Red Hat Enterprise Linux 10

Automatically installing RHEL

Deploying RHEL on one or more systems from a predefined configuration

Last Updated: 2025-12-30

Red Hat Enterprise Linux 10 Automatically installing RHEL

Deploying RHEL on one or more systems from a predefined configuration

Legal Notice

Abstract

You can automate the RHEL installation by using Kickstart. Use this method to deploy the same RHEL configuration on many systems. Kickstart installs RHEL based on the parameters that you specify in a configuration file. The installation source can be an installation media, an ISO file, the Red Hat content delivery network (CDN), or a server in your local network.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting feedback through Jira (account required)

1. Log in to the Jira website.
2. Click **Create** in the top navigation bar
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Create** at the bottom of the dialogue.

CHAPTER 1. SYSTEM REQUIREMENTS AND SUPPORTED ARCHITECTURES

Red Hat Enterprise Linux 10 delivers a stable, secure, consistent foundation across hybrid cloud deployments with the tools needed to deliver workloads faster with less effort. You can deploy RHEL as a guest on supported hypervisors and Cloud provider environments as well as on physical infrastructure. It helps the applications take advantage of innovations in the leading hardware architecture platforms.

Review the guidelines provided for system, hardware, security, memory, and storage configuration before installing.

If you want to use your system as a virtualization host, review the necessary hardware requirements for virtualization.

RHEL supports the following architectures:

- AMD and Intel 64-bit architectures
- The 64-bit ARM architecture
- IBM Power Systems, Little Endian
- 64-bit IBM Z architectures

1.1. SUPPORTED INSTALLATION TARGETS

An installation target is a storage device that stores Red Hat Enterprise Linux and boots the system. Red Hat Enterprise Linux supports the following installation targets for IBM Z, IBM Power, AMD64, Intel 64, and 64-bit ARM systems:

- Storage connected by a standard internal interface, such as DASD, SCSI, SATA, or SAS
- BIOS/firmware RAID devices on the Intel64, AMD64 and arm64 architectures
- Fibre Channel Host Bus Adapters and multipath devices. Some can require vendor-provided drivers.
- Xen block devices on Intel processors in Xen virtual machines.
- VirtIO block devices on Intel processors in KVM virtual machines.

Red Hat does not support installation to USB drives or SD memory cards. For information about support for third-party virtualization technologies, see the Red Hat Hardware Compatibility List .

1.2. DISK AND MEMORY REQUIREMENTS

If several operating systems are installed, it is important that you verify that the allocated disk space is separate from the disk space required by Red Hat Enterprise Linux. In some cases, it is important to dedicate specific partitions to Red Hat Enterprise Linux, for example, for AMD64, Intel 64, and 64-bit ARM, at least two partitions (/ and **swap**) must be dedicated to RHEL and for IBM Power Systems servers, up to three partitions (/, **swap**, and potentially a **PReP** boot partition) must be dedicated to RHEL.

Additionally, you must have a minimum of 10 GiB of available disk space. To install Red Hat Enterprise Linux, you must have a minimum of 10 GiB of space in either unpartitioned disk space or in partitions that can be deleted.

Table 1.1. Minimum RAM requirements

Installation type	Minimum RAM
Local media installation (USB, DVD)	• 1.5 GiB for aarch64, IBM Z and x86_64 architectures • 3 GiB for ppc64le architecture
NFS network installation	• 1.5 GiB for aarch64, IBM Z and x86_64 architectures • 3 GiB for ppc64le architecture
HTTP, HTTPS or FTP network installation	• 3 GiB for IBM Z and x86_64 architectures • 4 GiB for aarch64 and ppc64le architectures

It is possible to complete the installation with less memory than the minimum requirements. The exact requirements depend on your environment and installation path. Test various configurations to determine the minimum required RAM for your environment. Installing Red Hat Enterprise Linux using a Kickstart file has the same minimum RAM requirements as a standard installation. However, additional RAM may be required if your Kickstart file includes commands that require additional memory, or write data to the RAM disk. For more information, see the Automatically installing RHEL document.

1.3. GRAPHICS DISPLAY RESOLUTION REQUIREMENTS

Your system must have the following minimum resolution to ensure a smooth and error-free installation of Red Hat Enterprise Linux.

Table 1.2. Display resolution

Product version	Resolution
Red Hat Enterprise Linux 10	**Minimum**: 800 x 600 **Recommended**: 1024 x 768

1.4. UEFI SECURE BOOT AND BETA RELEASE REQUIREMENTS

If you plan to install a Beta release of Red Hat Enterprise Linux, on systems having UEFI Secure Boot enabled, first disable the UEFI Secure Boot option and then begin the installation.

UEFI Secure Boot requires that the operating system kernel is signed with a recognized private key, which the system's firmware verifies using the corresponding public key.

For Red Hat Enterprise Linux Beta releases, the kernel is signed with a Red Hat Beta-specific public key, which the system fails to recognize by default. As a result, the system fails to even boot the installation media.

Additional resources

- IBM installation documentation
- Security hardening
- Composing a customized RHEL system image
- Red Hat ecosystem catalog
- RHEL technology capabilities and limits

CHAPTER 2. THE VALUE OF REGISTERING YOUR RHEL SYSTEM TO RED HAT

Registration establishes an authorized connection between your system and Red Hat. Red Hat issues the registered system, whether a physical or virtual machine, a certificate that identifies and authenticates the system. The certificate helps to receive protected content, software updates, security patches, support, and managed services from Red Hat.

With a valid subscription, you can register a Red Hat Enterprise Linux (RHEL) system in the following ways:

- During the installation process, using an installer graphical user interface (GUI)
- After installation, using the command line interface (CLI)
- Automatically, during or after installation, using a kickstart script or an activation key

The specific steps to register your system depend on the version of RHEL that you are using and the registration method that you choose.

Registering your system to Red Hat enables features and capabilities that you can use to manage your system and report data. For example, a registered system is authorized to access protected content repositories for subscribed products through the Red Hat Content Delivery Network (CDN) or a Red Hat Satellite Server. These content repositories contain Red Hat software packages and updates, available only to customers with an active subscription. The packages and updates include security patches, bug fixes, and new features for RHEL and other Red Hat products.

CHAPTER 3. CUSTOMIZING THE INSTALLATION MEDIA

Customize RHEL installation media to create tailored system images with specific configurations, packages, and settings. It enables deployment of pre-configured systems that meet specific organizational requirements and reduces post-installation configuration time.

For details, see Composing a customized RHEL system image .

CHAPTER 4. CREATING A BOOTABLE INSTALLATION MEDIUM FOR RHEL

You can download the ISO file from the *Customer Portal* to prepare the bootable physical installation medium, such as a USB or DVD. Starting with RHEL 8, Red Hat no longer provides separate variants for **Server** and **Workstation**. **Red Hat Enterprise Linux for x86_64**includes both **Server** and **Workstation** capabilities. The distinction between **Server** and **Workstation** is managed through the System Purpose Role during the installation or configuration process.

After downloading an ISO file from the Customer Portal, create a bootable physical installation medium, such as a USB or DVD to continue the installation process.

For secure environment cases where USB drives are prohibited, consider using the Image Builder to create and deploy reference images. This method ensures compliance with security policies while maintaining system integrity. For more details, refer to the Image builder documentation.

4.1. INSTALLATION BOOT MEDIA OPTIONS

There are several options available to boot the Red Hat Enterprise Linux installation program.

Full installation DVD or USB flash drive

Create a full installation DVD or USB flash drive by using the **DVD ISO** image. The DVD or USB flash drive can be used as a boot device and as an installation source for installing software packages.

Minimal installation DVD or USB flash drive

Create a minimal installation DVD or USB flash drive by using the **Boot ISO** image, which contains only the minimum files necessary to boot the system and start the installation program. If you are not using the Content Delivery Network (CDN) to download the required software packages, the **Boot ISO** image requires an installation source that contains the required software packages.

4.2. CREATING A BOOTABLE DVD

You can create a bootable installation DVD by using a burning software and a DVD burner. The exact steps to produce a DVD from an ISO image file vary and depend on the operating system and disc burning software installed. Consult your system's burning software documentation for the exact steps to burn a DVD from an ISO image file.

WARNING

You can create a bootable DVD by using either the DVD ISO image (full install) or the Boot ISO image (minimal install). However, the DVD ISO image is larger than 4.7 GB, and as a result, it might not fit on a single or dual-layer DVD. Check the size of the DVD ISO image file before you proceed. Use a USB flash drive when using the DVD ISO image to create bootable installation media. For the environment cases where USB drives are prohibited, see Image builder documentation.

4.3. CREATING A BOOTABLE USB DEVICE ON LINUX

You can create a bootable USB device which you can then use to install Red Hat Enterprise Linux on other machines. This procedure overwrites the existing data on the USB drive without any warning. Back up any data or use an empty flash drive. A bootable USB drive cannot be used for storing data.

Prerequisites

- You have downloaded the full installation DVD ISO or minimal installation Boot ISO image from the Product Downloads page.
- You have a USB flash drive with enough capacity for the ISO image. The required size varies, but the minimum recommended USB size is 16 GB.

Procedure

1. Connect the USB flash drive to the system.
2. Log in as a root user:

   ```
   $ su -
   ```

   Enter your root password when prompted.
3. Find the device node assigned to the drive in the log of recent events. Messages resulting from the attached USB flash drive are displayed at the bottom of the log. In this example, the drive name is **sdd**.

   ```
   # dmesg|tail
   [288954.686557] usb 2-1.8: New USB device strings: Mfr=0, Product=1, SerialNumber=2
   [288954.686559] usb 2-1.8: Product: USB Storage
   [288954.686562] usb 2-1.8: SerialNumber: 000000009225
   [288954.712590] usb-storage 2-1.8:1.0: USB Mass Storage device detected
   [288954.712687] scsi host6: usb-storage 2-1.8:1.0
   [288954.712809] usbcore: registered new interface driver usb-storage
   [288954.716682] usbcore: registered new interface driver uas
   [288955.717140] scsi 6:0:0:0: Direct-Access     Generic  STORAGE DEVICE   9228 PQ: 0
   ANSI: 0
   [288955.717745] sd 6:0:0:0: Attached scsi generic sg4 type 0
   [288961.876382] sd 6:0:0:0: sdd Attached SCSI removable disk
   ```
4. If the inserted USB device mounts automatically, unmount it before continuing with the next steps. For unmounting, use the **umount** command. For more information, see Unmounting a file system with umount.
5. Write the ISO image directly to the USB device:

   ```
   # dd if=/image_directory/image.iso of=/dev/device
   ```

   - Replace */image_directory/image.iso* with the full path to the ISO image file that you downloaded,
   - Replace *device* with the device name that you retrieved with the **dmesg** command. In this example, the full path to the ISO image is **/home/testuser/Downloads/rhel-10-x86_64-boot.iso**, and the device name is **sdd**:

     ```
     # dd if=/home/testuser/Downloads/rhel-10-x86_64-boot.iso of=/dev/sdd
     ```

- Partition names are usually device names with a numerical suffix. For example, **sdd** is a device name, and **sdd1** is the name of a partition on the device **sdd**.

6. Wait for the **dd** command to finish writing the image to the device. Run the **sync** command to synchronize cached writes to the device. The data transfer is complete when the **#** shell prompt appears. When you see the prompt, log out of the root account and unplug the USB drive. The USB drive is now ready to use as a boot device.

4.4. CREATING A BOOTABLE USB DEVICE ON WINDOWS

You can create a bootable USB device on a Windows system with various tools. You can use Fedora Media Writer, available for download at https://github.com/FedoraQt/MediaWriter/releases. Fedora Media Writer is a community product and is not supported by Red Hat. You can report any issues with the tool at https://github.com/FedoraQt/MediaWriter/issues.

Creating a bootable drive overwrites existing data on the USB drive without any warning. Back up any data or use an empty flash drive. A bootable USB drive cannot be used for storing data.

Prerequisites

- You have downloaded the full installation DVD ISO or minimal installation Boot ISO image from the Product Downloads page.
- You have a USB flash drive with enough capacity for the ISO image. The required size varies.

Procedure

1. Download and install Fedora Media Writer from https://github.com/FedoraQt/MediaWriter/releases.
2. Connect the USB flash drive to the system.
3. Open Fedora Media Writer.
4. From the main window, click **Custom Image** and select the previously downloaded Red Hat Enterprise Linux ISO image.
5. From the **Write Custom Image** window, select the drive that you want to use.
6. Click **Write to disk**. The boot media creation process starts. Do not unplug the drive until the operation completes. The operation may take several minutes, depending on the size of the ISO image, and the write speed of the USB drive.
7. When the operation completes, unmount the USB drive. The USB drive is now ready to be used as a boot device.

4.5. CREATING A BOOTABLE USB DEVICE ON MACOS

You can create a bootable USB device which you can then use to install Red Hat Enterprise Linux on other machines. Creating a bootable USB drive overwrites any data previously stored on the USB drive without any warning. Back up any data or use an empty flash drive. A bootable USB drive cannot be used for storing data.

Prerequisites

- You have downloaded the full installation DVD ISO or minimal installation Boot ISO image from the Product Downloads page.
- You have a USB flash drive with enough capacity for the ISO image. The required size varies.

Procedure

1. Connect the USB flash drive to the system.
2. Identify the device path with the **diskutil list** command. The device path has the format of ***/dev/disknumber***, where **number** is the number of the disk. The disks are numbered starting at zero (0). Typically, **disk0** is the OS X recovery disk, and **disk1** is the main OS X installation. In the following example, the USB device is **disk2**:

```
$ diskutil list
/dev/disk0
#:                    TYPE NAME                    SIZE       IDENTIFIER
0:     GUID_partition_scheme                        *500.3 GB   disk0
1:                       EFI EFI                209.7 MB   disk0s1
2:          Apple_CoreStorage                      400.0 GB   disk0s2
3:              Apple_Boot Recovery HD            650.0 MB   disk0s3
4:          Apple_CoreStorage                      98.8 GB    disk0s4
5:              Apple_Boot Recovery HD            650.0 MB   disk0s5
/dev/disk1
#:                    TYPE NAME                    SIZE       IDENTIFIER
0:               Apple_HFS YosemiteHD             *399.6 GB   disk1
Logical Volume on disk0s1
8A142795-8036-48DF-9FC5-84506DFBB7B2
Unlocked Encrypted
/dev/disk2
#:                    TYPE NAME                    SIZE       IDENTIFIER
0:    FDisk_partition_scheme                        *8.1 GB     disk2
1:              Windows_NTFS SanDisk USB            8.1 GB     disk2s1
```

3. Identify your USB flash drive by comparing the NAME, TYPE and SIZE columns to your flash drive. For example, the NAME should be the title of the flash drive icon in the **Finder** tool. You can also compare these values to those in the information panel of the flash drive.
4. Unmount the flash drive’s file system volumes:

```
$ diskutil unmountDisk /dev/disknumber
Unmount of all volumes on disknumber was successful
```

   When the command completes, the icon for the flash drive disappears from your desktop. If the icon does not disappear, you may have selected the wrong disk. Attempting to unmount the system disk accidentally returns a **failed to unmount** error.

5. Write the ISO image to the flash drive. macOS provides both a block (**/dev/disk***) and character device (**/dev/rdisk***) file for each storage device. Writing an image to the **/dev/rdisknumber** character device is faster than writing to the **/dev/disknumber** block device. For example, to write the ***/Users/user_name/Downloads/rhel-{ProductNumber}-x86_64-boot.iso*** file to the ***/dev/rdisk2*** device, enter the following command:

```
# sudo dd if=/Users/user_name/Downloads/rhel-{ProductNumber}-x86_64-boot.iso
of=/dev/rdisk2 bs=512K status=progress
```

- **if=** - Path to the installation image.
- **of=** - The raw disk device *(/dev/rdisknumber)* representing the target disk.
- **bs=512K** - Sets the block size to 512 KB for faster data transfer.
- **status=progress** - Displays a progress indicator during the operation.

6. Wait for the **dd** command to finish writing the image to the device. The data transfer is complete when the **#** prompt appears. When the prompt is displayed, log out of the root account and unplug the USB drive. The USB drive is now ready to be used as a boot device.

Additional resources

- Configuring System Purpose
- ISO for RHEL 8/9 Server or Workstation

CHAPTER 5. PREPARING NETWORK-BASED REPOSITORIES

You must prepare repositories to install RHEL from your network system.

5.1. CREATING AN INSTALLATION SOURCE ON AN NFS SERVER

You can configure an NFS server to host RHEL installation files for network-based installations. It enables multiple systems to install from a shared network location, providing efficient deployment across your infrastructure without requiring physical media for each system.

Prerequisites

- You have administrator-level access to a server with Red Hat Enterprise Linux 10, and this server is on the same network as the system to be installed.
- You have downloaded the full installation DVD ISO from the Product Downloads page.

IMPORTANT

Ensure that you use different paths in **inst.ks** and **inst.repo**. When using NFS to host the installation source, you cannot use the same nfs share to host the Kickstart.

Procedure

1. Install the **nfs-utils** package:

   ```
   # dnf install nfs-utils
   ```

2. Copy the DVD ISO image to a directory on the NFS server. This example assumes the DVD ISO is copied to the **/nfs/rhel10-install/** directory on the NFS server.

3. Open the **/etc/exports** file using a text editor and add a line with the following syntax:

   ```
   /exported_directory/ clients
   ```

   - Replace */exported_directory/* with the full path to the directory with the ISO image.
   - Replace *clients* with one of the following:
     - The host name or IP address of the target system
     - The subnetwork that all target systems can use to access the ISO image
     - To allow any system with network access to the NFS server to use the ISO image, the asterisk sign (*)

   See the **exports(5)** man page for detailed information about the format of this field.

   For example, a basic configuration that makes the **/nfs/rhel10-install/** directory available as read-only to all clients is:

   ```
   /nfs/rhel10-install *
   ```

4. Save the **/etc/exports** file and exit the text editor.

5. Enable the NFS service in firewalld:

```
# firewall-cmd --permanent --add-service nfs
# firewall-cmd --permanent --add-service=mountd
# firewall-cmd --permanent --add-service=rpc-bind
# firewall-cmd --reload
```

6. Start the nfs service:

```
# systemctl enable --now nfs-server.service
```

   If the service was running before you changed the **/etc/exports** file, reload the NFS server configuration:

```
# systemctl reload nfs-server.service
```

   The ISO image is now accessible over NFS and ready to be used as an installation source.

   When configuring the installation source, use **nfs:** as the protocol, the server host name or IP address, the colon sign **(:)**, and the directory holding the ISO image. For example, if the server host name is **myserver.example.com** and you have saved the ISO image in **/nfs/rhel10-install/**:

   - On kernel command line, define the NFS installation source as **inst.repo=nfs:myserver.example.com:/nfs/rhel10-install/** and
   - In the Kickstart file, define the NFS installation source as **nfs --server=myserver.example.com --dir=/nfs/rhel10-install/**.

Additional resources

- Deploying an NFS server

5.2. CREATING AN INSTALLATION SOURCE BY USING HTTP OR HTTPS

You can create an installation source for a network-based installation by using an installation tree. It is a directory containing extracted contents of the DVD ISO image and a valid **.treeinfo** file. The installation source is accessed over HTTP or HTTPS.

Prerequisites

- You have administrator-level access to a server with Red Hat Enterprise Linux 10, and this server is on the same network as the system to be installed.
- You have downloaded the full installation DVD ISO from the Product Downloads page.

IMPORTANT

If you use an HTTPS server with a self-signed certificate, you must boot the installation program with the **noverifyssl** option.

Procedure

1. Install the httpd package.

   ```
   # dnf install httpd
   ```

2. Optional: Install the **mod_ssl** package if you want to use the https installation source:

   ```
   # dnf install mod_ssl
   ```

3. Copy the DVD ISO image to the HTTP(S) server.

4. Create a suitable directory for mounting the DVD ISO image, for example:

   ```
   # mkdir /mnt/rhel10-install/
   ```

5. Mount the DVD ISO image to the directory:

   ```
   # mount -o loop,ro -t iso9660 /image_directory/image.iso /mnt/rhel10-install/
   ```

   Replace */image_directory/image.iso* with the path to the DVD ISO image.

6. Copy the files from the mounted image to the HTTP(S) server root.

   ```
   # cp -r /mnt/rhel10-install/ /var/www/html/
   ```

   This command creates the **/var/www/html/rhel10-install/** directory with the content of the image. Note that some other copying methods might skip the **.treeinfo** file which is required for a valid installation source. Entering the **cp** command for entire directories copies **.treeinfo** correctly.

7. Unmount the DVD ISO:

   ```
   # umount /mnt/rhel10-install/
   ```

8. Enable the http service in firewalld:

   ```
   # firewall-cmd --permanent --add-service=http
   # firewall-cmd --reload
   ```

9. Optional: Enable the https service in firewalld:

   ```
   # firewall-cmd --permanent --add-service=https
   # firewall-cmd --reload
   ```

10. Start the **httpd** service:

    ```
    # systemctl enable --now httpd.service
    ```

    The installation tree is now accessible and ready to be used as the installation source.

NOTE

When configuring the installation source, use **http://** or **https://** as the protocol, the server host name or IP address, and the directory that contains the files from the ISO image, relative to the HTTP server root. For example, if you use HTTP, the server host name is **myserver.example.com**, and you have copied the files from the image to **/var/www/html/rhel10-install/**, specify **http://myserver.example.com/rhel10-install/** as the installation source.

Additional resources

- Deploying different types of servers
- Setting up the Apache HTTP web server

5.3. CREATING AN INSTALLATION SOURCE BY USING FTP

You can set up an FTP server to host RHEL installation files for network-based installations. It enables multiple systems to install from a centralized source, providing efficient deployment across your network infrastructure using the FTP protocol.

Prerequisites

- You have administrator-level access to a server with Red Hat Enterprise Linux 10, and this server is on the same network as the system to be installed.
- You have downloaded the full installation DVD ISO from the Product Downloads page.
- The **vsftpd** package is installed.

Procedure

1. Install the vsftpd package.

   ```
   # dnf install vsftpd
   ```

2. Open and edit the **/etc/vsftpd/vsftpd.conf** configuration file in a text editor.

   a. Change the line **anonymous_enable=NO** to **anonymous_enable=YES**

   b. Change the line **write_enable=YES** to **write_enable=NO**.

   c. Add lines **pasv_min_port=<*min_port*>** and **pasv_max_port=<*max_port*>**. Replace <*min_port*> and <*max_port*> with the port number range used by the FTP server in passive mode, for example, **10000** and **11000**.
   This step might be necessary in network environments featuring various firewall/NAT setups.

   d. Optional: Add custom changes to your configuration. For available options, see the **vsftpd.conf(5)** man page. This procedure assumes that default options are used.

   e. Configure the firewall to allow the FTP port and port range from the previous step:

   ```
   # firewall-cmd --add-port min_port-max_port/tcp --permanent
   ```

Replace <*min_port*> and <*max_port*> with the port numbers you entered into the **/etc/vsftpd/vsftpd.conf** configuration file.

f. Configure the firewall to allow FTP service

```
# firewall-cmd --add-service ftp --permanent
```

g. Reload the firewall to apply the new rules:

```
# firewall-cmd --reload
```

3. Copy the DVD ISO image to the FTP server.

4. Create a suitable directory for mounting the DVD ISO image, for example:

```
# mkdir /mnt/rhel10-install
```

5. Mount the DVD ISO image to the directory:

```
# mount -o loop,ro -t iso9660 /image-directory/image.iso /mnt/rhel10-install
```

Replace ***/image-directory/image.iso*** with the path to the DVD ISO image.

