images/SLES-16.0/x86_64/liveiso/LiveOS/
squashfs.img
```

2. Verify kernel parameter syntax in GRUB 2 configuration:

```
> grep "root=live:" /srv/tftpboot/boot/grub2/menu.cfg
```

3. Monitor client boot process:

```
> journalctl -f | grep -E "kernel|initrd|agama"
```

4. Check network configuration persistence:

```
# ip addr show
```

15.9 PXE server validation checklist

Use this checklist to systematically verify all aspects of your PXE server configuration.

TABLE 2: PXE SERVER VALIDATION CHECKLIST

Component	Validation Step	Status
TFTP Service	Service active, port 69 listening, files accessible	☐
HTTP Service	nginx active, port 80 listening, installer files accessible	☐
DNS Service	Host name resolution working, port 53 listening	☐
DHCP Service	IP assignment working, boot options delivered	☐
GRUB 2 Configuration	Menu loads, architecture detection working	☐
IPv6 Support	Router advertisement active, DHCPv6 functioning	☐
PXE Boot	Client boots successfully, receives correct boot loader	☐
Agama Installer	Installer starts; Web interface accessible	☐
Installation Source	Repository accessible, packages installable	☐
Network Persistence	Network configuration maintained during installation	☐

15.10 Validation conclusion

A properly validated PXE server should demonstrate successful end-to-end functionality from client network boot through Agama installer startup. All services should operate without errors, and clients should be able to complete SUSE Linux Enterprise Server 16.1 installations over the network. Regular validation testing ensures the continued reliability of the PXE infrastructure for automated deployments.

16 Legal Notice

Copyright© 2006– 2026 SUSE LLC and contributors. All rights reserved.

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

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

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

A GNU Free Documentation License

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image for-

mats 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.