6. Copy the files from the mounted image to the FTP server root:

```
# cp -r /mnt/rhel10-install/ /var/ftp/
```

This command creates the **/var/ftp/rhel10-install/** directory with the content of the image. Some copying methods can skip the **.treeinfo** file which is required for a valid installation source. Entering the **cp** command for whole directories as shown in this procedure will copy **.treeinfo** correctly.

7. Start the **vsftpd** service:

```
# systemctl enable --now vsftpd.service
```

If the service was running before you changed the **/etc/vsftpd/vsftpd.conf** file, restart the service to load the edited file:

```
# systemctl restart vsftpd.service
```

The installation tree is now accessible and ready to be used as the installation source.

When configuring the installation source, use **ftp://** as the protocol, the server host name or IP address, and the directory in which you have stored the files from the ISO image, relative to the FTP server root. For example, if the server host name is **myserver.example.com** and you have copied the files from the image to **/var/ftp/rhel10-install/**, specify **ftp://myserver.example.com/rhel10-install/** as the installation source.

CHAPTER 6. PREPARING A UEFI HTTP INSTALLATION SOURCE

As an administrator of a server on a local network, you can configure an HTTP server to enable HTTP boot and network installation for other systems on your network.

6.1. NETWORK INSTALLATION OVERVIEW

A network installation allows you to install Red Hat Enterprise Linux to a system that has access to an installation server. At a minimum, two systems are required for a network installation:

Server

A system running a DHCP server, an HTTP, HTTPS, FTP, or NFS server, and in the PXE boot case, a TFTP server. Although each server can run on a different physical system, the procedures in this section assume a single system is running all servers.

Client

The system to which you are installing Red Hat Enterprise Linux. Once installation starts, the client queries the DHCP server, receives the boot files from the HTTP or TFTP server, and downloads the installation environment image from the HTTP, HTTPS, FTP or NFS server. Unlike other installation methods, the client does not require any physical boot media for the installation to start.

To boot a client from the network, enable network boot in the firmware or in a quick boot menu on the client. On some hardware, the option to boot from a network might be disabled, or not available.

The workflow to prepare for an installation of Red Hat Enterprise Linux from a network by using HTTP or PXE is as follows:

1. Export the installation ISO image or the installation tree to an NFS, HTTPS, HTTP, or FTP server.

2. Configure the HTTP or TFTP server and DHCP server, and start the HTTP or TFTP service on the server.

3. Boot the client and start the installation.

You can choose between the following network boot protocols:

HTTP

Use the HTTP boot if your client UEFI supports it. HTTP boot is usually more reliable.

PXE (TFTP)

PXE boot is more widely supported by client systems, but sending the boot files over TFTP protocol might be slow and result in timeout failures.

Additional resources

- Preparing network based repositories

- Red Hat Satellite product documentation

6.2. CONFIGURING THE DHCPV4 SERVER FOR NETWORK BOOT

Enable the DHCP version 4 (DHCPv4) service on your server, so that it can provide network boot functionality.

Prerequisites

- You are preparing network installation over the IPv4 protocol.
- Find the network addresses of the server.
  In the following examples, the server has a network interface **enp1s0** with this configuration:

  IPv4 address

  192.168.124.2/24

  IPv4 gateway + DNS server

  192.168.124.1

Procedure

1. Install the DHCP server:

   ```
   # dnf install kea
   ```

2. Set up a DHCPv4 server. Enter the following configuration in the **/etc/kea/kea-dhcp4.conf** file. Replace the addresses to match your network card.

   ```
   {
     "Dhcp4": {
       "interfaces-config": {
         "interfaces": [ "enp1s0" ]
       },
       "subnet4": [
         {
           "id": 1,
           "subnet": "192.168.124.0/24",
           "pools": [
             {
               "pool": "192.168.124.100 - 192.168.124.200"
             }
           ],
           "option-data": [
             {
               "space": "dhcp4",
               "name": "routers",
               "code": 3,
               "data": "192.168.124.10"
             },
             {
               "space": "dhcp4",
               "name": "domain-name-servers",
               "code": 6,
               "data": "192.168.124.10"
             }
           ]
         }
       ],
   ```

```
    "client-classes": [
      {
        "name": "uefi PXE Clients",
        "test": "substring(option[60].hex,0,9) == 'PXEClient' and option[93].hex == 0x0007",
        "next-server": "192.168.124.2",
        "boot-file-name": "/uefi/BOOT/BOOTX64.EFI"
      },
      {
        "name": "bios PXE Clients",
        "test": "substring(option[60].hex,0,9) == 'PXEClient' and option[93].hex == 0x0000",
        "next-server": "192.168.124.2",
        "boot-file-name": "syslinux/pxelinux.0"
      },
      {
        "name": "uefi HTTP Clients",
        "test": "substring(option[60].hex,0,10) == 'HTTPClient' and option[93].hex == 0x0007",
        "option-data": [
          {
            "space": "dhcp4",
            "name": "vendor-class-identifier",
            "code": 60,
            "data": "HTTPClient"
          }
        ],
        "next-server": "192.168.124.2",
        "boot-file-name": "http://192.168.124.2/uefi/BOOT/BOOTX64.EFI"
      }
    ]
  }
}
```

3. Start the DHCPv4 service:

```
# systemctl enable --now kea-dhcp4
```

6.3. CONFIGURING THE DHCPV6 SERVER FOR NETWORK BOOT

Enable the DHCP version 6 (DHCPv4) service on your server, so that it can provide network boot functionality.

Prerequisites

- You are preparing network installation over the IPv6 protocol.
- Find the network addresses of the server.
  In the following examples, the server has a network interface **enp1s0** with this configuration:

  IPv6 address

  fd33::2/64

  IPv6 gateway

  fd33::1

Procedure

1. Install the DHCP server:

```
# dnf install kea
```

2. Set up a DHCPv6 server. Enter the following configuration in the **/etc/kea/kea-dhcp6.conf** file. Replace the addresses to match your network card.

```
{
  "Dhcp6": {
    "interfaces-config": {
      "interfaces": [ "enp1s0" ]
    },
    "subnet6": [
      {
        "id": 1,
        "subnet": "fd33::/64",
        "interface": "enp1s0",
        "pools": [
          {
            "pool": "fd33::100-fd33::200"
          }
        ]
      }
    ],
    "client-classes": [
      {
        "name": "uefi PXE Clients",
        "test": "substring(option[16].hex,6,9) == 'PXEClient' and substring(option[16].hex,21,5)
== '00007'",
        "option-data": [
          {
            "space": "dhcp6",
            "name": "bootfile-url",
            "code": 59,
            "data": "tftp://[fd33::2]/uefi/BOOT/BOOTX64.EFI"
          }
        ]
      },
      {
        "name": "bios PXE Clients",
        "test": "substring(option[16].hex,6,9) == 'PXEClient' and substring(option[16].hex,21,5)
== '00000'",
        "option-data": [
          {
            "space": "dhcp6",
            "name": "bootfile-url",
            "code": 59,
            "data": "tftp://[fd33::2]/syslinux/pxelinux.0"
          }
        ]
      },
      {
        "name": "uefi HTTP Clients",
        "test": "substring(option[16].hex,6,10) == 'HTTPClient' and
substring(option[16].hex,22,5) == '00007'",
        "option-data": [
          {
            "space": "dhcp6",
            "name": "bootfile-url",
            "code": 59,
            "data": "http://[fd33::2]/uefi/BOOT/BOOTX64.EFI"
          }
        ]
      }
    ]
  }
}
```

3. Start the DHCPv6 service:

```
# systemctl enable --now kea-dhcp6
```

4. If DHCPv6 packets are dropped by the RP filter in the firewall, check its log. If the log contains the **rpfilter_DROP** entry, disable the filter using the following configuration in the **/etc/firewalld/firewalld.conf** file:

```
IPv6_rpfilter=no
```

6.4. CONFIGURING THE HTTP SERVER FOR HTTP BOOT

You must install and enable the **httpd** service on your server so that the server can provide HTTP boot resources on your network.

Prerequisites

- Find the network addresses of the server.
  In the following examples, the server has a network card with the **192.168.124.2** IPv4 address.

Procedure

1. Install the HTTP server:

```
# dnf install httpd
```

2. Create the **/var/www/html/redhat** directory:

```
# mkdir -p /var/www/html/redhat
```

3. Download the RHEL DVD ISO file. See All Red Hat Enterprise Linux Downloads .

4. Create a mount point for the ISO file:

```
# mkdir -p /var/www/html/redhat/iso
```

5. Mount the ISO file:

```
# mount -o loop,ro -t iso9660 path-to-RHEL-DVD.iso /var/www/html/redhat/iso
```

6. Copy the boot loader, kernel, and **initramfs** from the mounted ISO file into your HTML directory:

   ```
   # cp -r /var/www/html/redhat/iso/images /var/www/html/redhat
   # cp -r /var/www/html/redhat/iso/EFI /var/www/html/redhat
   ```

7. Make the boot loader configuration editable and ensure the boot files are owned by the user running httpd server (apache):

   ```
   # chmod 644 /var/www/html/redhat/EFI/BOOT/grub.cfg
   # chown -R apache:apache /var/www/html/redhat/EFI
   ```

8. Edit the **/var/www/html/redhat/EFI/BOOT/grub.cfg** file and replace its content with the following:

   ```
   set default="1"

   function load_video {
     insmod efi_gop
     insmod efi_uga
     insmod video_bochs
     insmod video_cirrus
     insmod all_video
   }

   load_video
   set gfxpayload=keep

   set timeout=60

   menuentry 'Install Red Hat Enterprise Linux 10.0' --class fedora --class gnu-linux --class gnu
   --class os {
       linuxefi /redhat/images/pxeboot/vmlinuz inst.repo=http://192.168.124.2/redhat/iso quiet
       initrdefi /redhat/images/pxeboot/initrd.img
   }
   submenu 'Troubleshooting -->' {
       menuentry 'Install Red Hat Enterprise Linux 10.0 in text mode' --class fedora --class gnu-
   linux --class gnu --class os {
               linuxefi /redhat/images/pxeboot/vmlinuz inst.repo=http://192.168.124.2/redhat/iso
   inst.text quiet
               initrdefi /redhat/images/pxeboot/initrd.img
       }
       menuentry 'Rescue a Red Hat Enterprise Linux system' --class fedora --class gnu-linux --
   class gnu --class os {
               linuxefi /redhat/images/pxeboot/vmlinuz inst.stage2=http://192.168.124.2/redhat/iso
   inst.rescue quiet
               initrdefi /redhat/images/pxeboot/initrd.img
       }
   }
   ```

   In this file, update the following strings:

   Install Red Hat Enterprise Linux 10.0

   Edit the version number to match the version of RHEL that you downloaded.

192.168.124.2
Replace with the IP address of your server.

9. Open ports in the firewall to allow HTTP (80), DHCP (67, 68) and DHCPv6 (546, 547) traffic:

   ```
   # firewall-cmd --zone public \
                   --add-port={80/tcp,67/udp,68/udp,546/udp,547/udp}
   ```

   This command enables temporary access until the next server reboot.

10. Optional: To enable permanent access, add the **--permanent** option to the command.

11. Reload firewall rules:

    ```
    # firewall-cmd --reload
    ```

12. Start the HTTP server:

    ```
    # systemctl enable --now httpd
    ```

13. Make the **html** directory and its content readable and executable:

    ```
    # chmod -cR u=rwX,g=rX,o=rX /var/www/html
    ```

14. Restore the SELinux context of the **html** directory:

    ```
    # restorecon -FvvR /var/www/html
    ```

CHAPTER 7. PREPARING A PXE INSTALLATION SOURCE

You must configure TFTP and DHCP on a PXE server to enable PXE boot and network installation.

7.1. NETWORK INSTALLATION OVERVIEW

A network installation allows you to install Red Hat Enterprise Linux to a system that has access to an installation server. At a minimum, two systems are required for a network installation:

Server

A system running a DHCP server, an HTTP, HTTPS, FTP, or NFS server, and in the PXE boot case, a TFTP server. Although each server can run on a different physical system, the procedures in this section assume a single system is running all servers.

Client

The system to which you are installing Red Hat Enterprise Linux. Once installation starts, the client queries the DHCP server, receives the boot files from the HTTP or TFTP server, and downloads the installation environment image from the HTTP, HTTPS, FTP or NFS server. Unlike other installation methods, the client does not require any physical boot media for the installation to start.

To boot a client from the network, enable network boot in the firmware or in a quick boot menu on the client. On some hardware, the option to boot from a network might be disabled, or not available.

The workflow to prepare for an installation of Red Hat Enterprise Linux from a network by using HTTP or PXE is as follows:

1. Export the installation ISO image or the installation tree to an NFS, HTTPS, HTTP, or FTP server.
2. Configure the HTTP or TFTP server and DHCP server, and start the HTTP or TFTP service on the server.
3. Boot the client and start the installation.

You can choose between the following network boot protocols:

HTTP

Use the HTTP boot if your client UEFI supports it. HTTP boot is usually more reliable.

PXE (TFTP)

PXE boot is more widely supported by client systems, but sending the boot files over TFTP protocol might be slow and result in timeout failures.

Additional resources

- Preparing network based repositories
- Red Hat Satellite product documentation

7.2. CONFIGURING THE DHCPV4 SERVER FOR NETWORK BOOT

Enable the DHCP version 4 (DHCPv4) service on your server, so that it can provide network boot functionality.

Prerequisites

- You are preparing network installation over the IPv4 protocol.
- Find the network addresses of the server.
  In the following examples, the server has a network interface **enp1s0** with this configuration:

  IPv4 address

  192.168.124.2/24

  IPv4 gateway + DNS server

  192.168.124.1

Procedure

1. Install the DHCP server:

```
# dnf install kea
```

2. Set up a DHCPv4 server. Enter the following configuration in the **/etc/kea/kea-dhcp4.conf** file. Replace the addresses to match your network card.

```
{
  "Dhcp4": {
    "interfaces-config": {
      "interfaces": [ "enp1s0" ]
    },
    "subnet4": [
      {
        "id": 1,
        "subnet": "192.168.124.0/24",
        "pools": [
          {
            "pool": "192.168.124.100 - 192.168.124.200"
          }
        ],
        "option-data": [
          {
            "space": "dhcp4",
            "name": "routers",
            "code": 3,
            "data": "192.168.124.10"
          },
          {
            "space": "dhcp4",
            "name": "domain-name-servers",
            "code": 6,
            "data": "192.168.124.10"
          }
        ]
      }
    ],
    "client-classes": [
      {
        "name": "uefi PXE Clients",
        "test": "substring(option[60].hex,0,9) == 'PXEClient' and option[93].hex == 0x0007",
          "next-server": "192.168.124.2",
          "boot-file-name": "/uefi/BOOT/BOOTX64.EFI"
        },
        {
          "name": "bios PXE Clients",
          "test": "substring(option[60].hex,0,9) == 'PXEClient' and option[93].hex == 0x0000",
          "next-server": "192.168.124.2",
          "boot-file-name": "syslinux/pxelinux.0"
        },
        {
          "name": "uefi HTTP Clients",
          "test": "substring(option[60].hex,0,10) == 'HTTPClient' and option[93].hex == 0x0007",
          "option-data": [
            {
              "space": "dhcp4",
              "name": "vendor-class-identifier",
              "code": 60,
              "data": "HTTPClient"
            }
          ],
          "next-server": "192.168.124.2",
          "boot-file-name": "http://192.168.124.2/uefi/BOOT/BOOTX64.EFI"
        }
      ]
    }
  }
```

3. Start the DHCPv4 service:

```
# systemctl enable --now kea-dhcp4
```

7.3. CONFIGURING THE DHCPV6 SERVER FOR NETWORK BOOT

Enable the DHCP version 6 (DHCPv4) service on your server, so that it can provide network boot functionality.

Prerequisites

- You are preparing network installation over the IPv6 protocol.
- Find the network addresses of the server.
  In the following examples, the server has a network interface **enp1s0** with this configuration:

  IPv6 address

  fd33::2/64

  IPv6 gateway

  fd33::1

Procedure

1. Install the DHCP server:

```
# dnf install kea
```

2. Set up a DHCPv6 server. Enter the following configuration in the **/etc/kea/kea-dhcp6.conf** file. Replace the addresses to match your network card.

```
{
  "Dhcp6": {
    "interfaces-config": {
      "interfaces": [ "enp1s0" ]
    },
    "subnet6": [
      {
        "id": 1,
        "subnet": "fd33::/64",
        "interface": "enp1s0",
        "pools": [
          {
            "pool": "fd33::100-fd33::200"
          }
        ]
      }
    ],
    "client-classes": [
      {
        "name": "uefi PXE Clients",
        "test": "substring(option[16].hex,6,9) == 'PXEClient' and substring(option[16].hex,21,5)
== '00007'",
        "option-data": [
          {
            "space": "dhcp6",
            "name": "bootfile-url",
            "code": 59,
            "data": "tftp://[fd33::2]/uefi/BOOT/BOOTX64.EFI"
          }
        ]
      },
      {
        "name": "bios PXE Clients",
        "test": "substring(option[16].hex,6,9) == 'PXEClient' and substring(option[16].hex,21,5)
== '00000'",
        "option-data": [
          {
            "space": "dhcp6",
            "name": "bootfile-url",
            "code": 59,
            "data": "tftp://[fd33::2]/syslinux/pxelinux.0"
          }
        ]
      },
      {
        "name": "uefi HTTP Clients",
        "test": "substring(option[16].hex,6,10) == 'HTTPClient' and
substring(option[16].hex,22,5) == '00007'",
        "option-data": [
          {
            "space": "dhcp6",
            "name": "bootfile-url",
            "code": 59,
              "data": "http://[fd33::2]/uefi/BOOT/BOOTX64.EFI"
            }
          ]
        }
      ]
    }
  }
```

3. Start the DHCPv6 service:

   ```
   # systemctl enable --now kea-dhcp6
   ```

4. If DHCPv6 packets are dropped by the RP filter in the firewall, check its log. If the log contains the **rpfilter_DROP** entry, disable the filter using the following configuration in the **/etc/firewalld/firewalld.conf** file:

   ```
   IPv6_rpfilter=no
   ```

7.4. CONFIGURING A TFTP SERVER FOR BIOS-BASED CLIENTS

You must configure a TFTP server and DHCP server and start the TFTP service on the PXE server for BIOS-based AMD and Intel 64-bit systems to enable automated network booting.

Procedure

1. As root, install the following package.

   ```
   # dnf install tftp-server
   ```

2. Allow incoming connections to the **tftp service** in the firewall:

   ```
   # firewall-cmd --add-service=tftp
   ```

   This command enables temporary access until the next server reboot.

3. Optional: To enable permanent access, repeat the previous command together with the **--permanent** option.
   Depending on the location of the installation ISO file, you might have to allow incoming connections for HTTP or other services.

4. Install **syslinux-tftpboot** package:

   ```
   # dnf -y install syslinux-tftpboot
   ```

5. Create a **pxelinux** directory in the **/var/lib/tftpboot** directory and copy all the files provided by the **syslinux-tftpboot** package (located in **/tftpboot** directory) into the **pxelinux** directory:

   ```
   # mkdir /var/lib/tftpboot/pxelinux
   # cp /tftpboot/* /var/lib/tftpboot/pxelinux
   ```

6. Create the directory **pxelinux.cfg** in the **pxelinux** directory:

```
# mkdir /var/lib/tftpboot/pxelinux/pxelinux.cfg
```

7. Create a configuration file named **default** and add it to the **pxelinux.cfg** directory as shown in the following example:

```
DEFAULT menu.c32

MENU TITLE Red Hat Enterprise Linux 10.0 Installation
TIMEOUT 600

LABEL default
 MENU LABEL ^Install Red Hat Enterprise Linux 10.0
 KERNEL /pxelinux/images/RHEL-10/vmlinuz
 APPEND initrd=/pxelinux/images/RHEL-10/initrd.img
inst.repo=http://192.168.124.2/redhat/iso quiet

LABEL text
 MENU LABEL Install Red Hat Enterprise Linux 10.0 in ^text mode
 KERNEL /pxelinux/images/RHEL-10/vmlinuz
 APPEND initrd=/pxelinux/images/RHEL-10/initrd.img
inst.repo=http://192.168.124.2/redhat/iso inst.text quiet

LABEL rescue
 MENU LABEL ^Rescue a Red Hat Enterprise Linux system
 KERNEL /pxelinux/images/RHEL-10/vmlinuz
 APPEND initrd=/pxelinux/images/RHEL-10/initrd.img
inst.stage2=http://192.168.124.2/redhat/iso inst.rescue quiet

LABEL local
 MENU LABEL Boot from ^local drive
 LOCALBOOT 0xffff
```

- The installation program cannot boot without its runtime image. Use the **inst.stage2** boot option to specify location of the image. Alternatively, you can use the **inst.repo=** option to specify the image as well as the installation source.
- The installation source location used with **inst.repo** must contain a valid **.treeinfo** file.
- When you select the content of RHEL10 installation DVD as the installation source, the **.treeinfo** file points to the BaseOS and the AppStream repositories. You can use a single **inst.repo** option to load both repositories.

8. Create a subdirectory to store the boot image files in the **/var/lib/tftpboot** directory, and copy the boot image files to the directory. In this example, the directory is **/var/lib/tftpboot/pxelinux/images/RHEL-10**:

```
# mkdir -p /var/lib/tftpboot/pxelinux/images/RHEL-10
# cp /path_to_x86_64_images/pxeboot/{vmlinuz,initrd.img}
/var/lib/tftpboot/pxelinux/images/RHEL-10/
```

9. Start and enable the **tftp.socket** service:

```
# systemctl enable --now tftp.socket
```

The PXE boot server is now ready to serve PXE clients. You can start the client, which is the system to which you are installing Red Hat Enterprise Linux. Select **PXE Boot** or **Network Boot** when prompted to specify a boot source, and start the network installation.

7.5. CONFIGURING A TFTP SERVER FOR UEFI-BASED CLIENTS

You must configure a TFTP server and DHCP server and start the TFTP service on the PXE server for UEFI-based AMD64, Intel 64, and 64-bit ARM systems.

IMPORTANT

Red Hat Enterprise Linux 10 UEFI PXE boot supports, apart from the default grub menu file (grub.cfg), a lowercase file format for a MAC-based grub menu file. For example, the MAC address file format for grub2 can be **grub.cfg-01-aa-bb-cc-dd-ee-ff**

Procedure

1. As root, install the following package.

```
# dnf install tftp-server
```

2. Allow incoming connections to the **tftp service** in the firewall:

```
# firewall-cmd --add-service=tftp
```

This command enables temporary access until the next server reboot.

3. Optional: To enable permanent access, repeat the previous command together with the **--permanent** option.
Depending on the location of the installation ISO file, you might have to allow incoming connections for HTTP or other services.

4. Access the EFI boot image files from the DVD ISO image:

```
# mount -t iso9660 /path_to_image/name_of_image.iso /mount_point -o loop,ro
```

5. Copy the EFI boot images from the DVD ISO image:

```
# mkdir /var/lib/tftpboot/redhat
# cp -r /mount_point/EFI /var/lib/tftpboot/redhat/
```

6. Create a subdirectory to store the boot image files in the **/var/lib/tftpboot/** directory, and copy the boot image files to the directory. In this example, the directory is **/var/lib/tftpboot/images/RHEL-10/**:

```
# mkdir -p /var/lib/tftpboot/images/RHEL-10/
# cp /mount_point/images/pxeboot/{vmlinuz,initrd.img} /var/lib/tftpboot/images/RHEL-10/
# umount /mount_point
```

7. Fix the permissions of the copied files:

```
# chmod -R 755 /var/lib/tftpboot/redhat/
```

8. Replace the content of **/var/lib/tftpboot/redhat/EFI/BOOT/grub.cfg** with the following example:

```
set default="1"
set timeout=60

function load_video {
  insmod efi_gop
  insmod efi_uga
  insmod video_bochs
  insmod video_cirrus
  insmod all_video
}

load_video
set gfxpayload=keep

menuentry 'Install Red Hat Enterprise Linux 10.0' --class fedora --class gnu-linux --class gnu
--class os {
     linuxefi /images/RHEL-10/vmlinuz inst.repo=http://192.168.124.2/redhat/iso quiet
     initrdefi /images/RHEL-10/initrd.img
}
submenu 'Troubleshooting -->' {
     menuentry 'Install Red Hat Enterprise Linux 10.0 in text mode' --class fedora --class gnu-
linux --class gnu --class os {
             linuxefi /images/RHEL-10/vmlinuz inst.repo=http://192.168.124.2/redhat/iso inst.text
quiet
             initrdefi /images/RHEL-10/initrd.img
     }
     menuentry 'Rescue a Red Hat Enterprise Linux system' --class fedora --class gnu-linux --
class gnu --class os {
             linuxefi /images/RHEL-10/vmlinuz inst.stage2=http://192.168.124.2/redhat/iso
inst.rescue quiet
             initrdefi /images/RHEL-10/initrd.img
     }
}
```

- The installation program cannot boot without its runtime image. Use the **inst.stage2** boot option to specify location of the image. Alternatively, you can use the **inst.repo=** option to specify the image as well as the installation source.

- The installation source location used with **inst.repo** must contain a valid **.treeinfo** file.

- When you select the RHEL10 installation DVD as the installation source, the **.treeinfo** file points to the BaseOS and the AppStream repositories. You can use a single **inst.repo** option to load both repositories.

9. Start and enable the **tftp.socket** service:

```
# systemctl enable --now tftp.socket
```

The PXE boot server is now ready to serve PXE clients. You can start the client, which is the system to which you are installing Red Hat Enterprise Linux. Select **PXE Boot** or **Network Boot** when prompted to specify a boot source, and start the network installation.

Additional resources

- Using the Shim Program

CHAPTER 8. PREPARING A RHEL INSTALLATION ON 64-BIT IBM Z

You can install Red Hat Enterprise Linux on the 64-bit IBM Z architecture.

8.1. PLANNING FOR INSTALLATION ON 64-BIT IBM Z

Red Hat Enterprise Linux 10 runs on **IBM z14** or **IBM LinuxONE II** systems, or later.

The installation process assumes that you are familiar with the 64-bit IBM Z and can set up *logical partitions* (LPARs) and z/VM guest virtual machines. For installation of Red Hat Enterprise Linux on 64-bit IBM Z, Red Hat supports Direct Access Storage Device (DASD), SCSI disk devices attached over Fiber Channel Protocol (FCP), and **virtio-blk** and **virtio-scsi** devices. When using FCP devices, use them in multipath configuration for better reliability.

IMPORTANT

DASDs are disks that allow a maximum of three partitions per device. For example, **dasda** can have partitions **dasda1**, **dasda2**, and **dasda3**.

Pre-installation decisions

- Whether the operating system is to be run on an LPAR, KVM, or as a z/VM guest operating system.
- Network configuration. Red Hat Enterprise Linux 10 for 64-bit IBM Z supports the following network devices:
    - Real and virtual *Open Systems Adapter* (OSA)
    - Real and virtual HiperSockets
    - **virtio-net** devices
    - RDMA over Converged Ethernet (RoCE)
- Ensure you select machine type as ***ESA*** for your z/VM VMs, because selecting any other machine types might prevent RHEL from installing. See the *IBM documentation*.

NOTE

When initializing swap space on a Fixed Block Architecture (FBA) DASD by using the **SWAPGEN** utility, the **FBAPART** option must be used.

Additional resources

- RHEL Technology Capabilities and Limits
- IBM documentation
- Secure boot for Linux on IBM Z
- IBM installation documentation

- Red Hat Ecosystem Catalog

8.2. BOOT MEDIA COMPATIBILITY FOR IBM Z SERVERS

The following table provides detailed information about the supported boot media options for installing Red Hat Enterprise Linux (RHEL) on 64-bit IBM Z servers. It outlines the compatibility of each boot medium with different system types and indicates whether the **zipl** boot loader is used. This information helps you determine the most suitable boot medium for your specific environment.

System type / Boot media	Uses zipl boot loader	z/VM	KVM	LPAR
z/VM Reader	No	Yes	N/A	N/A
SE or HMC (remote SFTP, FTPS, FTP server, DVD)	No	N/A	N/A	Yes
DASD	Yes	Yes	Yes	Yes
FCP SCSI LUNs	Yes	Yes	Yes	Yes
FCP SCSI DVD	Yes	Yes	Yes	Yes

N/A indicates that the boot medium is not applicable for the specified system type.

8.3. SUPPORTED ENVIRONMENTS AND COMPONENTS FOR IBM Z SERVERS

The following tables provide information about the supported environments, network devices, machine types, and storage types for different system types when installing Red Hat Enterprise Linux (RHEL) on 64-bit IBM Z servers. Use these tables to identify the compatibility of various components with your specific system configuration.

Table 8.1. Network device compatibility for system types

Network device	z/VM	KVM	LPAR
Open Systems Adapter (OSA)	Yes	N/A	Yes
HiperSockets	Yes	N/A	Yes
virtio-net	N/A	Yes	N/A
RDMA over Converged Ethernet (RoCE)	Yes	Yes	Yes

N/A indicates that the component is not applicable for the specified system type.

Table 8.2. Machine type compatibility for system types

Machine type	z/VM	KVM	LPAR
ESA	Yes	N/A	N/A
s390-virtio-ccw	N/A	Yes	N/A

N/A indicates that the component is not applicable for the specified system type.

Table 8.3. Storage type compatibility for system types

Storage type	z/VM	KVM	LPAR
DASD	Yes	Yes	Yes
FCP SCSI	Yes	Yes[a]	Yes
virtio-blk	N/A	Yes	N/A
[a] Conditional support based on configuration			

N/A indicates that the component is not applicable for the specified system type.

8.4. OVERVIEW OF INSTALLATION PROCESS ON 64-BIT IBM Z SERVERS

You can install Red Hat Enterprise Linux on 64-bit IBM Z interactively or in unattended mode. Installation on 64-bit IBM Z differs from other architectures as it is typically performed over a network, and not from local media. The installation consists of three phases:

1. Booting the installation.
    - Connect to the mainframe.
    - Customize the boot parameters.
    - Perform an initial program load (IPL), or boot from the media containing the installation program.
2. Connecting to the installation system.
    - From a local machine, connect to the remote 64-bit IBM Z system by using SSH, and start the installation program by using Remote Desktop Protocol (RDP).
3. Completing the installation by using the RHEL installation program.

8.5. BOOT MEDIA FOR INSTALLING RHEL ON 64-BIT IBM Z SERVERS

After establishing a connection with the mainframe, you need to perform an initial program load (IPL), or boot, from the medium containing the installation program. In general, any method may be used to boot the Linux installation system, which consists of a kernel (**kernel.img**) and initial RAM disk (**initrd.img**) with parameters in the **generic.prm** file supplemented by user defined parameters. Additionally, a **generic.ins** file is loaded which determines file names and memory addresses for the initrd, kernel and **generic.prm**.

The Linux installation system is also called the *installation program* in this book.

You can use the following boot media only if Linux is to run as a guest operating system under z/VM:

- z/VM reader

You can use the following boot media only if Linux is to run in LPAR mode:

- SE or HMC through a remote SFTP, FTPS or FTP server
- SE or HMC DVD

You can use the following boot media for both z/VM and LPAR:

- DASD
- SCSI disk device that is attached through an FCP channel

If you use DASD or an FCP-attached SCSI disk device as boot media, you must have a configured **zipl** boot loader.

8.6. CUSTOMIZATION OF BOOT PARAMETERS

Before the installation begins, you must configure some mandatory boot parameters. When installing through **z/VM**, these parameters must be configured before you boot into the **generic.prm** file. When installing on an **LPAR**, the **rd.cmdline** parameter is set to **ask** by default. It means that you will be given a prompt to enter these boot parameters. In both cases, the required parameters are the same.

All network configuration can either be specified by using a parameter file, or at the prompt.

Installation source

An installation source must always be configured.

You can use the **inst.repo** option to specify the package source for the installation.

Network devices

Network configuration must be provided if network access will be required during the installation. If you plan to perform an unattended (Kickstart-based) installation by using only local media such as a disk, network configuration can be omitted.

ip=

You can use the **ip=** option for basic network configuration, and other options as required.

rd.znet=

Also, you can use the **rd.znet=** kernel option, which takes a network protocol type. It is a comma delimited list of sub-channels, and, optionally, comma delimited **sysfs** parameter and value pairs for qeth devices. This parameter can be specified multiple times to activate multiple network devices.
For example:

```
rd.znet=qeth,0.0.0600,0.0.0601,0.0.0602,layer2=1,portno=<number>
```

- When specifying multiple **rd.znet** boot options, only the last one is passed on to the kernel command line of the installed system. This does not affect the networking of the system since all network devices configured during installation are properly activated and configured at boot.

  The qeth device driver assigns the same interface name for Ethernet and Hipersockets devices: **enc*<device number>***. The bus ID is composed of the channel subsystem ID, subchannel set ID, and device number, separated by dots; the device number is the last part of the bus ID, without leading zeroes and dots. For example, the interface name will be **enca00** for a device with the bus ID **0.0.0a00**.

Storage devices

At least one storage device must always be configured for text mode installations.
The **rd.dasd=** option takes a Direct Access Storage Device (DASD) adapter device bus identifier. For multiple DASDs, specify the parameter multiple times, or use a comma separated list of bus IDs. To specify a range of DASDs, specify the first and the last bus ID.

For example:

```
rd.dasd=0.0.0200 rd.dasd=0.0.0202(ro),0.0.0203(ro:failfast),0.0.0205-0.0.0207
```

All DASDs that were specified using the **rd.dasd=** kernel argument during the installation process are automatically activated on the installed system. This activation occurs regardless of whether the DASDs were used for partitioning during the installation. All disks made visible to the installer via **rd.dasd=** will be online and available on the first boot of the installed system.

The **rd.zfcp=** option takes a SCSI over FCP (zFCP) adapter device bus identifier, a target world wide port name (WWPN), and an FCP LUN, then activates one path to a SCSI disk. This parameter needs to be specified at least twice to activate multiple paths to the same disk. This parameter can be specified multiple times to activate multiple disks, each with multiple paths.

Since 10, a target world wide port name (WWPN) and an FCP LUN have to be provided only if the **zFCP** device is not configured in NPIV mode or when **auto LUN** scanning is disabled by the **zfcp.allow_lun_scan=0** kernel module parameter. It provides access to all SCSI devices found in the storage area network attached to the FCP device with the specified bus ID. This parameter needs to be specified at least twice to activate multiple paths to the same disks.

```
rd.zfcp=0.0.4000,0x5005076300C213e9,0x5022000000000000
rd.zfcp=0.0.4000
```

Kickstart options

If you are using a Kickstart file to perform an automatic installation, you must always specify the location of the Kickstart file by using the **inst.ks=** option. For an unattended, fully automatic Kickstart installation, the **inst.cmdline** option is also useful. An example customized **generic.prm** file containing all mandatory parameters look similar to the following example:

```
ro ramdisk_size=40000 cio_ignore=all,!condev
inst.repo=http://example.com/path/to/repository
rd.znet=qeth,0.0.0600,0.0.0601,0.0.0602,layer2=1,portno=0,portname=foo
ip=192.168.17.115::192.168.17.254:24:foobar.systemz.example.com:enc600:none
nameserver=192.168.17.1
rd.dasd=0.0.0200 rd.dasd=0.0.0202
rd.zfcp=0.0.4000,0x5005076300c213e9,0x5022000000000000
rd.zfcp=0.0.5000,0x5005076300dab3e9,0x5022000000000000
inst.ks=http://example.com/path/to/kickstart
```

Some installation methods also require a file with a mapping of the location of installation data in the file system of the HMC DVD or FTP server and the memory locations where the data is to be copied.

The file is typically named **generic.ins**, and contains file names for the initial RAM disk, kernel image, and parameter file (**generic.prm**) and a memory location for each file. An example **generic.ins** will look similar to the following example:

```
images/kernel.img 0x00000000
images/initrd.img 0x02000000
images/genericdvd.prm 0x00010480
images/initrd.addrsize 0x00010408
```

A valid **generic.ins** file is provided by Red Hat along with all other files required to boot the installer. Modify this file only if you want to, for example, load a different kernel version than default.

8.7. PREPARING AN INSTALLATION IN A Z/VM GUEST VIRTUAL MACHINE

Use the **x3270** or **c3270** terminal emulator, to log in to z/VM from other Linux systems, or use the IBM 3270 terminal emulator on the 64-bit IBM Z Hardware Management Console (HMC). If you are running Microsoft Windows operating system, there are several options available, and can be found through an internet search. A free native Windows port of **c3270** called **wc3270** also exists.

Ensure you select machine type as ***ESA*** for your z/VM VMs, because selecting any other machine types might prevent installing RHEL. See the *IBM documentation*.

Procedure

1. Log on to the z/VM guest virtual machine chosen for the Linux installation.

2. optional: If your 3270 connection is interrupted and you cannot log in again because the previous session is still active, you can replace the old session with a new one by entering the following command on the z/VM logon screen:

   ```
   logon user here
   ```

   Replace *user* with the name of the z/VM guest virtual machine. Depending on whether an external security manager, for example RACF, is used, the logon command might vary.

3. If you are not already running **CMS** (single-user operating system shipped with z/VM) in your guest, boot it now by entering the command:

   ```
   cp ipl cms
   ```

4. Be sure not to use CMS disks such as your A disk (often device number 0191) as installation targets. To find out which disks are in use by CMS, use the following query:

   ```
   query disk
   ```

5. You can use the following CP (z/VM Control Program, which is the z/VM hypervisor) query commands to find out about the device configuration of your z/VM guest virtual machine:

    a. Query the available main memory, which is called *storage* in 64-bit IBM Z terminology. Your guest should have at least 1 GiB of main memory.

    ```
    cp query virtual storage
    ```

    b. Query available network devices by type:

    osa

    OSA - CHPID type OSD, real or virtual (VSWITCH or GuestLAN), both in QDIO mode

    hsi

    HiperSockets - CHPID type IQD, real or virtual (GuestLAN type Hipers)

    lcs

    LCS - CHPID type OSE

    For example, to query all of the network device types mentioned above, run:

    ```
    cp query virtual osa
    ```

    c. Query available DASDs. Only those that are flagged **RW** for read-write mode can be used as installation targets:

    ```
    cp query virtual dasd
    ```

    d. Query available FCP devices (vHBAs):

    ```
    cp query virtual fcp
    ```

8.8. PARAMETERS AND CONFIGURATION FILES ON 64-BIT IBM Z

You can customize and configure RHEL by using the parameters and configuration files on 64-bit IBM Z.

8.8.1. Required configuration file parameters on 64-bit IBM Z

Several parameters are required and must be included in the parameter file. These parameters are also provided in the file **generic.prm** in directory **images/** of the installation DVD.

- **ro**
  Mounts the root file system, which is a RAM disk, read-only.

- **ramdisk_size=*size***
  Modifies the memory size reserved for the RAM disk to ensure that the Red Hat Enterprise Linux installation program fits within it. For example: **ramdisk_size=40000**.

The **generic.prm** file also contains the additional parameter cio_ignore=all,!condev. This setting speeds up boot and device detection on systems with many devices. The installation program transparently handles the activation of ignored devices.

8.8.2. 64-bit IBM z/VM configuration file

Under z/VM, you can use a configuration file on a CMS-formatted disk. The purpose of the CMS configuration file is to save space in the parameter file by moving the parameters that configure the initial network setup, the DASD, and the FCP specification out of the parameter file.

Each line of the CMS configuration file contains a single variable and its associated value, in the following shell-style syntax: ***variable=value***.

You must also add the **CMSDASD** and **CMSCONFFILE** parameters to the parameter file. These parameters point the installation program to the configuration file:

CMSDASD=*cmsdasd_address*

Where *cmsdasd_address* is the device number of a CMS-formatted disk that contains the configuration file. This is usually the CMS user's **A** disk.
For example: **CMSDASD=191**

CMSCONFFILE=*configuration_file*

Where *configuration_file* is the name of the configuration file. **This value must be specified in lower case.** It is specified in a Linux file name format: ***CMS_file_name.CMS_file_type***.
The CMS file **REDHAT CONF** is specified as **redhat.conf**. The CMS file name and the file type can each be from one to eight characters that follow the CMS conventions.

For example: **CMSCONFFILE=redhat.conf**

8.8.3. Installation network, DASD and FCP parameters on 64-bit IBM Z

These parameters can be used to automatically set up the preliminary network, and can be defined in the CMS configuration file. These parameters are the only parameters that can also be used in a CMS configuration file. All other parameters in other sections must be specified in the parameter file.

NETTYPE="*type*"

Where *type* must be one of the following: **qeth**, **lcs**, or **ctc**. The default is **qeth**.
Choose **qeth** for:

- OSA-Express features
- HiperSockets
- Virtual connections on z/VM, including VSWITCH and Guest LAN
  Select **ctc** for:
- Channel-to-channel network connections

SUBCHANNELS="*device_bus_IDs*"

Where *device_bus_IDs* is a comma-separated list of two or three device bus IDs. The IDs must be specified in lowercase.
Provides required device bus IDs for the various network interfaces:

```
qeth: SUBCHANNELS="read_device_bus_id,write_device_bus_id,data_device_bus_id"
lcs or ctc: SUBCHANNELS="read_device_bus_id,write_device_bus_id"
```

For example (a sample qeth SUBCHANNEL statement):

```
SUBCHANNELS="0.0.f5f0,0.0.f5f1,0.0.f5f2"
```

PORTNO="*portnumber*"

You can add either **PORTNO="0"** (to use port 0) or **PORTNO="1"** (to use port 1 of OSA features with two ports per CHPID).

LAYER2="*value*"

Where *value* can be **0** or **1**.
Use **LAYER2="0"** to operate an OSA or HiperSockets device in layer 3 mode (**NETTYPE="qeth"**). Use **LAYER2="1"** for layer 2 mode. For virtual network devices under z/VM this setting must match the definition of the GuestLAN or VSWITCH to which the device is coupled.

To use network services that operate on layer 2 (the Data Link Layer or its MAC sublayer) such as DHCP, layer 2 mode is a good choice.

The qeth device driver default for OSA devices is now layer 2 mode. To continue using the previous default of layer 3 mode, set **LAYER2="0"** explicitly.

VSWITCH="*value*"

Where *value* can be **0** or **1**.
Specify **VSWITCH="1"** when connecting to a z/VM VSWITCH or GuestLAN, or **VSWITCH="0"** (or nothing at all) when using directly attached real OSA or directly attached real HiperSockets.

MACADDR="*MAC_address*"

If you specify **LAYER2="1"** and **VSWITCH="0"**, you can optionally use this parameter to specify a MAC address. Linux requires six colon-separated octets as pairs lower case hex digits - for example, **MACADDR=62:a3:18:e7:bc:5f**. This is different from the notation used by z/VM.
If you specify **LAYER2="1"** and **VSWITCH="1"**, you must not specify the **MACADDR**, because z/VM assigns a unique MAC address to virtual network devices in layer 2 mode.

CTCPROT="*value*"

Where *value* can be **0**, **1**, or **3**.
Specifies the CTC protocol for **NETTYPE="ctc"**. The default is **0**.

HOSTNAME="*string*"

Where *string* is the host name of the newly-installed Linux instance.

IPADDR="*IP*"

Where *IP* is the IP address of the new Linux instance.

NETMASK="*netmask*"

Where *netmask* is the netmask.
The netmask supports the syntax of a prefix integer (from 1 to 32) as specified in IPv4 *classless interdomain routing* (CIDR). For example, you can specify **24** instead of **255.255.255.0**, or **20** instead of **255.255.240.0**.

GATEWAY="*gw*"

Where *gw* is the gateway IP address for this network device.

MTU="*mtu*"

Where *mtu* is the *Maximum Transmission Unit* (MTU) for this network device.

DNS="*server1:server2:additional_server_terms:serverN*"

Where "*server1:server2:additional_server_terms:serverN*" is a list of DNS servers, separated by colons. For example:

```
DNS="10.1.2.3:10.3.2.1"
```

SEARCHDNS="*domain1:domain2:additional_dns_terms:domainN*"

Where "*domain1:domain2:additional_dns_terms:domainN*" is a list of the search domains, separated by colons. For example:

```
SEARCHDNS="subdomain.domain:domain"
```

You only need to specify **SEARCHDNS=** if you specify the **DNS=** parameter.

DASD=

Defines the DASD or range of DASDs to configure for the installation.
The installation program supports a comma-separated list of device bus IDs, or ranges of device bus IDs with the optional attributes **ro**, **diag**, **erplog**, and **failfast**. Optionally, you can abbreviate device bus IDs to device numbers with leading zeros stripped. Any optional attributes should be separated by colons and enclosed in parentheses. Optional attributes follow a device bus ID or a range of device bus IDs.

The only supported global option is **autodetect**. This does not support the specification of non-existent DASDs to reserve kernel device names for later addition of DASDs. Use persistent DASD device names such as **/dev/disk/by-path/name** to enable transparent addition of disks later. Other global options such as **probeonly**, **nopav**, or **nofcx** are not supported by the installation program.

Only specify those DASDs that need to be installed on your system. All unformatted DASDs specified here must be formatted after a confirmation later on in the installation program.

Add any data DASDs that are not needed for the root file system or the **/boot** partition after installation.

For example:

```
DASD="eb1c,0.0.a000-0.0.a003,eb10-eb14(diag),0.0.ab1c(ro:diag)"
```

FCP_*n*="*device_bus_ID* [*WWPN FCP_LUN*]"

For FCP-only environments, remove the **DASD=** option from the CMS configuration file to indicate no DASD is present.

```
FCP_n="device_bus_ID [WWPN FCP_LUN]"
```

Where:

- *n* is typically an integer value (for example **FCP_1** or **FCP_2**) but could be any string with alphabetic or numeric characters or underscores.

- *device_bus_ID* specifies the device bus ID of the FCP device representing the *host bus adapter* (HBA) (for example **0.0.fc00** for device fc00).
- *WWPN* is the world wide port name used for routing (often in conjunction with multipathing) and is as a 16-digit hex value (for example **0x50050763050b073d**).
- *FCP_LUN* refers to the storage logical unit identifier and is specified as a 16-digit hexadecimal value padded with zeroes to the right (for example **0x4020400100000000**).

NOTE

A target world wide port name (WWPN) and an FCP_LUN have to be provided if the **zFCP** device is not configured in NPIV mode, when auto LUN scanning is disabled by the **zfcp.allow_lun_scan=0** kernel module parameter or when installing RHEL-9.0 or older releases. Otherwise only the **device_bus_ID** value is mandatory.

- These variables can be used on systems with FCP devices to activate FCP LUNs such as SCSI disks. Additional FCP LUNs can be activated during the installation interactively or by means of a Kickstart file. An example value looks similar to the following:

```
FCP_1="0.0.fc00 0x50050763050b073d 0x4020400100000000"
FCP_2="0.0.4000"
```

  Each of the values used in the FCP parameters (for example **FCP_1** or **FCP_2**) are site-specific and are normally supplied by the FCP storage administrator.

8.8.4. Parameters for kickstart installations on 64-bit IBM Z

The following parameters can be defined in a parameter file but do not work in a CMS configuration file.

inst.ks=*URL*

References a Kickstart file, which usually resides on the network for Linux installations on 64-bit IBM Z. Replace *URL* with the full path including the file name of the Kickstart file. This parameter activates automatic installation with Kickstart.

inst.cmdline

This requires installation with a Kickstart file that answers all questions, because the installation program does not support interactive user input in cmdline mode. Ensure that your Kickstart file contains all required parameters before you use the **inst.cmdline** option. If a required command is missing, the installation will fail.

8.8.5. Additional Boot Parameters for 64-bit IBM Z

You can define the following parameters in a parameter file but they do not work in a CMS configuration file.

rd.live.check

Turns on testing of an ISO-based installation source; for example, when using **inst.repo=** with an ISO on local disk or mounted with NFS.

inst.nompath

Disables support for multipath devices.

inst.proxy=[*protocol*://][*username*[:*password*]@]*host*[:*port*]

Specify a proxy to use with installation over HTTP, HTTPS or FTP.

inst.rescue

Boot into a rescue system running from a RAM disk that can be used to fix and restore an installed system.

inst.stage2=*URL*

Specifies a path to a tree containing **install.img**, not to the **install.img** directly. Otherwise, follows the same syntax as **inst.repo=**. If **inst.stage2** is specified, it typically takes precedence over other methods of finding **install.img**. However, if **Anaconda** finds **install.img** on local media, the **inst.stage2** URL will be ignored.
If **inst.stage2** is not specified and **install.img** cannot be found locally, **Anaconda** looks to the location given by **inst.repo=** or **method=**.

If only **inst.stage2=** is given without **inst.repo=** or **method=**, **Anaconda** uses whatever repos the installed system would have enabled by default for installation.

Use the option multiple times to specify multiple HTTP, HTTPS or FTP sources. The HTTP, HTTPS or FTP paths are then tried sequentially until one succeeds:

```
inst.stage2=http://hostname/path_to_install_tree/
inst.stage2=http://hostname/path_to_install_tree/
inst.stage2=http://hostname/path_to_install_tree/
```

inst.syslog=*IP/hostname*[:*port*]

Sends log messages to a remote syslog server.

The boot parameters described here are the most useful for installations and trouble shooting on 64-bit IBM Z, but only a subset of those that influence the installation program.

8.8.6. Sample parameter file and CMS configuration file on 64-bit IBM Z

To change the parameter file, begin by extending the shipped **generic.prm** file.

Example of **generic.prm** file:

```
ro ramdisk_size=40000 cio_ignore=all,!condev
CMSDASD="191" CMSCONFFILE="redhat.conf"
inst.rdp
inst.repo=http://example.com/path/to/dvd-contents
```

Example of **redhat.conf** file configuring a QETH network device (pointed to by **CMSCONFFILE** in **generic.prm**):

```
NETTYPE="qeth"
SUBCHANNELS="0.0.0600,0.0.0601,0.0.0602"
PORTNAME="FOOBAR"
PORTNO="0"
LAYER2="1"
MACADDR="02:00:be:3a:01:f3"
HOSTNAME="foobar.systemz.example.com"
IPADDR="192.168.17.115"
```

```
NETMASK="255.255.255.0"
GATEWAY="192.168.17.254"
DNS="192.168.17.1"
SEARCHDNS="systemz.example.com:example.com"
DASD="200-203"
```

CHAPTER 9. AUTOMATED INSTALLATION WORKFLOW

You can perform the Kickstart installations by using a local DVD, a local disk, or an NFS, FTP, HTTP, or HTTPS server.

The high level overview of Kickstart usage is as follows:

1. Create a Kickstart file. You can write it by hand, copy a Kickstart file saved after a manual installation, or use an online generator tool to create the file, and edit it afterward. See Creating Kickstart files.

2. Make the Kickstart file available to the installation program on removable media, a disk or a network location by using an HTTP(S), FTP, or NFS server. See Adding the Kickstart file to a UEFI HTTP or PXE installation source or Making Kickstart files available to the RHEL installer .

3. Create the boot medium which will be used to begin the installation.

4. Make the installation source available to the installation program. See Creating installation sources for Kickstart installations.

5. Start the installation by using the boot medium and the Kickstart file. See Starting Kickstart installations.

If the Kickstart file contains all mandatory commands and sections, the installation starts and finishes automatically. If one or more of these mandatory parts are missing, user input is required to start the automated installation. If a critical error occurs, the installation is aborted and the user is informed about the error.

CHAPTER 10. CREATING KICKSTART FILES

You can create a Kickstart file by using the following different methods:

- Use the online Kickstart configuration tool.
- Copy the Kickstart file created as a result of installation.
- Write the entire Kickstart file manually.
- Convert the Red Hat Enterprise Linux 9 Kickstart file for Red Hat Enterprise Linux 10 installation.
  For more information about the conversion tool, see Kickstart generator lab .
- In case of virtual and cloud environments, create a custom system image by using Image Builder.

Some highly specific installation options can be configured only by manual editing of the Kickstart file.

10.1. CREATING A KICKSTART FILE WITH THE KICKSTART CONFIGURATION TOOL

Use the Red Hat Customer Portal Kickstart Generator to create automated installation files through a web-based interface. The Kickstart generator simplifies the creation of Kickstart files for automated RHEL deployments without requiring manual file editing or syntax knowledge.

Prerequisites

- You have a Red Hat Customer Portal account and an active Red Hat subscription.

Procedure

1. Open the Kickstart generator lab.
2. Click the **Go to Application** button to the left of the heading and wait for the next page to load.
3. Select **Red Hat Enterprise Linux 10** in the drop-down menu and wait for the page to update.
4. Describe the system to be installed by using the fields in the form.
   You can use the links on the left side of the form to quickly navigate between sections of the form.
5. To download the generated Kickstart file, click the red **Download** button at the top of the page.
   Your web browser saves the file.
6. Install the **pykickstart** package.

   ```
   # dnf install pykickstart
   ```

7. Run **ksvalidator** on your Kickstart file.

   ```
   $ ksvalidator -v RHEL10 /path/to/kickstart.ks
   ```

   Replace */path/to/kickstart.ks* with the path to the Kickstart file you want to verify.

The validation tool cannot guarantee the installation will be successful. It ensures only that the syntax is correct and that the file does not include deprecated options. It does not attempt to validate the **%pre**, **%post** and **%packages** sections of the Kickstart file.

10.2. CREATING A KICKSTART FILE BY PERFORMING A MANUAL INSTALLATION

You can use the Kickstart file created by a manual installation of Red Hat Enterprise Linux. After a graphical installation completes, all choices made during the installation are saved into a Kickstart file named **anaconda-ks.cfg**. This file is located in the **/root/** directory on the installed system.

You can use this file to reproduce the installation in the same way as before. Alternatively, copy this file, make any changes you need, and use the resulting configuration file for further installations.

Procedure

1. Install RHEL. For more details, see Interactively installing RHEL from installation media . During the installation, create a user with administrator privileges.
2. Finish the installation and reboot into the installed system.
3. Log into the system with the administrator account.
4. Copy the file **/root/anaconda-ks.cfg** to a location of your choice. The file contains information about users and passwords.
    - To display the file contents in terminal:

      ```
      # cat /root/anaconda-ks.cfg
      ```

      You can copy the output and save to another file of your choice.
    - To copy the file to another location, use the file manager. Remember to change permissions on the copy, so that the file can be read by non-root users.
5. Install the **pykickstart** package.

   ```
   # dnf install pykickstart
   ```
6. Run **ksvalidator** on your Kickstart file.

   ```
   $ ksvalidator -v RHEL10 /path/to/kickstart.ks
   ```

   Replace */path/to/kickstart.ks* with the path to the Kickstart file you want to verify.

   IMPORTANT

   The validation tool cannot guarantee the installation will be successful. It ensures only that the syntax is correct and that the file does not include deprecated options. It does not attempt to validate the **%pre**, **%post** and **%packages** sections of the Kickstart file.

10.3. CONVERTING A KICKSTART FILE FROM PREVIOUS RHEL INSTALLATION

You can use the Kickstart Converter tool to convert a RHEL 8 Kickstart file for use in a RHEL 9 installation or convert a RHEL 9 Kickstart file for use in RHEL 10. For more information about the tool and how to use it to convert a RHEL Kickstart file, see Kickstart converter.

Procedure

- After you prepare your kickstart file, install the **pykickstart** package.

  ```
  # dnf install pykickstart
  ```

- Run **ksvalidator** on your Kickstart file.

  ```
  $ ksvalidator -v RHEL10 /path/to/kickstart.ks
  ```

  Replace */path/to/kickstart.ks* with the path to the Kickstart file you want to verify.

IMPORTANT

The validation tool cannot guarantee the installation will be successful. It ensures only that the syntax is correct and that the file does not include deprecated options. It does not attempt to validate the **%pre**, **%post** and **%packages** sections of the Kickstart file.

10.4. CREATING A CUSTOM IMAGE BY USING IMAGE BUILDER

You can use Red Hat Image Builder to create a customized system image for virtual and cloud deployments.

For more information about creating customized images, using Image Builder, see the *Composing a customized RHEL system image* document.

CHAPTER 11. ADDING THE KICKSTART FILE TO A UEFI HTTP OR PXE INSTALLATION SOURCE

After your Kickstart file is ready, you can make it available for the installation on the destination system.

11.1. SHARING THE INSTALLATION FILES ON AN NFS SERVER

You can store Kickstart files on an NFS server to enable automated installations across multiple systems. It eliminates the need for physical media and provides centralized management of installation configurations for efficient system deployment over network file sharing.

Prerequisites

- You have administrator-level access to a server with Red Hat Enterprise Linux 10 on the local network.
- The system to be installed can connect to the server.

IMPORTANT

Ensure that you use different paths in **inst.ks** and **inst.repo**. When using NFS to host the Kickstart, you cannot use the same nfs share to host the installation source.

Procedure

1. Install the **nfs-utils** package by running the following command as root:

```
# dnf install nfs-utils
```

2. Copy the Kickstart file to a directory on the NFS server.

3. Open the **/etc/exports** file by using a text editor and add a line with the following syntax:

```
/exported_directory/ clients
```

   Replace */exported_directory/* with the full path to the directory holding the Kickstart file. Instead of *clients*, use the host name or IP address of the computer that is to be installed from this NFS server, the subnetwork from which all computers are to have access the ISO image, or the asterisk sign (*) if you want to allow any computer with network access to the NFS server to use the ISO image. See the *exports(5)* man page for detailed information about the format of this field. A basic configuration that makes the **/nfs/rhel10-kickstart/** directory available as read-only to all clients is:

```
/nfs/rhel10-kickstart/ *
```

4. Enable the NFS service in firewalld.

```
# firewall-cmd --permanent --add-service nfs
# firewall-cmd --permanent --add-service=mountd
# firewall-cmd --permanent --add-service=rpc-bind
# firewall-cmd --reload
```

5. Save the **/etc/exports** file and exit the text editor.

6. Start the nfs service:

```
# systemctl enable --now nfs-server.service
```

   If the service was running before you changed the **/etc/exports** file, enter the following command, in order for the running NFS server to reload its configuration:

```
# systemctl reload nfs-server.service
```

   The Kickstart file is now accessible over NFS and ready to be used for installation.

   NOTE

   When specifying the Kickstart source, use **nfs:** as the protocol, the server's host name or IP address, the colon sign (**:**), and the path inside the directory holding the file. For example, if the server's host name is **myserver.example.com** and you have saved the file in **/nfs/rhel10-kickstart/my-ks.cfg**, specify **inst.ks=nfs:myserver.example.com:/nfs/rhel10-kickstart/my-ks.cfg** as the installation source boot option.

Additional resources

- Preparing a remote installation by using RDP

11.2. SHARING THE INSTALLATION FILES ON AN HTTP OR HTTPS SERVER

You can store Kickstart files on an HTTP or HTTPS server to enable automated installations across multiple systems. It eliminates the need for physical media and provides centralized management of installation configurations for efficient system deployment.

Prerequisites

- You have administrator-level access to a server with Red Hat Enterprise Linux 10 on the local network.
- The system to be installed can connect to the server.

Procedure

1. To store the Kickstart file on an HTTP, install the **httpd** package:

```
# dnf install httpd
```

2. Optional: To store the Kickstart file on an HTTPS, install **httpd** and **mod_ssl** packages:

```
# dnf install httpd mod_ssl
```

IMPORTANT

If you use an HTTPS server with a self-signed certificate, you must boot the installation program with the **inst.noverifyssl** option.

3. Copy the Kickstart file to the HTTP(S) server into a subdirectory of the **/var/www/html/** directory.

4. Enable the http service in firewalld.

```
# firewall-cmd --permanent --add-service=http
# firewall-cmd --reload
```

5. Optional: Enable the **https** service in firewalld:

```
# firewall-cmd --permanent --add-service=https
# firewall-cmd --reload
```

6. Start the **httpd** service:

```
# systemctl enable --now httpd.service
```

The Kickstart file is now accessible and ready to be used for installation.

When specifying the location of the Kickstart file, use **http://** or **https://** as the protocol, the server's host name or IP address, and the path of the Kickstart file, relative to the HTTP server root. For example, if you are using HTTP, the server's host name is **myserver.example.com**, and you have copied the Kickstart file as **/var/www/html/rhel10-install/my-ks.cfg**, specify **http://myserver.example.com/rhel10-install/my-ks.cfg** as the file location.

Additional resources

- Deploying Web Servers and Proxies
- Configuring and using Database Servers

11.3. SHARING THE INSTALLATION FILES ON AN FTP SERVER

You can store Kickstart files on an FTP server to enable automated installations across multiple systems. It eliminates the need for physical media and provides centralized management of installation configurations for efficient system deployment by using the FTP protocol.

Prerequisites

- You have administrator-level access to a server with Red Hat Enterprise Linux 10 on the local network.
- The system to be installed can connect to the server.

Procedure

1. Install the **vsftpd** package by running the following command as root:

```
# dnf install vsftpd
```

2. Open and edit the **/etc/vsftpd/vsftpd.conf** configuration file in a text editor.

   a. Change the line **anonymous_enable=NO** to **anonymous_enable=YES**

   b. Change the line **write_enable=YES** to **write_enable=NO**.

   c. Add lines **pasv_min_port=*min_port*** and **pasv_max_port=*max_port***. Replace ***min_port*** and ***max_port*** with the port number range used by the FTP server in passive mode, for example, **10021** and **10031**.
   This step can be necessary in network environments featuring various firewall/NAT setups.

   d. Optional: Add custom changes to your configuration. For available options, see the **vsftpd.conf(5)** man page. This procedure assumes that default options are used.

3. Configure the firewall to allow the FTP port and port range from the previous step:

```
# firewall-cmd --add-port min_port-max_port/tcp --permanent
```

   Replace *min_port-max_port* with the port numbers you entered into the **/etc/vsftpd/vsftpd.conf** configuration file.

4. Configure the firewall to allow FTP service.

```
# firewall-cmd --add-service ftp --permanent
```

5. Reload the firewall to apply the new rules.

```
# firewall-cmd --reload
```

6. Copy the Kickstart file to the FTP server into the **/var/ftp/** directory or its subdirectory.

7. Ensure that the correct SELinux context and access mode is set on the file:

```
# restorecon -r /var/ftp/your-kickstart-file.ks
# chmod 444 /var/ftp/your-kickstart-file.ks
```

8. Start the **vsftpd** service:

```
# systemctl enable --now vsftpd.service
```

   If the service was running before you changed the **/etc/vsftpd/vsftpd.conf** file, restart the service to load the edited file:

```
# systemctl restart vsftpd.service
```

   The Kickstart file is now accessible and ready to be used for installations by systems on the same network.

NOTE

When configuring the installation source, use **ftp:**// as the protocol, the server’s host name or IP address, and the path of the Kickstart file, relative to the FTP server root. For example, if the server’s host name is **myserver.example.com** and you have copied the file to /**var**/**ftp**/**my-ks.cfg**, specify **ftp:**//**myserver.example.com**/**my-ks.cfg** as the installation source.

CHAPTER 12. SEMI-AUTOMATED INSTALLATIONS: MAKING KICKSTART FILES AVAILABLE TO THE RHEL INSTALLER

After your Kickstart file is ready, you can make it available for installation on the destination system.

12.1. SHARING THE INSTALLATION FILES ON A LOCAL VOLUME

You can store Kickstart files on a local storage volume to enable automated installations without network dependencies. It is ideal for isolated environments or systems where network access is limited, providing a portable solution for unattended RHEL deployments.

Prerequisites

- You have a drive that can be moved to the machine to be installed, such as a USB stick.
- The drive contains a partition that can be read by the installation program. The supported types are **ext2**, **ext3**, **ext4**, **xfs**, and **fat**.
- The drive is connected to the system and its volumes are mounted.

Procedure

1. List volume information and note the UUID of the volume to which you want to copy the Kickstart file.

   ```
   # lsblk -l -p -o name,rm,ro,hotplug,size,type,mountpoint,uuid
   ```

2. Navigate to the file system on the volume.
3. Copy the Kickstart file to this file system.
4. Make a note of the string to use later with the **inst.ks=** option. This string is in the form **hd:UUID=*volume-UUID*:*path/to/kickstart-file.cfg***. Note that the path is relative to the file system root, not to the / root of file system hierarchy. Replace *volume-UUID* with the UUID you noted earlier from the output of **lsblk**.
5. Unmount all drive volumes:

   ```
   # umount /dev/xyz ...
   ```

   (add all the volumes to the command, separated by spaces.)

12.2. SHARING THE INSTALLATION FILES ON A LOCAL VOLUME FOR AUTOMATIC LOADING

You can configure a local storage volume with a specially named and labeled Kickstart file for automatic detection during installation. It enables completely automated installations without manual intervention or network dependencies, making it ideal for unattended deployments.

Prerequisites

- You have a drive that can be moved to the machine to be installed, such as a USB stick.

- The drive contains a partition that can be read by the installation program. The supported types are **ext2**, **ext3**, **ext4**, **xfs**, and **fat**.
- The drive is connected to the system and its volumes are mounted.

Procedure

1. List volume information to which you want to copy the Kickstart file.

   ```
   # lsblk -l -p
   ```

2. Navigate to the file system on the volume.
3. Copy the Kickstart file into the root of this file system.
4. Rename the Kickstart file to **ks.cfg**.
5. Rename the volume as **OEMDRV**:
   - For **ext2**, **ext3**, and **ext4** file systems:

     ```
     # e2label /dev/xyz OEMDRV
     ```

   - For the XFS file system:

     ```
     # xfs_admin -L OEMDRV /dev/xyz
     ```

   Replace */dev/xyz* with the path to the volume's block device.
6. Unmount all drive volumes:

   ```
   # umount /dev/xyz ...
   ```

   (add all the volumes to the command, separated by spaces.)

CHAPTER 13. STARTING KICKSTART INSTALLATIONS

You can start Kickstart installations in multiple ways:

- Automatically by editing the boot options in network boot configuration.
- Automatically by providing the file on a volume with a specific name.
- Manually by providing inst.ks= kernel command line option in bootloader menu.

You can register RHEL using the Red Hat Content Delivery Network (CDN). CDN is a geographically distributed series of web servers. These servers provide, for example, packages and updates to RHEL hosts with a valid subscription.

During the installation, registering and installing RHEL from the CDN offers following benefits:

- Utilizing the latest packages for an up-to-date system immediately after installation and
- Integrated support for connecting to Red Hat Lightspeed and enabling System Purpose.

13.1. STARTING A KICKSTART INSTALLATION AUTOMATICALLY BY USING PXE OR UEFI HTTP BOOT

AMD64, Intel 64, and 64-bit ARM systems and IBM Power Systems servers have the ability to boot by using a PXE server. When you configure the PXE server or UEFI HTTP boot, add the boot option into the boot loader configuration file. It in turn starts the installation automatically. By using this approach, it is possible to automate the installation completely, including the boot process.

This procedure is intended as a general reference; detailed steps differ based on your system's architecture, and not all options are available on all architectures (for example, you cannot use PXE boot on 64-bit IBM Z).

Prerequisites

- You have a Kickstart file ready in a location accessible from the system to be installed.
- You have a PXE server that can be used to boot the system and begin the installation and/or an HTTP server configured for UEFI HTTP boot.

Procedure

1. Open the boot loader configuration file on your PXE server, and add the **inst.ks=** boot option to the appropriate line. The name of the file and its syntax depends on your system's architecture and hardware:

   a. On AMD64 and Intel 64 systems with BIOS, the file name can be either default or based on your system's IP address. In this case, add the **inst.ks=** option to the append line in the installation entry. A sample append line in the configuration file looks similar to the following:

   ```
   append initrd=initrd.img inst.ks=http://192.168.124.2/kickstarts/ks.cfg
   ```

   b. On systems using the GRUB2 boot loader (AMD64, Intel 64, and 64-bit ARM systems with UEFI firmware and IBM Power Systems servers), the file name will be **grub.cfg**. In this file, append the **inst.ks=** option to the kernel line in the installation entry. A sample kernel line in the configuration file will look similar to the following:

```
linuxefi /redhat/images/pxeboot/vmlinuz inst.ks=http://192.168.124.2/kickstarts/ks.cfg
```

2. If you want to configure an automated Kickstart installation by using UEFI HTTP boot, locate the bootloader configuration file (grub.cfg) served by the HTTP server and edit it in a similar way as in step 1.b:

```
linuxefi /redhat/images/pxeboot/vmlinuz inst.ks=http://192.168.124.2/kickstarts/ks.cfg
```

3. Boot the installation from the network server.
   The installation begins now by using the installation options specified in the Kickstart file. If the Kickstart file is valid and contains all required commands, the installation is completely automated.

   NOTE

   If you have installed a Red Hat Enterprise Linux Beta release, on systems having UEFI Secure Boot enabled, then add the Beta public key to the system's Machine Owner Key (MOK) list.

Additional resources

- Preparing a PXE installation source
- Configuring the HTTP server for HTTP boot
- Sharing the installation files on an HTTP or HTTPS server

13.2. STARTING A KICKSTART INSTALLATION AUTOMATICALLY BY USING A LOCAL VOLUME

You can configure automated Kickstart installations using a local storage volume with a properly labeled and named Kickstart file. It enables unattended installations without network dependencies, making it suitable for isolated environments or systems without network access.

Prerequisites

- You have a volume prepared with label **OEMDRV** and the Kickstart file present in its root as **ks.cfg**.
- A drive containing this volume is available on the system as the installation program boots.

Procedure

1. Boot the system by using a local media (a CD, DVD, or a USB flash drive).
2. At the boot prompt, specify the required boot options.
   a. If a required repository is in a network location, you may need to configure the network by using the **ip=** option. The installer tries to configure all network devices by using the DHCP protocol by default without this option.

   b. In order to access a software source from which necessary packages will be installed, you may need to add the **inst.repo=** option. If you do not specify this option, you must specify the installation source in the Kickstart file.
      For more information about installation sources, see Kickstart commands for installation program configuration and flow control.

3. Start the installation by confirming your added boot options.
   The installation begins now, and the Kickstart file is automatically detected and used to start an automated Kickstart installation.

> **NOTE**
>
> If you have installed a Red Hat Enterprise Linux Beta release, on systems having UEFI Secure Boot enabled, then add the Beta public key to the system's Machine Owner Key (MOK) list. For more information about UEFI Secure Boot and Red Hat Enterprise Linux Beta releases, see the UEFI Secure Boot and Beta release requirements.

13.3. BOOTING THE INSTALLATION ON IBM Z TO INSTALL RHEL IN AN LPAR

You can boot the installation on IBM Z to install RHEL in an LPAR.

13.3.1. Booting the RHEL installation from an SFTP, FTPS, or FTP server to install in an IBM Z LPAR

You can install RHEL into an LPAR by using an SFTP, FTPS, or FTP server.

Procedure

1. Log in on the IBM Z Hardware Management Console (HMC) or the Support Element (SE) as a user with sufficient privileges to install a new operating system to an LPAR.

2. On the **Systems** tab, select the mainframe you want to work with, then on the **Partitions** tab select the LPAR to which you wish to install.

3. At the bottom of the screen, under **Daily**, find **Operating System Messages**. Double-click **Operating System Messages** to show the text console on which Linux boot messages will appear.

4. Double-click **Load from Removable Media or Server**.

5. In the dialog box that follows, select **SFTP/FTPS/FTP Server**, and enter the following information:

   - **Host Computer** - Host name or IP address of the FTP server you want to install from, for example **ftp.redhat.com**

   - **User ID** - Your user name on the FTP server. Or, specify anonymous.

   - **Password** - Your password. Use your email address if you are logging in as anonymous.

   - **File location (optional)** - Directory on the FTP server holding the Red Hat Enterprise Linux for IBM Z, for example **/rhel/s390x/**.

6. Click **Continue**.
7. In the dialog that follows, keep the default selection of **generic.ins** and click **Continue**.

13.3.2. Booting the RHEL installation from a prepared DASD to install in an IBM Z LPAR

You can install Red Hat Enterprise Linux into an LPAR by using an already prepared DASD.

Procedure

1. Log in on the IBM Z Hardware Management Console (HMC) or the Support Element (SE) as a user with sufficient privileges to install a new operating system to an LPAR.
2. On the **Systems** tab, select the mainframe you want to work with, then on the **Partitions** tab select the LPAR to which you wish to install.
3. At the bottom of the screen, under **Daily**, find **Operating System Messages**. Double-click **Operating System Messages** to show the text console on which Linux boot messages will appear.
4. Double-click **Load**.
5. In the dialog box that follows, select **Normal** as the **Load type**.
6. As Load **address**, fill in the device number of the DASD.
7. Click the **OK** button.

13.3.3. Booting the RHEL installation from an FCP-attached SCSI disk to install in an IBM Z LPAR

You can install Red Hat Enterprise Linux into an LPAR by using an already prepared FCP attached SCSI disk.

Procedure

1. Log in on the IBM Z Hardware Management Console (HMC) or the Support Element (SE) as a user with sufficient privileges to install a new operating system to an LPAR.
2. On the **Systems** tab, select the mainframe you want to work with, then on the **Partitions** tab select the LPAR to which you wish to install.
3. At the bottom of the screen, under **Daily**, find **Operating System Messages**. Double-click **Operating System Messages** to show the text console on which Linux boot messages will appear.
4. Double-click **Load**.
5. In the dialog box that follows, select **SCSI** as the **Load type**.
6. As **Load address**, fill in the device number of the FCP channel connected with the SCSI disk.
7. As **World wide port name**, fill in the WWPN of the storage system containing the disk as a 16-digit hexadecimal number.

8. As **Logical unit number**, fill in the LUN of the disk as a 16-digit hexadecimal number.

9. Leave the **Boot record logical block address** as **0** and the **Operating system specific load parameters** empty.

10. Click the **OK** button.

13.4. BOOTING THE INSTALLATION ON IBM Z TO INSTALL RHEL IN Z/VM

When installing under z/VM, you can boot from:

- The z/VM virtual reader
- A DASD or an FCP-attached SCSI disk prepared with the **zipl** boot loader

13.4.1. Booting the RHEL installation by using the z/VM Reader

Booting the Red Hat Enterprise Linux (RHEL) installation by using the z/VM Reader is a method designed for IBM Z (s390x) mainframe environments. This approach delivers installation boot images directly to a virtual machine through the z/VM Reader. It enables installations even when other boot methods (such as DASD or FCP) are unavailable or impractical.

Procedure

1. If necessary, add the device containing the z/VM TCP/IP tools to your CMS disk list. For example:

   ```
   cp link tcpmaint 592 592
   acc 592 fm
   ```

   Replace *fm* with any **FILEMODE** letter.

2. For a connection to an FTPS server, enter:

   ```
   ftp <host> (secure
   ```

   Where **host** is the host name or IP address of the FTP server that hosts the boot images (**kernel.img** and **initrd.img**).

3. Log in and execute the following commands. Use the **(repl** option if you are overwriting existing **kernel.img**, **initrd.img**, **generic.prm**, or **redhat.exec** files:

   ```
   cd /location/of/install-tree/images/
   ascii
   get generic.prm (repl
   get redhat.exec (repl
   locsite fix 80
   binary
   get kernel.img (repl
   get initrd.img (repl
   quit
   ```

4. Optional: Check whether the files were transferred correctly by using the CMS command **filelist**

to show the received files and their format. It is important that **kernel.img** and **initrd.img** have a fixed record length format denoted by F in the Format column and a record length of 80 in the Lrecl column. For example:

```
VMUSER FILELIST A0 V 169 Trunc=169 Size=6 Line=1 Col=1 Alt=0
Cmd Filename Filetype Fm Format Lrecl Records Blocks Date Time
REDHAT EXEC  B1 V 22 1  1 4/15/10 9:30:40
GENERIC PRM  B1 V 44 1 1 4/15/10 9:30:32
INITRD IMG  B1 F 80 118545 2316 4/15/10 9:30:25
KERNEL IMG  B1 F 80 74541 912 4/15/10 9:30:17
```

Press **PF3** to quit filelist and return to the CMS prompt.

5. Customize boot parameters in **generic.prm** as necessary. For details, see Customizing boot parameters.
   Another way to configure storage and network devices is by using a CMS configuration file. In such a case, add the ***CMSDASD=*** and ***CMSCONFFILE=*** parameters to **generic.prm**.

6. Finally, execute the REXX script redhat.exec to boot the installation program:

   ```
   redhat
   ```

13.4.2. Booting the RHEL installation by using a prepared DASD

Booting the Red Hat Enterprise Linux (RHEL) installation from a prepared Direct Access Storage Device (DASD) is a method commonly used on IBM Z (s390x) mainframe systems. Using a prepared DASD ensures that the installation process is consistent, repeatable, and can be integrated into larger provisioning workflows.

Procedure

- Boot from the prepared DASD and select the **zipl** boot menu entry referring to the Red Hat Enterprise Linux installation program. Use a command of the following form:

  ```
  cp ipl DASD_device_number loadparm boot_entry_number
  ```

  Replace *DASD_device_number* with the device number of the boot device, and *boot_entry_number* with the **zipl** configuration menu for this device. For example:

  ```
  cp ipl eb1c loadparm 0
  ```

13.4.3. Booting the RHEL installation by using a prepared FCP-attached SCSI disk

Booting Red Hat Enterprise Linux (RHEL) from a prepared FCP-attached SCSI disk is a common method on IBM Z (s390x) mainframe systems, especially when using Fibre Channel Protocol (FCP) storage. This approach is used when you need to install RHEL on systems that rely on external SAN storage rather than local DASD devices.

Procedure

1. Configure the SCSI boot loader of z/VM to access the prepared SCSI disk in the FCP Storage Area Network. Select the prepared **zipl** boot menu entry referring to the Red Hat Enterprise Linux installation program. Use a command of the following form:

```
cp set loaddev portname WWPN lun LUN bootprog boot_entry_number
```

Replace *WWPN* with the World Wide Port Name of the storage system and *LUN* with the Logical Unit Number of the disk. The 16-digit hexadecimal numbers must be split into two pairs of eight digits each. For example:

```
cp set loaddev portname 50050763 050b073d lun 40204011 00000000 bootprog 0
```

2. Optional: Confirm your settings with the command:

```
query loaddev
```

3. Boot the FCP device connected with the storage system containing the disk with the following command:

```
cp ipl FCP_device
```

For example:

```
cp ipl fc00
```

CHAPTER 14. CONSOLES AND LOGGING DURING INSTALLATION

The RHEL installer uses the **tmux** terminal multiplexer to display and control several windows in addition to the main interface. Each of these windows serve a different purpose; they display several different logs, which can be used to troubleshoot issues during the installation process. One of the windows provides an interactive shell with **root** privileges, unless this was specifically disabled by using a boot option or a Kickstart command.

The terminal multiplexer is running in virtual console 1. To switch from the actual installation environment to **tmux**, press **Ctrl**+**Alt**+**F1**. To go back to the main installation interface which runs in virtual console 6, press **Ctrl**+**Alt**+**F6**. During the text mode, installation starts in virtual console 1 (**tmux**), and switching to console 6 will open a shell prompt instead of a graphical interface.

The console running **tmux** has five available windows; their contents are described in the following table, along with keyboard shortcuts. Note that the keyboard shortcuts are two-part: first press **Ctrl**+**b**, then release both keys, and press the number key for the window you want to use.

You can also use **Ctrl**+**b n**, **Alt+ Tab**, and **Ctrl**+**b p** to switch to the next or previous **tmux** window, respectively.

Table 14.1. Available tmux windows

Shortcut	Contents
Ctrl+**b 1**	Main installation program window. Contains text-based prompts (used for text mode and also for interactive entry of RDP credentials), and also some debugging information.
Ctrl+**b 2**	Interactive shell with **root** privileges.
Ctrl+**b 3**	Installation log; displays messages stored in **/tmp/anaconda.log**.
Ctrl+**b 4**	Storage log; displays messages related to storage devices and configuration, stored in **/tmp/storage.log**.
Ctrl+**b 5**	Program log; displays messages from utilities executed during the installation process, stored in **/tmp/program.log**.
Ctrl+**b 6**	Packaging log; displays messages related to packages, stored in **/tmp/packaging.log**.

CHAPTER 15. INSTALLING KERNEL-64K ON ARM BY USING THE COMMAND LINE

By default, RHEL 10 is distributed with a kernel supporting a 4k page size. The 4k kernel is sufficient for efficient memory usage in smaller environments or small cloud instances. In these cases, the usage of a 64k page kernel is not practical due to space, power, and cost constraints.

If you have already installed RHEL with the default kernel (supporting 4k page size), you can install **kernel-64k** post installation by using the command line.

IMPORTANT

It is not recommended to move between 4k and 64k page size kernels after the initial boot without reinstallation of the OS.

Procedure

1. Open the terminal as the root user, and enter:

   ```
   # dnf -y install kernel-64k
   ```

2. To set the **kernel-64k** as default, enter:

   ```
   # k=$(echo /boot/vmlinuz*64k)
   # grubby --set-default=$k \
             --update-kernel=$k \
             --args="crashkernel=2G-:640M"
   ```

3. Set the system boot order to use RHEL as the default option.

   a. Obtain the current boot order. For example:

      ```
      # efibootmgr
      BootCurrent: 0000
      Timeout: 5 seconds
      BootOrder: 0003,0004,0001,0000,0002,0005
      Boot0000\* Red Hat Enterprise Linux
      ```

   b. Set the boot order to prioritize RHEL. For example, for the output in the previous step, use the following command:

      ```
      # efibootmgr -o 0000,0001,0002,0003,0004,0005
      ```

4. Reboot the system:

   ```
   # reboot
   ```

5. Optional: After reboot, remove the 4k kernel:

   ```
   # dnf remove kernel-core
   ```

Keeping both versions accidentally can make the 4k kernel default when you update the kernel in future using the **yum update** command.

Verification

- To verify the page size, open the terminal and run the following command as any user:

  ```
  $ getconf PAGESIZE
  65536
  ```

  The output **65536** indicates that the 64k kernel is in use.

- To verify swap is enabled, enter:

  ```
  $ free
               total        used        free      shared  buff/cache   available
  Mem:       35756352     3677184    34774848       25792      237120    32079168
  Swap:       6504384           0     6504384
  ```

  The total and free columns are non-zero, which indicates the swap is enabled successfully.

CHAPTER 16. REGISTERING YOUR RHEL SYSTEM

After installing RHEL, register your system to receive updates and to access Red Hat services. You can register your system by using various methods:

- RHC client
- Graphical user interface
- Subscription manager
- Registration assistant

The RHC client provides the following flexible features to register your RHEL system and manage the extent of data collection:

- Access to Red Hat content: Provides access to Red Hat CDN repositories.
- Analytics for Red Hat Lightspeed: Enables data collection for Red Hat Lightspeed.
- Remote management: Establishes an additional MQTT network connection to Red Hat services for remote execution of certain actions from console.redhat.com.

By default, the RHC client enables all available features. However, you can override this behavior by enabling or disabling specific features based on your connectivity requirements. Review the available options and select the appropriate features based on your internal policies on data and network connectivity.

NOTE

Use the RHC client only for systems directly connecting to Hybrid Cloud Console, the RHC client currently does not support Satellite. For more information on registering a system to a Red Hat Satellite or Capsule, see Red Hat Satellite Documentation .

16.1. REGISTERING A SYSTEM BY USING THE RHC CLIENT

Register your RHEL system by using the RHC client to connect to Red Hat services with the default feature levels. This enables all available features: access to Red Hat content, analytics for Red Hat Lightspeed, and remote management.

Prerequisites

- You have an activation key and an organization ID to register your system.

Procedure

1. Open the terminal window.
2. Do one of the following:
   a. To register the system with the default feature level and ensure system executes the **Remediation** and **Tasks** from Red Hat Lightspeed:

```
# rhc connect --activation-key=<activation_key> --organization=<organization_ID>
# dnf install -y rhc-worker-playbook
```

b. To disable any of the features while registering your system:

```
# rhc connect --activation-key=<activation_key> --organization=<organization_ID>
--disable-feature <feature>
```

Where *feature* can be replaced with:

- content - Provides access to Red Hat CDN repositories.
- analytics - Enables data collection for Red Hat Lightspeed.
- remote-management - Establishes an additional MQTT network connection to Red Hat services for remote execution of certain actions from console.redhat.com.
  For example, if you want to register your system with a remote management feature disabled so that your system cannot be managed remotely but the system has access to RHEL content and collects data for Red Hat Lightspeed analytics, enter:

  ```
  # rhc connect --activation-key=<activation_key> --
  organization=<organization_ID> --disable-feature remote-management
  ```

Verification

- Confirm the active features when registering with default options:

  ```
  # rhc status
  Connection status:
  ✓ Connected to Red Hat Subscription Management
  ✓ Connected to Red Hat Lightspeed
  ✓ The yggdrasil service is active
  Manage your connected systems: https://red.ht/connector
  ```

- Confirm the active features with remote-management disabled:

  ```
  # rhc status
  Connection status:
  ✓ Connected to Red Hat Subscription Management
  ✓ Connected to Red Hat Lightspeed
  ● The yggdrasil service is inactive
  ```

Additional resources

- Remote Host Configuration and Management

16.2. REGISTERING YOUR SYSTEM USING THE SUBSCRIPTION MANAGER

You can register your Red Hat Enterprise Linux subscription by using the command line to get software updates and access Red Hat services.

TIP

For an improved and simplified experience registering your hosts to Red Hat, use the RHC client. The RHC client registers your system to Red Hat making your system ready for Red Hat Lightspeed data collection and enabling direct issue remediation from Red Hat Lightspeed for Red Hat Enterprise Linux. For more information, see information, see RHC registration.

Prerequisites

- You have an active, non-evaluation Red Hat Enterprise Linux subscription.
- Your Red Hat subscription status is verified.
- You have successfully installed Red Hat Enterprise Linux and logged into the system as root.

Procedure

1. Open a terminal window as a root user.
2. Register your Red Hat Enterprise Linux system by using the activation key:

   ```
   # subscription-manager register --activationkey=<activation_key_name> --
   org=<organization_ID>
   ```

   When the system is successfully registered, an output similar to the following is displayed:

   ```
   The system has been registered with ID:
   62edc0f8-855b-4184-b1b8-72a9dc793b96
   ```

Additional resources

- Using an activation key to register a system with Red Hat Subscription Manager
- Getting Started with RHEL System Registration

16.3. REGISTERING RHEL 10 USING THE INSTALLER GUI

You can register a Red Hat Enterprise Linux by using the RHEL installer GUI to get software updates and access Red Hat services.

Prerequisites

- You have a valid user account on the Red Hat Customer Portal. See the Create a Red Hat Login page.
- You have a valid Activation Key and Organization ID.

Procedure

1. From the **Installation Summary** screen, under **Software**, click **Connect to Red Hat**
2. Authenticate your Red Hat account using the **Account** or **Activation Key** option.

3. Optional: In the **Set System Purpose** field select the **Role**, **SLA**, and **Usage** attribute that you want to set from the drop-down menu.
   At this point, your Red Hat Enterprise Linux system has been successfully registered.

16.4. REGISTRATION ASSISTANT

Registration Assistant is designed to help you choose the most suitable registration option for your Red Hat Enterprise Linux environment:

- For assistance with using a username and password to register RHEL with the Subscription Manager client, see the RHEL registration assistant on the Customer Portal.
- For assistance with registering your RHEL system to Red Hat Lightspeed, see the Red Hat Lightspeed registration assistant on the Hybrid Cloud Console.

16.5. ADDITIONAL RESOURCES

- Using an activation key to register a system with Red Hat Subscription Manager
- Getting Started with RHEL System Registration

CHAPTER 17. CONFIGURING SYSTEM PURPOSE USING THE SUBSCRIPTION-MANAGER COMMAND-LINE TOOL

System purpose is a feature of the Red Hat Enterprise Linux installation to help RHEL customers get the benefit of our subscription experience and services offered in the Red Hat Hybrid Cloud Console, a dashboard-based, Software-as-a-Service (SaaS) application that enables you to view subscription usage in your Red Hat account.

You can configure system purpose attributes either on the activation keys or by using the subscription manager tool.

Prerequisites

- You have installed and registered your Red Hat Enterprise Linux 10 system, but system purpose is not configured.
- You are logged in as a **root** user.

Procedure

1. From a terminal window, run the following command to set the intended role of the system:

   ```
   # subscription-manager syspurpose role --set VALUE
   ```

   Replace **VALUE** with the role that you want to assign:

   - **Red Hat Enterprise Linux Server**
   - **Red Hat Enterprise Linux Workstation**
   - **Red Hat Enterprise Linux Compute Node**

   For example:

   ```
   # subscription-manager syspurpose role --set Red Hat Enterprise Linux Server
   ```

   a. Optional: Before setting a value, see the available roles supported by the subscriptions for your organization:

      ```
      # subscription-manager syspurpose role --list
      ```

   b. Optional: Run the following command to unset the role:

      ```
      # subscription-manager syspurpose role --unset
      ```

2. Run the following command to set the intended Service Level Agreement (SLA) of the system:

   ```
   # subscription-manager syspurpose service-level --set VALUE
   ```

   Replace **VALUE** with the SLA that you want to assign:

   - **Premium**

- **Standard**
- **Self-Support**

For example:

```
# subscription-manager syspurpose service-level --set "Standard"
```

a. Optional: Before setting a value, see the available service-levels supported by the subscriptions for your organization:

```
# subscription-manager syspurpose service-level --list
```

b. Optional: Run the following command to unset the SLA:

```
# subscription-manager syspurpose service-level --unset
```

3. Run the following command to set the intended usage of the system:

```
# subscription-manager syspurpose usage --set "VALUE"
```

Replace **VALUE** with the usage that you want to assign:

- **Production**
- **Disaster Recovery**
- **Development/Test**

For example:

```
# subscription-manager syspurpose usage --set "Production"
```

a. Optional: Before setting a value, see the available usages supported by the subscriptions for your organization:

```
# subscription-manager syspurpose usage --list
```

b. Optional: Run the following command to unset the usage:

```
# subscription-manager syspurpose usage --unset
```

4. Run the following command to show the current system purpose properties:

```
# subscription-manager syspurpose --show
```

a. Optional: For more detailed syntax information run the following command to access the **subscription-manager** man page and browse to the SYSPURPOSE OPTIONS:

```
# man subscription-manager
```

Verification

- To verify the system’s subscription status:

  ```
  # subscription-manager status
  +-------------------------------------------+
     System Status Details
  +-------------------------------------------+
  Overall Status: Registered
  ```

  For more information on the subscriptions services, see the Getting Started with the Subscriptions Service guide.

CHAPTER 18. POST-INSTALLATION SECURITY HARDENING

RHEL is designed with robust security features enabled by default. However, you can enhance its security further through additional hardening measures.

For more information about:

- Processes and practices for securing RHEL servers and workstations against local and remote intrusion, exploitation, and malicious activity, see Security hardening.
- Control how users and processes interact with the files on the system or control which users can perform which actions by mapping them to specific SELinux confined users, see Using SELinux.
- Tools and techniques to improve the security of your networks and lower the risks of data breaches and intrusions, see Securing networks.
- Packet filters, such as firewalls, that use rules to control incoming, outgoing, and forwarded network traffic, see Configuring firewalls and packet filters.

CHAPTER 19. CHANGING A SUBSCRIPTION SERVICE

To manage the subscriptions, you can register a RHEL system with either Red Hat Subscription Management Server or Red Hat Satellite Server. If required, you can change the subscription service at a later point. To change the subscription service under which you are registered, unregister the system from the current service and then register it with a new service.

To receive the system updates, register your system with either of the management servers.

This section contains information about how to unregister your RHEL system from the Red Hat Subscription Management Server and Red Hat Satellite Server.

19.1. PREREQUISITES

You have registered your system with any one of the following:

- Red Hat Subscription Management Server
- Red Hat Satellite Server version 6.17

To receive the system updates, register your system with either of the management servers.

19.2. UNREGISTERING FROM SUBSCRIPTION MANAGEMENT SERVER

You can unregister a RHEL system from Red Hat Subscription Management Server by using a command line and the Subscription Manager user interface.

19.2.1. Unregistering using command line

Use the **unregister** command to unregister a RHEL system from Red Hat Subscription Management Server.

Procedure

- Run the unregister command as a root user or **sudo**, without any additional parameters.

  ```
  # subscription-manager unregister
  ```

  The system is unregistered from the Subscription Management Server, and the status 'System Not Registered' is displayed with the **Register System..** button enabled.

  To continue uninterrupted services, re-register the system with either of the management services. If you do not register the system with a management service, you may fail to receive the system updates.

Additional resources

- Using and Configuring Red Hat Subscription Manager
- Registering your system using the command line

19.2.2. Unregistering by using Subscription Manager user interface

You can unregister a RHEL system from Red Hat Subscription Management Server by using the Subscription Manager user interface.

Procedure

1. Log in to your system.
2. From the top left-hand side of the window, click **Activities**.
3. From the menu options, click the **Show Applications** icon.
4. Click the **Settings** icon, or enter **Settings Manager** in the search.
5. In the left side column, select the **System** options, then on the new screen select **Registration**. The Registration screen displays the current status of your Subscription.
6. Click the **Remove Registration...** button and confirm your choice.
7. Enter your administrator password in the **Authentication Required** dialog box. The system is unregistered from the **Subscription Management Server**, and the status **System Not Registered** is displayed with the **Register System...** button enabled.
   To continue uninterrupted services, re-register the system with either of the management services. If you do not register the system with a management service, you may fail to receive the system updates.

Additional resources

- Registering your system using the Subscription Manager User Interface
- Using and Configuring Red Hat Subscription Manager

19.3. UNREGISTERING FROM SATELLITE SERVER

Removing a Red Hat Enterprise Linux (RHEL) system from Satellite Server updates the system's status on the server side. However, the system's local registration remains unchanged. To ensure a complete transition to Red Hat Subscription Management (RHSM), manually unregister the system and restore the default configuration in the **/etc/rhsm/rhsm.conf** file.

Procedure

1. Remove the host from the Satellite server .
2. Unregister the system:

   ```
   # subscription-manager unregister
   ```

3. Revert the **/etc/rhsm/rhsm.conf** configuration file to its default values:

   ```
   # cd /etc/rhsm/
   # mv rhsm.conf.bak rhsm.conf
   ```

NOTE

You can restore this file only if the original content of the file was backed up earlier. Otherwise, you must manually restore the file from a non-modified RHEL machine.

4. Verify that the **/etc/rhsm/rhsm.conf** file has been reverted to default values:

```
# subscription-manager config --list
[server]
   hostname = [subscription.rhsm.redhat.com]
   insecure = [0]
   no_proxy = []
   port = [443]
   prefix = [/subscription]
   proxy_hostname = []
   proxy_password = []
   proxy_port = []
   proxy_scheme = [http]
   proxy_user = []
   server_timeout = [180]
   ssl_verify_depth = [3]

[rhsm]
   auto_enable_yum_plugins = [1]
   baseurl = [https://cdn.redhat.com]
   ca_cert_dir = [/etc/rhsm/ca/]
   consumercertdir = [/etc/pki/consumer]
   entitlementcertdir = [/etc/pki/entitlement]
   full_refresh_on_yum = [0]
   inotify = [1]
   manage_repos = [1]
   package_profile_on_trans = [0]
   pluginconfdir = [/etc/rhsm/pluginconf.d]
   plugindir = [/usr/share/rhsm-plugins]
   productcertdir = [/etc/pki/product]
   repo_ca_cert = /etc/rhsm/ca/redhat-uep.pem
   repomd_gpg_url = []
   report_package_profile = [1]

[rhsmcertd]
   auto_registration = [0]
   auto_registration_interval = [60]
   autoattachinterval = [1440]
   certcheckinterval = [240]
   disable = [0]
   splay = [1]

[logging]
   default_log_level = [INFO]

[] - Default value in use
```

5. Register the system:

```
# subscription-manager register
Username: <_redhat_portal_admin_username_>
Password:
```

6. Enable the required repositories:

```
# subscription-manager repos --disable "*"
# subscription-manager repos
# subscription-manager repos --enable=<repo-id>
```

CHAPTER 20. CONFIGURING A LINUX INSTANCE ON 64-BIT IBM Z

Perform following common tasks for configuring Red Hat Enterprise Linux on 64-bit IBM Z.

20.1. ADDING DASDS TO A Z/VM SYSTEM

Direct Access Storage Devices (DASDs) are a type of storage commonly used with 64-bit IBM Z. For more information, see Working with DASDs in the IBM Knowledge Center. The following example is how to set a DASD online, format it, and make the change persistent.

Verify that the device is attached or linked to the Linux system if running under z/VM.

```
CP ATTACH EB1C TO *
```

To link a mini disk to which you have access, run the following commands:

```
CP LINK RHEL7X 4B2E 4B2E MR
DASD 4B2E LINKED R/W
```

20.2. DYNAMICALLY SETTING DASDS ONLINE

You can set DASDs online dynamically to add or activate storage devices without rebooting.

Procedure

1. Use the **cio_ignore** utility to remove the DASD from the list of ignored devices and make it visible to Linux:

   ```
   # cio_ignore -r device_number
   ```

   Replace *device_number* with the device number of the DASD. For example:

   ```
   # cio_ignore -r 4b2e
   ```

2. Set the device online. Use a command of the following form:

   ```
   # chccwdev -e device_number
   ```

   Replace *device_number* with the device number of the DASD. For example:

   ```
   # chccwdev -e 4b2e
   ```

   For instructions on how to set a DASD online persistently, see Persistently setting DASDs online.

20.3. PREPARING A NEW DASD WITH LOW-LEVEL FORMATTING

Once the disk is online, change back to the **/root** directory and low-level format the device. This is only required once for a DASD during its entire lifetime:

```
# cd /root
# dasdfmt -b 4096 -d cdl -p /dev/disk/by-path/ccw-0.0.4b2e
Drive Geometry: 10017 Cylinders * 15 Heads =  150255 Tracks

I am going to format the device /dev/disk/by-path/ccw-0.0.4b2e in the following way:
Device number of device : 0x4b2e
Labelling device        : yes
Disk label              : VOL1
Disk identifier         : 0X4B2E
Extent start (trk no)   : 0
Extent end (trk no)     : 150254
Compatible Disk Layout  : yes
Blocksize               : 4096

--->> ATTENTION! <<---
All data of that device will be lost.
Type "yes" to continue, no will leave the disk untouched: yes
cyl    97 of  3338 |#----------------------------------------------|   2%
```

When the progress bar reaches the end and the format is complete, **dasdfmt** prints the following output:

```
Rereading the partition table...
Exiting...
```

Now, use **fdasd** to partition the DASD. You can create up to three partitions on a DASD. In our example here, we create one partition spanning the whole disk:

```
# fdasd -a /dev/disk/by-path/ccw-0.0.4b2e
reading volume label ..: VOL1
reading vtoc ..........: ok

auto-creating one partition for the whole disk...
writing volume label...
writing VTOC...
rereading partition table...
```

After a (low-level formatted) DASD is online, it can be used like any other disk under Linux. For example, you can create file systems, LVM physical volumes, or swap space on its partitions, for example **/dev/disk/by-path/ccw-0.0.4b2e-part1**. Never use the full DASD device (**dev/dasdb**) for anything but the commands **dasdfmt** and **fdasd**. If you want to use the entire DASD, create one partition spanning the entire drive as in the **fdasd** example above.

To add additional disks later without breaking existing disk entries in, for example, **/etc/fstab**, use the persistent device symbolic links under **/dev/disk/by-path/**.

20.4. PERSISTENTLY SETTING DASDS ONLINE

The above instructions described how to activate DASDs dynamically in a running system. However, such changes are not persistent and do not survive a reboot. Making changes to the DASD configuration persistent in your Linux system depends on whether the DASDs belong to the root file system. Those DASDs required for the root file system need to be activated very early during the boot process by the **initramfs** to be able to mount the root file system.

The **cio_ignore** commands are handled seamlessly for persistent device configurations and you do not need to free devices from the ignore list manually.

20.5. DASDS THAT ARE PART OF THE ROOT FILE SYSTEM

The file you have to modify to add DASDs that are part of the root file system has changed in Red Hat Enterprise Linux 8. Instead of editing the **/etc/zipl.conf** file, the new file to be edited, and its location, may be found by running the following commands:

```
# machine_id=$(cat /etc/machine-id)
# kernel_version=$(uname -r)
# ls /boot/loader/entries/$machine_id-$kernel_version.conf
```

There is one boot option to activate DASDs early in the boot process: **rd.dasd=**. This option takes a Direct Access Storage Device (DASD) adapter device bus identifier. For multiple DASDs, specify the parameter multiple times, or use a comma separated list of bus IDs. To specify a range of DASDs, specify the first and the last bus ID. Below is an example of the **/boot/loader/entries/4ab74e52867b4f998e73e06cf23fd761-4.18.0-80.el8.s390x.conf** file for a system that uses physical volumes on partitions of two DASDs for an LVM volume group **vg_devel1** that contains a logical volume **lv_root** for the root file system.

```
title Red Hat Enterprise Linux (4.18.0-80.el8.s390x) 8.0 (Ootpa)
version 4.18.0-80.el8.s390x
linux /boot/vmlinuz-4.18.0-80.el8.s390x
initrd /boot/initramfs-4.18.0-80.el8.s390x.img
options root=/dev/mapper/vg_devel1-lv_root crashkernel=auto rd.dasd=0.0.0200 rd.dasd=0.0.0207
rd.lvm.lv=vg_devel1/lv_root rd.lvm.lv=vg_devel1/lv_swap cio_ignore=all,!condev
rd.znet=qeth,0.0.0a00,0.0.0a01,0.0.0a02,layer2=1,portno=0
id rhel-20181027190514-4.18.0-80.el8.s390x
grub_users $grub_users
grub_arg --unrestricted
grub_class kernel
```

To add another physical volume on a partition of a third DASD with device bus ID **0.0.202b**. To do this, add **rd.dasd=0.0.202b** to the parameters line of your boot kernel in **/boot/loader/entries/4ab74e52867b4f998e73e06cf23fd761-4.18.0-32.el8.s390x.conf**:

```
title Red Hat Enterprise Linux (4.18.0-80.el8.s390x) 8.0 (Ootpa)
version 4.18.0-80.el8.s390x
linux /boot/vmlinuz-4.18.0-80.el8.s390x
initrd /boot/initramfs-4.18.0-80.el8.s390x.img
options root=/dev/mapper/vg_devel1-lv_root crashkernel=auto rd.dasd=0.0.0200 rd.dasd=0.0.0207
rd.dasd=0.0.202b rd.lvm.lv=vg_devel1/lv_root rd.lvm.lv=vg_devel1/lv_swap cio_ignore=all,!condev
rd.znet=qeth,0.0.0a00,0.0.0a01,0.0.0a02,layer2=1,portno=0
id rhel-20181027190514-4.18.0-80.el8.s390x
grub_users $grub_users
grub_arg --unrestricted
grub_class kernel
```

Run **zipl** to apply the changes of the configuration file for the next IPL:

```
# zipl -V
Using config file '/etc/zipl.conf'
Using BLS config file '/boot/loader/entries/4ab74e52867b4f998e73e06cf23fd761-4.18.0-
80.el8.s390x.conf'
Target device information
  Device..........................: 5e:00
  Partition.......................: 5e:01
  Device name.....................: dasda
  Device driver name..............: dasd
  DASD device number..............: 0201
  Type............................: disk partition
  Disk layout.....................: ECKD/compatible disk layout
  Geometry - heads................: 15
  Geometry - sectors..............: 12
  Geometry - cylinders............: 13356
  Geometry - start................: 24
  File system block size..........: 4096
  Physical block size.............: 4096
  Device size in physical blocks..: 262152
Building bootmap in '/boot'
Building menu 'zipl-automatic-menu'
Adding #1: IPL section '4.18.0-80.el8.s390x' (default)
  initial ramdisk...: /boot/initramfs-4.18.0-80.el8.s390x.img
  kernel image......: /boot/vmlinuz-4.18.0-80.el8.s390x
  kernel parmline...: 'root=/dev/mapper/vg_devel1-lv_root crashkernel=auto rd.dasd=0.0.0200
rd.dasd=0.0.0207 rd.dasd=0.0.202b rd.lvm.lv=vg_devel1/lv_root rd.lvm.lv=vg_devel1/lv_swap
cio_ignore=all,!condev rd.znet=qeth,0.0.0a00,0.0.0a01,0.0.0a02,layer2=1,portno=0'
  component address:
    kernel image....: 0x00010000-0x0049afff
    parmline........: 0x0049b000-0x0049bfff
    initial ramdisk.: 0x004a0000-0x01a26fff
    internal loader.: 0x0000a000-0x0000cfff
Preparing boot menu
  Interactive prompt......: enabled
  Menu timeout............: 5 seconds
  Default configuration...: '4.18.0-80.el8.s390x'
Preparing boot device: dasda (0201).
Syncing disks...
Done.
```

20.6. DASDS THAT ARE NOT PART OF THE ROOT FILE SYSTEM

Direct Access Storage Devices (DASDs) that are not part of the root file system, that is, *data disks*, are persistently configured in the **/etc/dasd.conf** file. This file contains one DASD per line, where each line begins with the DASD's bus ID.

When adding a DASD to the **/etc/dasd.conf** file, use key-value pairs to specify the options for each entry. Separate the key and its value with an equal (=) sign. When adding multiple options, use a space or a tab to separate each option. For example, the **/etc/dasd.conf** file contains:

```
0.0.0207
0.0.0200 use_diag=1 readonly=1
```

Changes to the **/etc/dasd.conf** file take effect after a system reboot or after a new DASD is dynamically added by changing the system's I/O configuration (that is, the DASD is attached under z/VM).

Alternatively, to activate a DASD that you have added to the **/etc/dasd.conf** file, complete the following steps:

1. Remove the DASD from the list of ignored devices and make it visible using the **cio_ignore** utility:

   ```
   # cio_ignore -r device_number
   ```

   where ***device_number*** is the DASD device number.

   For example, if the device number is **021a**, run:

   ```
   # cio_ignore -r 021a
   ```

2. Activate the DASD by writing to the device's **uevent** attribute:

   ```
   # echo add > /sys/bus/ccw/devices/dasd-bus-ID/uevent
   ```

   where ***dasd-bus-ID*** is the DASD's bus ID.

   For example, if the bus ID is **0.0.021a**, run:

   ```
   # echo add > /sys/bus/ccw/devices/0.0.021a/uevent
   ```

20.7. FCP LUNS THAT ARE PART OF THE ROOT FILE SYSTEM

The only file you have to modify for adding FCP LUNs that are part of the root file system has changed in Red Hat Enterprise Linux 10. Instead of editing the **/etc/zipl.conf** file, the new file to be edited, and its location, may be found by running the following commands:

```
# machine_id=$(cat /etc/machine-id)
# kernel_version=$(uname -r)
# ls /boot/loader/entries/$machine_id-$kernel_version.conf
```

Red Hat Enterprise Linux provides a parameter to activate FCP LUNs early in the boot process: **rd.zfcp=**. The value is a comma-separated list containing the FCP device bus ID, the target WWPN as 16 digit hexadecimal number prefixed with **0x**, and the FCP LUN prefixed with 0x and padded with zeroes to the right to have 16 hexadecimal digits.

The WWPN and FCP LUN values are only necessary if the **zFCP** device is not configured in NPIV mode, when auto LUN scanning is disabled by the **zfcp.allow_lun_scan=0** kernel module parameter or when installing RHEL-9.0 or older releases. Otherwise they can be omitted, for example, **rd.zfcp=0.0.4000**. Below is an example of the **/boot/loader/entries/4ab74e52867b4f998e73e06cf23fd761-5.14.0-55.el9.s390x.conf** file for a system that uses a physical volume on a partition of an FCP-attached SCSI disk, with two paths, for an LVM volume group **vg_devel1** that contains a logical volume **lv_root** for the root file system.

```
title Red Hat Enterprise Linux (5.14.0-55.el9.s390x) 9.0 (Plow)
version 5.14.0-55.el9.s390x
linux /boot/vmlinuz-5.14.0-55.el9.s390x
initrd /boot/initramfs-5.14.0-55.el9.s390x.img
options root=/dev/mapper/vg_devel1-lv_root crashkernel=auto
rd.zfcp=0.0.fc00,0x5105074308c212e9,0x401040a000000000
```

```
rd.zfcp=0.0.fcd0,0x5105074308c2aee9,0x401040a000000000 rd.lvm.lv=vg_devel1/lv_root
rd.lvm.lv=vg_devel1/lv_swap cio_ignore=all,!condev
rd.znet=qeth,0.0.0a00,0.0.0a01,0.0.0a02,layer2=1,portno=0
id rhel-20181027190514-5.14.0-55.el9.s390x
grub_users $grub_users
grub_arg --unrestricted
grub_class kernel
```

To add another physical volume on a partition of a second FCP-attached SCSI disk with FCP LUN **0x401040a300000000** using the same two paths as the already existing physical volume, add **rd.zfcp=0.0.fc00,0x5105074308c212e9,0x401040a300000000** and **rd.zfcp=0.0.fcd0,0x5105074308c2aee9,0x401040a300000000** to the parameters line of your boot kernel in **/boot/loader/entries/4ab74e52867b4f998e73e06cf23fd761-5.14.0-55.el9.s390x.conf**. For example:

```
title Red Hat Enterprise Linux (5.14.0-55.el9.s390x) 9.0 (Plow)
version 5.14.0-55.el9.s390x
linux /boot/vmlinuz-5.14.0-55.el9.s390x
initrd /boot/initramfs-5.14.0-55.el9.s390x.img
options root=/dev/mapper/vg_devel1-lv_root crashkernel=auto
rd.zfcp=0.0.fc00,0x5105074308c212e9,0x401040a000000000
rd.zfcp=0.0.fcd0,0x5105074308c2aee9,0x401040a000000000
rd.zfcp=0.0.fc00,0x5105074308c212e9,0x401040a300000000
rd.zfcp=0.0.fcd0,0x5105074308c2aee9,0x401040a300000000 rd.lvm.lv=vg_devel1/lv_root
rd.lvm.lv=vg_devel1/lv_swap cio_ignore=all,!condev
rd.znet=qeth,0.0.0a00,0.0.0a01,0.0.0a02,layer2=1,portno=0
id rhel-20181027190514-5.14.0-55.el9.s390x
grub_users $grub_users
grub_arg --unrestricted
grub_class kernel
```

WARNING

Make sure the length of the kernel command line in the configuration file does not exceed 896 bytes. Otherwise, the boot loader cannot be saved, and the installation fails.

- Run **dracut -f** to update the initial RAM disk of your target kernel.
- Run **zipl** to apply the changes of the configuration file for the next IPL:

```
# zipl -V
Using config file '/etc/zipl.conf'
Using BLS config file '/boot/loader/entries/4ab74e52867b4f998e73e06cf23fd761-5.14.0-
55.el9.s390x.conf'
Run /lib/s390-tools/zipl_helper.device-mapper /boot
Target device information
Device..........................: fd:00
Partition.......................: fd:01
Device name.....................: dm-0
Device driver name..............: device-mapper
Type...........................: disk partition
Disk layout.....................: SCSI disk layout
Geometry - start................: 2048
File system block size..........: 4096
Physical block size.............: 512
Device size in physical blocks..: 10074112
Building bootmap in '/boot/'
Building menu 'zipl-automatic-menu'
Adding #1: IPL section '5.14.0-55.el9.s390x' (default)
kernel image......: /boot/vmlinuz-5.14.0-55.el9.s390x
kernel parmline...: 'root=/dev/mapper/vg_devel1-lv_root crashkernel=auto
rd.zfcp=0.0.fc00,0x5105074308c212e9,0x401040a000000000
rd.zfcp=0.0.fcd0,0x5105074308c2aee9,0x401040a000000000
rd.zfcp=0.0.fc00,0x5105074308c212e9,0x401040a300000000
rd.zfcp=0.0.fcd0,0x5105074308c2aee9,0x401040a300000000 rd.lvm.lv=vg_devel1/lv_root
rd.lvm.lv=vg_devel1/lv_swap cio_ignore=all,!condev
rd.znet=qeth,0.0.0a00,0.0.0a01,0.0.0a02,layer2=1,portno=0'
initial ramdisk...: /boot/initramfs-5.14.0-55.el9.s390x.img component address:
kernel image....: 0x00010000-0x007a21ff
parmline........: 0x00001000-0x000011ff
initial ramdisk.: 0x02000000-0x028f63ff
internal loader.: 0x0000a000-0x0000a3ff
Preparing boot device: dm-0.
Detected SCSI PCBIOS disk layout.
Writing SCSI master boot record.
Syncing disks...
Done.
```

WARNING

IBM Z systems use **zipl** to create a bootmap containing raw block addresses of the kernel and **initramfs**. If the **initramfs** is regenerated, its physical layout on disk might change. Because **dracut** does not automatically update the bootmap, manually run **zipl** after regenerating the **initramfs**. Failing to do so may result in a stale bootloader reference and cause the system to fail to boot.

Additional resources

- Kernel panic on s390x after manual **initramfs** regeneration without zipl execution (Red Hat Knowledgebase)

20.8. FCP LUNS THAT ARE NOT PART OF THE ROOT FILE SYSTEM

FCP LUNs that are not part of the root file system, such as data disks, are persistently configured in the file **/etc/zfcp.conf**. It contains one FCP LUN per line. Each line contains the device bus ID of the FCP adapter, the target WWPN as 16 digit hexadecimal number prefixed with **0x**, and the FCP LUN prefixed with **0x** and padded with zeroes to the right to have 16 hexadecimal digits, separated by a space or tab.

The WWPN and FCP LUN values are only necessary if the **zFCP** device is not configured in NPIV mode, when **auto LUN** scanning is disabled by the **zfcp.allow_lun_scan=0** kernel module parameter or when installing RHEL-9.0 or older releases. Otherwise they can be omitted and only the device bus ID is mandatory.

Entries in **/etc/zfcp.conf** are activated and configured by udev when an FCP adapter is added to the system. At boot time, all FCP adapters visible to the system are added and trigger **udev**.

Example content of **/etc/zfcp.conf**:

```
0.0.fc00 0x5105074308c212e9 0x401040a000000000
0.0.fc00 0x5105074308c212e9 0x401040a100000000
0.0.fc00 0x5105074308c212e9 0x401040a300000000
0.0.fcd0 0x5105074308c2aee9 0x401040a000000000
0.0.fcd0 0x5105074308c2aee9 0x401040a100000000
0.0.fcd0 0x5105074308c2aee9 0x401040a300000000
0.0.4000
0.0.5000
```

Modifications of **/etc/zfcp.conf** only become effective after a reboot of the system or after the dynamic addition of a new FCP channel by changing the system's I/O configuration (for example, a channel is attached under z/VM). Alternatively, you can trigger the activation of a new entry in **/etc/zfcp.conf** for an FCP adapter which was previously not active, by executing the following commands:

1. Use the **zfcp_cio_free** utility to remove the FCP adapters from the list of ignored devices and make them visible to Linux:

   ```
   # zfcp_cio_free
   ```

2. To apply the additions from **/etc/zfcp.conf** to the running system, issue:

   ```
   # zfcpconf.sh
   ```

20.9. ADDING A QETH DEVICE

Add qeth network devices on 64-bit IBM Z systems to enable network connectivity by using OSA-Express features, HiperSockets, z/VM guest LAN, and z/VM VSWITCH. This configuration provides network access for IBM Z systems in various virtualization and networking environments.

For more information about the qeth device driver naming scheme, see Customizing boot parameters.

20.10. DYNAMICALLY ADDING A QETH DEVICE

You can dynamically add a **qeth** device to the system to make a new network interface available without rebooting. Adding devices dynamically supports maintaining connectivity and expanding capacity on running systems.

Procedure

1. Determine whether the **qeth** device driver modules are loaded. The following example shows loaded **qeth** modules:

   ```
   # lsmod | grep qeth
   qeth_l3              69632  0
   ```

```
qeth_l2              49152  1
qeth               131072  2 qeth_l3,qeth_l2
qdio                65536  3 qeth,qeth_l3,qeth_l2
ccwgroup             20480  1 qeth
```

If the output of the **lsmod** command shows that the **qeth** modules are not loaded, run the **modprobe** command to load them:

```
# modprobe qeth
```

2. Use the **cio_ignore** utility to remove the network channels from the list of ignored devices and make them visible to Linux:

```
# cio_ignore -r read_device_bus_id,write_device_bus_id,data_device_bus_id
```

Replace *read_device_bus_id*,*write_device_bus_id*,*data_device_bus_id* with the three device bus IDs representing a network device. For example, if the *read_device_bus_id* is **0.0.f500**, the *write_device_bus_id* is **0.0.f501**, and the *data_device_bus_id* is **0.0.f502**:

```
# cio_ignore -r 0.0.f500,0.0.f501,0.0.f502
```

3. Use the **znetconf** utility to sense and list candidate configurations for network devices:

```
# znetconf -u
Scanning for network devices...
Device IDs                Type    Card Type      CHPID Drv.
------------------------------------------------------------
0.0.f500,0.0.f501,0.0.f502 1731/01 OSA (QDIO)        00 qeth
0.0.f503,0.0.f504,0.0.f505 1731/01 OSA (QDIO)        01 qeth
0.0.0400,0.0.0401,0.0.0402 1731/05 HiperSockets      02 qeth
```

4. Select the configuration you want to work with and use **znetconf** to apply the configuration and to bring the configured group device online as network device.

```
# znetconf -a f500
Scanning for network devices...
Successfully configured device 0.0.f500 (encf500)
```

5. Optional: You can also pass arguments that are configured on the group device before it is set online:

```
# znetconf -a f500 -o portname=myname
Scanning for network devices...
Successfully configured device 0.0.f500 (encf500)
```

Now you can continue to configure the **encf500** network interface. Alternatively, you can use **sysfs** attributes to set the device online as follows:

6. Create a **qeth** group device:

```
# echo read_device_bus_id,write_device_bus_id,data_device_bus_id >
/sys/bus/ccwgroup/drivers/qeth/group
```

For example:

```
# echo 0.0.f500,0.0.f501,0.0.f502 > /sys/bus/ccwgroup/drivers/qeth/group
```

7. Next, verify that the **qeth** group device was created properly by looking for the read channel:

```
# ls /sys/bus/ccwgroup/drivers/qeth/0.0.f500
```

You can optionally set additional parameters and features, depending on the way you are setting up your system and the features you require, such as:

- **portno**
- **layer2**
- **portname**

8. Bring the device online by writing **1** to the online **sysfs** attribute:

```
# echo 1 > /sys/bus/ccwgroup/drivers/qeth/0.0.f500/online
```

9. Then verify the state of the device:

```
# cat /sys/bus/ccwgroup/drivers/qeth/0.0.f500/online
          1
```

A return value of **1** indicates that the device is online, while a return value **0** indicates that the device is offline.

10. Find the interface name that was assigned to the device:

```
# cat /sys/bus/ccwgroup/drivers/qeth/0.0.f500/if_name
encf500
```

Now you can continue to configure the **encf500** network interface.

The following command from the **s390utils** package shows the most important settings of your **qeth** device:

```
# lsqeth encf500
Device name                     : encf500
-------------------------------------------------
card_type               : OSD_1000
cdev0                   : 0.0.f500
cdev1                   : 0.0.f501
cdev2                   : 0.0.f502
chpid                   : 76
online                  : 1
portname                : OSAPORT
portno                  : 0
state                   : UP (LAN ONLINE)
priority_queueing       : always queue 0
```

```
buffer_count          : 16
layer2                : 1
isolation             : none
```

20.11. PERSISTENTLY ADDING A QETH DEVICE

To make a new **qeth** device persistent, create a configuration file for the new interface. The network interface configuration files are placed in the **/etc/NetworkManager/system-connections/** directory.

The network configuration files use the naming convention *device*.nmconnection, where *device* is the value found in the interface-name file in the *qeth* group device that was created earlier, for example *enc9a0*. The *cio_ignore* commands are handled seamlessly for persistent device configurations and you do not need to free devices from the ignore list manually.

If a configuration file for another device of the same type already exists, copy it to the new name and edit it:

```
# cd /etc/NetworkManager/system-connections/
# cp enc9a0.nmconnection enc600.nmconnection
```

To learn IDs of your network devices, use the *lsqeth* utility:

```
# lsqeth -p
devices                    CHPID interface        cardtype       port chksum prio-q'ing rtr4 rtr6 lay'2 cnt
-------------------------- ----- ---------------- -------------- ---- ------ ---------- ---- ---- ----- -----
0.0.09a0/0.0.09a1/0.0.09a2 x00   enc9a0    Virt.NIC QDIO  0    sw     always_q_2 n/a  n/a  1     64
0.0.0600/0.0.0601/0.0.0602 x00   enc600    Virt.NIC QDIO  0    sw     always_q_2 n/a  n/a  1     64
```

If you do not have a similar device defined, create a new file. Use this example:

```
[connection]
type=ethernet
interface-name=enc600

[ipv4]
address1=10.12.20.136/24,10.12.20.1
dns=10.12.20.53;
method=manual

[ethernet]
mac-address=00:53:00:8f:fa:66
```

Edit the new *enc600.nmconnection* file as follows:

1. Ensure the new connection file is owned by **root:root**:

   ```
   # chown root:root /etc/NetworkManager/system-connections/enc600.nmconnection
   ```

2. Add more details in this file or modify these parameters based on your connection requirements.

3. Save the file.

4. Reload the connection profile:

```
# nmcli connection reload
```

5. To view complete details of the connection newly added, enter:

```
# nmcli connection show enc600
```

Changes to the *enc600.nmconnection* file become effective after either rebooting the system, dynamic addition of new network device channels by changing the system's I/O configuration (for example, attaching under z/VM), or reloading network connections. Alternatively, you can trigger the activation of *enc600.nmconnection* for network channels, which were previously not active yet, by executing the following commands:

1. Use the **cio_ignore** utility to remove the network channels from the list of ignored devices and make them visible to Linux:

```
# cio_ignore -r read_device_bus_id,write_device_bus_id,data_device_bus_id
```

   Replace *read_device_bus_id*, *write_device_bus_id*, *data_device_bus_id* with the three device bus IDs representing a network device. For example, if the *read_device_bus_id* is **0.0.0600**, the *write_device_bus_id* is **0.0.0601**, and the *data_device_bus_id* is **0.0.0602**:

```
#  cio_ignore -r 0.0.0600,0.0.0601,0.0.0602
```

2. To trigger the uevent that activates the change, issue:

```
# echo add > /sys/bus/ccw/devices/read-channel/uevent
```

   For example:

```
# echo add > /sys/bus/ccw/devices/0.0.0600/uevent
```

3. Check the status of the network device:

```
# lsqeth
```

4. If the default route information has changed, you must also update the *ipaddress1* parameters in both the **[ipv4]** and **[ipv6]** sections of the **/etc/NetworkManager/system-connections/<profile_name>.nmconnection** file accordingly:

```
[ipv4]
address1=10.12.20.136/24,10.12.20.1
[ipv6]
address1=2001:db8:1::1,2001:db8:1::fffe
```

5. Now start the new interface:

```
# nmcli connection up enc600
```

6. Check the status of the interface:

```
# ip addr show enc600
3: enc600:  <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
```

```
UP group default qlen 1000
link/ether 3c:97:0e:51:38:17 brd ff:ff:ff:ff:ff:ff
10.12.20.136/24 brd 10.12.20.1 scope global dynamic enc600
valid_lft 81487sec preferred_lft 81487sec
inet6 1574:12:5:1185:3e97:eff:fe51:3817/64 scope global noprefixroute dynamic
valid_lft 2591994sec preferred_lft 604794sec
inet6 fe45::a455:eff:d078:3847/64 scope link
valid_lft forever preferred_lft forever
```

7. Check the routing for the new interface:

```
# ip route
default via 10.12.20.136 dev enc600 proto dhcp src
```

8. Verify your changes by using the **ping** utility to ping the gateway or another host on the subnet of the new device:

```
# ping -c 1 10.12.20.136
PING 10.12.20.136 (10.12.20.136) 56(84) bytes of data.
64 bytes from 10.12.20.136: icmp_seq=0 ttl=63 time=8.07 ms
```

9. If the default route information has changed, you must also update **/etc/sysconfig/network** accordingly.
   For more information, see the **nm-settings-keyfile** man page on your system.

20.12. CONFIGURING AN 64-BIT IBM Z NETWORK DEVICE FOR NETWORK ROOT FILE SYSTEM

To add a network device that is required to access the root file system, you only have to change the boot options. The boot options can be in a parameter file, however, the **/etc/zipl.conf** file no longer contains specifications of the boot records.

The file that needs to be modified can be located using the following commands:

```
# machine_id=$(cat /etc/machine-id)
# kernel_version=$(uname -r)
# ls /boot/loader/entries/$machine_id-$kernel_version.conf
```

Dracut, the **mkinitrd** successor that provides the functionality in the initramfs that in turn replaces **initrd**, provides a boot parameter to activate network devices on 64-bit IBM Z early in the boot process: **rd.znet=**.

As input, this parameter takes a comma-separated list of the **NETTYPE** (qeth, lcs, ctc), two (lcs, ctc) or three (qeth) device bus IDs, and optional additional parameters consisting of key-value pairs corresponding to network device sysfs attributes. This parameter configures and activates the 64-bit IBM Z network hardware. The configuration of IP addresses and other network specifics works the same as for other platforms. See the **dracut** documentation for more details.

The **cio_ignore** commands for the network channels are handled seamlessly on boot.

Example boot options for a root file system accessed over the network through NFS:

```
root=10.16.105.196:/nfs/nfs_root cio_ignore=all,!condev
```

```
rd.znet=qeth,0.0.0a00,0.0.0a01,0.0.0a02,layer2=1,portno=0,portname=OSAPORT
ip=10.16.105.197:10.16.105.196:10.16.111.254:255.255.248.0:nfs-server.subdomain.domain:enc9a0:n
one rd_NO_LUKS rd_NO_LVM rd_NO_MD rd_NO_DM LANG=en_US.UTF-8
SYSFONT=latarcyrheb-sun16 KEYTABLE=us
```

20.13. ADDITIONAL RESOURCES

- Device Drivers, Features, and Commands on RHEL

CHAPTER 21. KICKSTART SCRIPT FILE FORMAT REFERENCE

This reference describes in detail the kickstart file format.

21.1. KICKSTART FILE FORMAT

Kickstart scripts are plain text files that contain keywords recognized by the installation program, which serve as directions for the installation. Any text editor able to save files as ASCII text, such as **Gedit** or **vim** on Linux systems or **Notepad** on Windows systems, can be used to create and edit Kickstart files. The file name of your Kickstart configuration does not matter; however, it is recommended to use a simple name as you will need to specify this name later in other configuration files or dialogs.

Commands

Commands are keywords that serve as directions for installation. Each command must be on a single line. Commands can take options. Specifying commands and options is similar to using Linux commands in a shell.

Sections

Certain special commands that begin with the percent **%** character start a section. Interpretation of commands in sections is different from commands placed outside sections. Every section must be finished with the **%end** command.

Section types

The available sections are:

- **Add-on sections**. These sections use the **%addon *addon_name*** command.
- **Package selection sections**. Start with **%packages**. Use it to list packages for installation, including indirect means such as package groups or modules.
- **Script sections**. These start with **%pre**, **%pre-install**, **%post**, and **%onerror**. These sections are not required.
- **Certificate sections**. These sections could be used to include certificates to be used by installation. For more information, see Kickstart certificates section.

Command section

The command section is a term used for the commands in the Kickstart file that are not part of any script section or **%packages** section.

Script section count and ordering

All sections except the command section are optional and can be present multiple times. When a particular type of script section is to be evaluated, all sections of that type present in the Kickstart are evaluated in order of appearance: two **%post** sections are evaluated one after another, in the order as they appear. However, you do not have to specify the various types of script sections in any order: it does not matter if there are **%post** sections before **%pre** sections.

Comments

Kickstart comments are lines starting with the hash **#** character. These lines are ignored by the installation program.

Items that are not required can be omitted. Omitting any required item results in the installation program changing to the interactive mode so that the user can provide an answer to the related item, just as during a regular interactive installation. It is also possible to declare the kickstart script as non-interactive with the **cmdline** command. In non-interactive mode, any missing answer aborts the installation process.

NOTE

If user interaction is needed during kickstart installation in text or graphical mode, enter only the windows where updates are mandatory to complete the installation. Entering spokes might lead to resetting the kickstart configuration. Resetting of the configuration applies specifically to the kickstart commands related to storage after entering the Installation Destination window.

21.2. PACKAGE SELECTION IN KICKSTART

Kickstart uses sections started by the **%packages** command for selecting packages to install. You can install packages, groups, and environments this way.

21.2.1. Package selection section

Use the **%packages** command to begin a Kickstart section which describes the software packages to be installed. The **%packages** section must end with the **%end** command.

You can specify packages by environment, group, module stream, module profile, or by their package names. Several environments and groups that contain related packages are defined. See the ***repository*/repodata/*-comps-*repository.architecture*.xml** file on the Red Hat Enterprise Linux 10 Installation DVD for a list of environments and groups.

The ***-comps-*repository.architecture*.xml** file contains a structure describing available environments (marked by the **<environment>** tag) and groups (the **<group>** tag). Each entry has an ID, user visibility value, name, description, and package list. If the group is selected for installation, the packages marked **mandatory** in the package list are always installed, the packages marked **default** are installed if they are not specifically excluded elsewhere, and the packages marked **optional** must be specifically included elsewhere even when the group is selected.

You can specify a package group or environment by using either its ID (the **<id>** tag) or name (the **<name>** tag).

If you are not sure what package should be installed, Red Hat recommends you to select the **Minimal Install** environment. **Minimal Install** provides only the packages which are essential for running Red Hat Enterprise Linux 10. This will substantially reduce the chance of the system being affected by a vulnerability. If necessary, additional packages can be added later after the installation. For more details on **Minimal Install**, see the Installing the Minimum Amount of Packages Required .

IMPORTANT

To install a 32-bit package on a 64-bit system:

- specify the **--multilib** option for the **%packages** section
- append the package name with the 32-bit architecture for which the package was built; for example, **glibc.i686**

21.2.2. Package selection commands

You can use the following commands within the **%packages** section of a Kickstart file.

Specifying an environment

Specify an entire environment to be installed as a line starting with the **@^** symbols:

```
%packages
@^Infrastructure Server
%end
```

This installs all packages which are part of the **Infrastructure Server** environment. All available environments are described in the ***repository*/repodata/*-comps-*repository.architecture*.xml** file on the Red Hat Enterprise Linux 10 Installation DVD.

Only a single environment should be specified in the Kickstart file. If more environments are specified, only the last specified environment is used.

Specifying groups

Specify groups, one entry to a line, starting with an **@** symbol, and then the full group name or group id as given in the ***-comps-*repository.architecture*.xml** file. For example:

```
%packages
@X Window System
@Desktop
@Sound and Video
%end
```

The **Core** group is always selected - it is not necessary to specify it in the **%packages** section.

Specifying individual packages

Specify individual packages by name, one entry to a line. You can use the asterisk character (*) as a wildcard in package names. For example:

```
%packages
sqlite
curl
aspell
docbook*
%end
```

The **docbook*** entry includes the packages **docbook-dtds** and **docbook-style** that match the pattern represented with the wildcard.

Specifying profiles of module streams

Specify profiles for module streams, one entry to a line, using the syntax for profiles:

```
%packages
@module:stream/profile
%end
```

This installs all packages listed in the specified profile of the module stream.

- When a module has a default stream specified, you can leave it out. When the default stream is not specified, you must specify it.

- When a module stream has a default profile specified, you can leave it out. When the default profile is not specified, you must specify it.
- Installing a module multiple times with different streams is not possible.
- Installing multiple profiles of the same module and stream is possible.

Modules and groups use the same syntax starting with the **@** symbol. When a module and a package group exist with the same name, the module takes precedence.

In Red Hat Enterprise Linux 10, modules are present only in the AppStream repository. To list available modules, use the **dnf module list** command on an installed Red Hat Enterprise Linux 10 system.

It is also possible to enable module streams using the **module** Kickstart command and then install packages contained in the module stream by naming them directly.

Excluding environments, groups, or packages

Use a leading dash (**-**) to specify packages or groups to exclude from the installation. For example:

```
%packages
-@Graphical Administration Tools
-autofs
-ipa*compat
%end
```

IMPORTANT

Installing all available packages using only ***** in a Kickstart file is not supported.

You can change the default behavior of the **%packages** section by using several options. Some options work for the entire package selection, others are used with only specific groups.

Additional resources

- Managing Software with the DNF Tool

21.2.3. Common package selection options

You can use the following options in the **%packages** sections. To use an option, append it to the start of the package selection section.

For example:

+

```
%packages --multilib --ignoremissing
```

--default

Install the default set of packages. This corresponds to the package set which would be installed if no other selections were made in the **Package Selection** screen during an interactive installation.

--excludedocs

Do not install any documentation contained within packages. In most cases, this excludes any files normally installed in the **/usr/share/doc** directory, but the specific files to be excluded depend on individual packages.

--ignoremissing

Ignore any packages, groups, module streams, module profiles, and environments missing in the installation source, instead of halting the installation to ask if the installation should be aborted or continued.

--inst-langs

Specify a list of languages to install. This is different from package group level selections. This option does not describe which package groups should be installed; instead, it sets RPM macros controlling which translation files from individual packages should be installed.

--multilib

Configure the installed system for multilib packages, to allow installing 32-bit packages on a 64-bit system, and install packages specified in this section as such.
Normally, on an AMD64 and Intel 64 system, you can install only the x86_64 and the noarch packages. However, with the --multilib option, you can automatically install the 32-bit AMD and the i686 Intel system packages available, if any.

This only applies to packages explicitly specified in the **%packages** section. Packages which are only being installed as dependencies without being specified in the Kickstart file are only installed in architecture versions in which they are needed, even if they are available for more architectures.

You can configure Anaconda to install packages in **multilib** mode during the installation of the system. Use one of the following options to enable **multilib** mode:

1. Configure Kickstart file with the following lines:

```
%packages --multilib --default
%end
```

2. Add the inst.multilib boot option during booting the installation image.

--nocore

Disables installation of the **@Core** package group which is otherwise always installed by default. Disabling the **@Core** package group with **--nocore** should be only used for creating lightweight containers; installing a desktop or server system with **--nocore** will result in an unusable system.

NOTE

Using **-@Core** to exclude packages in the **@Core** package group does not work. The only way to exclude the **@Core** package group is with the **--nocore** option. Also, the **@Core** package group is defined as a minimal set of packages needed for installing a working system. It is not related in any way to core packages as defined in the Package Manifest and Scope of Coverage Details.

--exclude-weakdeps

Disables installation of packages from weak dependencies. These are packages linked to the selected package set by Recommends and Supplements flags. By default weak dependencies will be installed.

--retries=

Sets the number of times **dnf** will attempt to download packages (retries). The default value is 10. This option only applies during the installation, and will not affect **dnf** configuration on the installed system.

--timeout=

Sets the **dnf** timeout in seconds. The default value is 30. This option only applies during the installation, and will not affect **dnf** configuration on the installed system.

21.2.4. Options for specific package groups

The options in this list only apply to a single package group. Instead of using them at the **%packages** command in the Kickstart file, append them to the group name. For example:

```
%packages
@Graphical Administration Tools --optional
%end
```

--nodefaults

Only install the group's mandatory packages, not the default selections.

--optional

Install packages marked as optional in the group definition in the ***-comps-*repository*.*architecture*.xml** file, in addition to installing the default selections.
Some package groups, such as **Scientific Support**, do not have any mandatory or default packages specified - only optional packages. In this case the **--optional** option must always be used, otherwise no packages from this group will be installed.

IMPORTANT

The **--nodefaults** and **--optional** options cannot be used together. You can install only mandatory packages during the installation using **--nodefaults** and install the optional packages on the installed system post installation.

21.2.5. Installing Kernel-64k on ARM using Kickstart

RHEL offers the ARM64 hardware architecture to support workloads that require large physical memory configuration for optimal performance. Such large memory configuration requires the use of a large MMU page size (64k).

While installing RHEL, you can select the **kernel-64k** package to install with a kernel supporting 64k page size.

Procedure

- In the **%packages** section of the kickstart file, add the following list of packages:

```
%packages
kernel-64k
-kmod-kvdo
-vdo
-kernel
%end
```

Verification

- To verify the page size, after installation is completed and the system is rebooted, open the terminal and run:

  ```
  $ getconf PAGESIZE
  65536
  ```

  The output **65536** indicates that the 64k kernel is in use.

- To verify that the swap partition is enabled, enter:

  ```
  $ free
                total        used        free      shared  buff/cache   available
  Mem:       35756352     3677184    34774848       25792      237120    32079168
  Swap:       6504384           0     6504384
  ```

The total and free columns are non-zero, which indicates the swap is enabled successfully.

21.3. SCRIPTS IN KICKSTART FILE

A kickstart file can include the following scripts:

- **%pre**
- **%pre-install**
- **%post**

This section provides the following details about the scripts:

- Execution time
- Types of commands that can be included in the script
- Purpose of the script
- Script options

21.3.1. %pre script

The **%pre** scripts are run on the system immediately after the Kickstart file has been loaded, but before it is completely parsed and installation begins. Each of these sections must start with **%pre** and end with **%end**.

The **%pre** script can be used for activation and configuration of networking and storage devices. It is also possible to run scripts by using interpreters available in the installation environment. Adding a **%pre** script can be useful if you have networking and storage that needs special configuration before proceeding with the installation, or have a script that, for example, sets up additional logging parameters or environment variables.

Debugging problems with **%pre** scripts can be difficult, so it is recommended only to use a **%pre** script when necessary.

IMPORTANT

The **%pre** section of Kickstart is executed at the stage of installation which happens after the installer image (**inst.stage2**) is fetched: it means **after** root switches to the installer environment (the installer image) and **after** the **Anaconda** installer itself starts. Then the configuration in **%pre** is applied and can be used to fetch packages from installation repositories configured, for example, by URL in Kickstart. However, it **cannot** be used to configure the network to fetch the image (**inst.stage2**) from the network.

Commands related to networking, storage, and file systems are available to use in the **%pre** script, in addition to most of the utilities in the installation environment **/sbin** and **/bin** directories.

You can access the network in the **%pre** section. However, the name service has not been configured at this point, so only IP addresses work, not URLs.

NOTE

The pre script does not run in the chroot environment.

21.3.1.1. %pre script section options

The following options can be used to change the behavior of pre-installation scripts. To use an option, append it to the **%pre** line at the beginning of the script. For example:

```
%pre --interpreter=/usr/libexec/platform-python
-- Python script omitted --
%end
```

--interpreter=

Allows you to specify a different scripting language, such as Python. Any scripting language available on the system can be used; in most cases, these are **/usr/bin/sh**, **/usr/bin/bash**, and **/usr/libexec/platform-python**.
Note that the **platform-python** interpreter uses Python version 3.6. You must change your Python scripts from previous RHEL versions for the new path and version. Additionally, **platform-python** is meant for system tools: Use the **python36** package outside the installation environment. For more details about Python in Red Hat Enterprise Linux, see Introduction to Python.

--erroronfail

Displays an error and halts the installation if the script fails. The error message will direct you to where the cause of the failure is logged. The installed system might get into an unstable and unbootable state. You can use the **inst.nokill** option to debug the script.

--log=

Logs the script's output into the specified log file. For example:

```
%pre --log=/tmp/ks-pre.log
```

21.3.2. %pre-install script

The commands in the **pre-install** script are run after the following tasks are complete:

- System is partitioned
- Filesystems are created and mounted under /mnt/sysroot
- Network has been configured according to any boot options and kickstart commands

Each of the **%pre-install** sections must start with **%pre-install** and end with **%end**.

The **%pre-install** scripts can be used to modify the installation, and to add users and groups with guaranteed IDs before package installation.

It is recommended to use the **%post** scripts for any modifications required in the installation. Use the **%pre-install** script only if the **%post** script falls short for the required modifications.

The **pre-install** script does not run in the chroot environment.

21.3.2.1. %pre-install script section options

The following options can be used to change the behavior of **pre-install** scripts. To use an option, append it to the **%pre-install** line at the beginning of the script. For example:

```
%pre-install --interpreter=/usr/libexec/platform-python
-- Python script omitted --
%end
```

You can have multiple **%pre-install** sections, with same or different interpreters. They are evaluated in their order of appearance in the Kickstart file.

--interpreter=

Allows you to specify a different scripting language, such as Python. Any scripting language available on the system can be used; in most cases, these are **/usr/bin/sh**, **/usr/bin/bash**, and **/usr/libexec/platform-python**.
The **platform-python** interpreter uses Python version 3.6. You must change your Python scripts from previous RHEL versions for the new path and version. Additionally, **platform-python** is meant for system tools: Use the **python36** package outside the installation environment. For more details about Python in Red Hat Enterprise Linux, see Introduction to Python.

--erroronfail

Displays an error and halts the installation if the script fails. The error message will direct you to where the cause of the failure is logged. The installed system might get into an unstable and unbootable state. You can use the **inst.nokill** option to debug the script.

--log=

Logs the script’s output into the specified log file. For example:

```
%pre-install --log=/mnt/sysroot/root/ks-pre.log
```

21.3.3. %post script

The %post script is a post-installation script that is run after the installation is complete, but before the system is rebooted for the first time. You can use this section to run tasks such as system subscription.

You have the option of adding commands to run on the system once the installation is complete, but before the system is rebooted for the first time. This section must start with **%post** and end with **%end**.

The **%post** section is useful for functions such as installing additional software or configuring an additional name server. The post-install script is run in a **chroot** environment, therefore, performing tasks such as copying scripts or RPM packages from the installation media do not work by default. You can change this behavior by using the **--nochroot** option as described below. Then the **%post** script will run in the installation environment, not in **chroot** on the installed target system.

Because the post-install script runs in a **chroot** environment, most **systemctl** commands will refuse to perform any action.

During execution of the **%post** section, the installation media must be still inserted.

21.3.3.1. %post script section options

The following options can be used to change the behavior of post-installation scripts. To use an option, append it to the **%post** line at the beginning of the script. For example:

```
%post --interpreter=/usr/libexec/platform-python
-- Python script omitted --
%end
```

--interpreter=

Allows you to specify a different scripting language, such as Python. For example:

```
%post --interpreter=/usr/libexec/platform-python
```

Any scripting language available on the system can be used; in most cases, these are **/usr/bin/sh**, **/usr/bin/bash**, and **/usr/libexec/platform-python**.

The **platform-python** interpreter uses Python version 3.6. You must change your Python scripts from previous RHEL versions for the new path and version. Additionally, **platform-python** is meant for system tools: Use the **python36** package outside the installation environment. For more details about Python in Red Hat Enterprise Linux, see Introduction to Python.

--nochroot

Allows you to specify commands that you would like to run outside of the chroot environment. The following example copies the file /etc/resolv.conf to the file system that was just installed.

```
%post --nochroot
cp /etc/resolv.conf /mnt/sysroot/etc/resolv.conf
%end
```

--erroronfail

Displays an error and halts the installation if the script fails. The error message will direct you to where the cause of the failure is logged. The installed system might get into an unstable and unbootable state. You can use the **inst.nokill** option to debug the script.

--log=

Logs the script's output into the specified log file. The path of the log file must take into account whether or not you use the **--nochroot** option. For example, without **--nochroot**:

```
%post --log=/root/ks-post.log
```

and with **--nochroot**:

```
%post --nochroot --log=/mnt/sysroot/root/ks-post.log
```

21.3.3.2. Example: Mounting NFS in a post-install script

This example of a **%post** section mounts an NFS share and executes a script named **runme** located at **/usr/new-machines/** on the share. The NFS file locking is not supported while in Kickstart mode, therefore the **-o nolock** option is required.

```
# Start of the %post section with logging into /root/ks-post.log
%post --log=/root/ks-post.log

# Mount an NFS share
mkdir /mnt/temp
mount -o nolock 10.10.0.2:/usr/new-machines /mnt/temp
openvt -s -w -- /mnt/temp/runme
umount /mnt/temp

# End of the %post section
%end
```

21.4. KICKSTART ERROR HANDLING SECTION

Starting with Red Hat Enterprise Linux 7, Kickstart installations run custom scripts when any fatal error encounters in the installation program. Example scenarios include requesting the installation of a missing package, RDP failing to start if specified in the configuration, or an error while scanning storage devices. In case of such events, installation aborts. To analyze these events, the installation program runs all **%onerror** scripts chronologically as provided in the Kickstart file. In the event of traceback, you can run the **%onerror** scripts.

Each **%onerror** script is required to end with **%end**.

You can trigger command-line mode by using inst.cmdline; in this mode, all errors are treated as fatal by default.

Error handling sections accept the following options:

--erroronfail

Displays an error and halts the installation if the script fails. The error message will direct you to where the cause of the failure is logged. The installed system might get into an unstable and unbootable state. You can use the **inst.nokill** option to debug the script.

--interpreter=

Allows you to specify a different scripting language, such as Python. For example:

```
%onerror --interpreter=/usr/libexec/platform-python
```

Any scripting language available on the system can be used; in most cases, these are **/usr/bin/sh**, **/usr/bin/bash**, and **/usr/libexec/platform-python**.

The **platform-python** interpreter uses Python version 3.6. You must change your Python scripts from previous RHEL versions for the new path and version. Additionally, **platform-python** is meant for system tools: Use the **python36** package outside the installation environment. For more details about Python in Red Hat Enterprise Linux, see Introduction to Python.

--log=

Logs the script's output into the specified log file.

21.5. KICKSTART ADD-ON SECTIONS

Starting with Red Hat Enterprise Linux 7, Kickstart installations support add-ons. These add-ons can expand the basic Kickstart (and Anaconda) functionality in many ways.

To use an add-on in your Kickstart file, use the **%addon *addon_name options*** command, and finish the command with an **%end** statement, similar to pre-installation and post-installation script sections. For example, if you want to use the Kdump add-on, which is distributed with Anaconda by default, use the following commands:

```
%addon com_redhat_kdump --enable --reserve-mb=auto
%end
```

The **%addon** command does not include any options of its own - all options are dependent on the actual add-on.

21.6. KICKSTART CERTIFICATES SECTION

The **%certificate** section specifies the certificates that will be installed into the installer environment and the installed system. The certificate content must be in Base64 ASCII encoding format. The certificate will be written into a **.pem** file. You can specify the location to install the certificate and file details by using the following options:

Options

- **--filename** FILENAME
  Specifies the name of the certificate file.

- **--dir** DIR
  Specifies the directory where the certificate should be installed.

Example Usage

A certificate bundle can be installed as part of a single %certificate section.

```
%certificate --dir /etc/pki/dns/extracted/pem/ --filename tls-ca-bundle.pem
-----BEGIN CERTIFICATE-----
MIIBhDCCASugAwIBAgIUO27jF1hTk3lMZ5O6yMCC87FdLZcwCgYIKoZIzj0EAwIw
EzERMA8GA1UEAwwIVHJ1c3RlZENBMB4XDTI1MDQwMTEyMDAwMFoXDTM1MDMzMDEy
MDAwMFowEzERMA8GA1UEAwwIVHJ1c3RlZENBMFYwEAYHKoZIzj0CAQYFK4EEAAoD
QgAErPu3eF/MBcNfFh1lYWfLzQz9O5sWu/bOXMlQ5iU6GZ4H17Q5aHpDJvuBq+nv
uZYucGrdKj6GSa2KoD2Rz6GxDQIDAQABozEwLzAOBgNVHQ8BAf8EBAMCAQYwDwYD
VR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAQYwCgYIKoZIzj0EAwIDRwAwRAIg
A+3Txsfb5XQxpyq1dhK1c9Wy+F0dHZBaM4lKl8Zlf50CIHZ+bE5+8q90f5oRO2/f
```

```
CZtCWZZRAquC/hXoNcGxM2b7
-----END CERTIFICATE-----
%end
```

CHAPTER 22. KICKSTART COMMANDS AND OPTIONS REFERENCE

This reference is a complete list of all Kickstart commands supported by the Red Hat Enterprise Linux installation program. The commands are sorted alphabetically in a few broad categories. If a command can fall under multiple categories, it is listed in all of them.

22.1. KICKSTART CHANGES

The following sections describe the changes in Kickstart commands and options in Red Hat Enterprise Linux 10.

22.1.1. Deprecated Kickstart commands and options

The following Kickstart commands and options have been deprecated in Red Hat Enterprise Linux 10.

Where only specific options are listed, the base command and its other options are still available and not deprecated.

module

Anaconda has deprecated its support for DNF modularity, and as a consequence the module kickstart command has been deprecated. This might impact you if you are using modules in the **%packages** section of your kickstart files or the module kickstart command.

22.1.2. Removed Kickstart commands and options

The following Kickstart commands and options have been removed in Red Hat Enterprise Linux 10.

auth or authconfig

As a replacement, use the **authselect** kickstart command.

inst.xdriver and **inst.usefbx**

The graphical system for the installation image switched from the Xorg server to a Wayland compositor. As Wayland operates without relying on X drivers, making it incompatible with loading any such drivers, the **inst.xdriver** option is no longer applicable. Additionally, the **inst.usefbx** boot option, previously used to load a generic framebuffer X driver, has also been removed.

Few options of the timezone command

The following options of the timezone Kickstart command have been removed:

- **--isUtc**: Use the option **--utc** instead.
- **--ntpservers**: Use the option **--ntp-server** of the **timesource** kickstart command instead.
- **--nontp**: Use the option **--ntp-disable** of the **timesource** kickstart command instead.

--level parameter of the logging command

The **--level** parameter of the logging kickstart command has been removed. It is no longer possible to set the level of logging of the installation process.

pwpolicy

The support for the deprecated **pwpolicy** Kickstart command has been removed.

%anaconda

Alternatively, use the kernel arguments and command line line options to update the configuration in the Anaconda configuration files.

network

The **--teamslaves** and **--teamconfig** options used for configuring team devices in the network kickstart command have been removed. To configure similar network settings, use the **--bondslaves** and **--bondopts** options to set up a Bond device.

%packages

the **--excludeWeakdeps** and **--instLangs** options used in the **%packages** section have been removed. To maintain similar functionality, use the updated options **--exclude-weakdeps** and **--inst-langs** instead.

btrfs

The support for the deprecated **btrfs** Kickstart command has been removed.

nvdimm

The support for reconfiguring NVDIMM devices during the Kickstart installation has been removed. However, the NVDIMM devices in the sector mode can still be usable in the installation program.

method

The support for the deprecated **method** Kickstart command has been removed.

vnc

VNC kickstart command is removed in RHEL 10. Use RDP instead.

22.2. KICKSTART COMMANDS FOR INSTALLATION PROGRAM CONFIGURATION AND FLOW CONTROL

The Kickstart commands in this list control the mode and course of installation, and what happens at its end.

22.2.1. cdrom

The **cdrom** Kickstart command is optional. It performs the installation from the first optical drive on the system. Use this command only once.

Syntax

```
cdrom
```

Notes

- This command has no options.
- To actually run the installation, you must specify one of **cdrom**, **harddrive**, **hmc**, **nfs**, **liveimg**, **ostreesetup**, **rhsm**, or **url** unless the **inst.repo** option is specified on the kernel command line.

22.2.2. cmdline

The **cmdline** Kickstart command is optional. It performs the installation in a completely non-interactive command line mode. Any prompt for interaction halts the installation. Use this command only once.

Syntax

```
cmdline
```

Notes

- For a fully automatic installation, you must either specify one of the available modes (**graphical**, **text**, or **cmdline**) in the Kickstart file, or you must use the **console=** boot option. If no mode is specified, the system will use graphical mode if possible, or prompt you to choose from RDP and text mode.
- This command has no options.
- This mode is useful on 64-bit IBM Z systems with the x3270 terminal.

22.2.3. driverdisk

The **driverdisk** Kickstart command is optional. Use it to provide additional drivers to the installation program.

Driver disks can be used during Kickstart installations to provide additional drivers not included by default. You must copy the driver disks contents to the root directory of a partition on the system’s disk. Then, you must use the **driverdisk** command to specify that the installation program should look for a driver disk and its location. Use this command only once.

Syntax

```
driverdisk [partition|--source=url|--biospart=biospart]
```

Options

You must specify the location of driver disk in one way out of these:

- *partition* - Partition containing the driver disk. The partition must be specified as a full path (for example, **/dev/sdb1**), *not* just the partition name (for example, **sdb1**).
- **--source=** - URL for the driver disk. Examples include:

  ```
  driverdisk --source=ftp://path/to/dd.img
  driverdisk --source=http://path/to/dd.img
  driverdisk --source=nfs:host:/path/to/dd.img
  ```

- **--biospart=** - BIOS partition containing the driver disk (for example, **82p2**).

Notes

You can load the driver disks from a local disk or a similar device instead of being loaded over the network or from **initrd**. To do this:

1. Load the driver disk on a disk drive, a USB or any similar device.
2. Set the label, for example, *DD*, to this device.
3. Add the following line to your Kickstart file:

```
driverdisk LABEL=DD:/e1000.rpm
```

Replace *DD* with a specific label and replace *e1000.rpm* with a specific name. Use anything supported by the **inst.repo** command instead of *LABEL* to specify your disk drive.

22.2.4. eula

The **eula** Kickstart command is optional. Use this option to accept the End User License Agreement (EULA) without user interaction. Specifying this option prevents Initial Setup from prompting you to accept the license agreement after you finish the installation and reboot the system for the first time. Use this command only once.

Syntax

```
eula [--agreed]
```

Options

- **--agreed** (required) - Accepts the EULA. This option must always be used, otherwise the **eula** command is meaningless.

22.2.5. firstboot

The **firstboot** Kickstart command is optional. It determines whether the **Initial Setup** application starts the first time the system is booted. If enabled, the **initial-setup** package must be installed. If not specified, this option is disabled by default. Use this command only once.

Syntax

```
firstboot OPTIONS
```

Options

- **--enable** or **--enabled** - Initial Setup is started the first time the system boots.
- **--disable** or **--disabled** - Initial Setup is not started the first time the system boots.
- **--reconfig** - Enable the Initial Setup to start at boot time in reconfiguration mode. This mode enables the root password, time & date, and networking & host name configuration options in addition to the default ones.

22.2.6. graphical

The **graphical** Kickstart command is optional. It performs the installation in graphical mode. Use this command only once.

Syntax

```
graphical [--non-interactive]
```

Options

- **--non-interactive** - Performs the installation in a completely non-interactive mode. This mode will terminate the installation when user interaction is required.

Notes

- For a fully automatic installation, you must either specify one of the available modes (**graphical**, **text**, or **cmdline**) in the Kickstart file, or you must use the **console=** boot option. If no mode is specified, the system will use graphical mode if possible, or prompt you to choose from RDP and text mode.

22.2.7. halt

The **halt** Kickstart command is optional. Use this command to halt the system after the installation has successfully completed. This is similar to a manual installation, where Anaconda displays a message and waits for the user to press a key before rebooting. During a Kickstart installation, if no completion method is specified, this option is used as the default. Use this command only once. This command has no options.

Syntax

```
halt
```

Notes

- The **halt** command is equivalent to the **shutdown -H** command. For more details, see the *shutdown(8)* man page on your system.
- For other completion methods, see the **poweroff**, **reboot**, and **shutdown** commands.

22.2.8. harddrive

The **harddrive** Kickstart command is optional. It performs the installation from a Red Hat installation tree or full installation ISO image on a local drive. The drive must be formatted with a file system the installation program can mount: **ext2**, **ext3**, **ext4**, **vfat**, or **xfs**. Use this command only once.

Syntax

```
harddrive OPTIONS
```

Mandatory options

- **--partition=** - Partition to install from (such as **sdb2**).
- **--dir=** - Directory containing the installation tree, or the ISO image of the full installation DVD.

Example

```
harddrive --partition=hdb2 --dir=/tmp/install-tree
```

Note

- To actually run the installation, you must specify one of **cdrom**, **harddrive**, **hmc**, **nfs**, **liveimg**, **ostreesetup**, **rhsm**, or **url** unless the **inst.repo** option is specified on the kernel command line.

22.2.9. liveimg

The **liveimg** Kickstart command is optional. It performs the installation from a disk image instead of packages. Use this command only once.

Syntax

```
liveimg --url=SOURCE [OPTIONS]
```

Mandatory options

- **--url=** – The location to install from. Supported protocols are **HTTP**, **HTTPS**, **FTP**, and **file**.

Optional options

- **--proxy=** – Specify an **HTTP**, **HTTPS** or **FTP** proxy to use while performing the installation.
- **--checksum=** – An argument with the **SHA256** checksum of the image file, used for verification.
- **--noverifyssl** – Disable SSL verification when connecting to an **HTTPS** server.

Example

```
liveimg --url=file:///images/install/squashfs.img --
checksum=03825f567f17705100de3308a20354b4d81ac9d8bed4bb4692b2381045e56197 --
noverifyssl
```

Notes

- The image can be the **squashfs.img** file from a live ISO image, a compressed tar file (**.tar**, **.tbz**, **.tgz**, **.txz**, **.tar.bz2**, **.tar.gz**, or **.tar.xz**.), or any file system that the installation media can mount. Supported file systems are **ext2**, **ext3**, **ext4**, **vfat**, and **xfs**.
- When using the **liveimg** installation mode with a driver disk, drivers on the disk will not automatically be included in the installed system. If necessary, these drivers should be installed manually, or in the **%post** section of a kickstart script.
- To actually run the installation, you must specify one of **cdrom**, **harddrive**, **hmc**, **nfs**, **liveimg**, **ostreesetup**, **rhsm**, or **url** unless the **inst.repo** option is specified on the kernel command line.

22.2.10. logging

The **logging** Kickstart command is optional. It controls the error logging of Anaconda during installation. It has no effect on the installed system. Use this command only once.

Logging is supported over TCP only. For remote logging, ensure that the port number that you specify in **--port=** option is open on the remote server. The default port is 514.

Syntax

```
logging OPTIONS
```

Optional options

- **--host=** - Send logging information to the given remote host, which must be running a syslogd process configured to accept remote logging.
- **--port=** - If the remote syslog process uses a port other than the default port, set it by using this option.

22.2.11. mediacheck

The **mediacheck** Kickstart command is optional. This command forces the installation program to perform a media check before starting the installation. This command requires that installations be attended, so it is disabled by default. Use this command only once. This command has no options.

Syntax

```
mediacheck
```

Note

- This Kickstart command is equivalent to the **rd.live.check** boot option.

22.2.12. nfs

The **nfs** Kickstart command is optional. It performs the installation from a specified NFS server. Use this command only once.

Syntax

```
nfs OPTIONS
```

Options

- **--server=** - Server from which to install (host name or IP).
- **--dir=** - Directory containing the **Packages/** directory of the installation tree.
- **--opts=** - Mount options to use for mounting the NFS export. (optional)

Example

```
nfs --server=nfsserver.example.com --dir=/tmp/install-tree
```

Note

- To actually run the installation, you must specify one of **cdrom**, **harddrive**, **hmc**, **nfs**, **liveimg**, **ostreesetup**, **rhsm**, or **url** unless the **inst.repo** option is specified on the kernel command line.

22.2.13. ostreesetup

The **ostreesetup** Kickstart command is optional. It is used to set up OStree-based installations. Use this command only once.

Syntax

```
ostreesetup --osname=OSNAME [--remote=REMOTE] --url=URL --ref=REF [--nogpg]
```

Mandatory options

- **--osname=*OSNAME*** - Management root for OS installation.
- **--url=*URL*** - URL of the repository to install from.
- **--ref=*REF*** - Name of the branch from the repository to be used for installation.

Optional options

- **--remote=*REMOTE*** - A remote repository location.
- **--nogpg** - Disable GPG key verification.
  For more information about the OStree tools, see the upstream documentation: https://ostreedev.github.io/ostree/

22.2.14. ostreecontainer

The **ostreecontainer** Kickstart command is optional. Use this command for OSTree installations from your custom containers.

IMPORTANT

The **ostreecontainer** is provided as a Technology Preview only. Technology Preview features are not supported with Red Hat production Service Level Agreements (SLAs), might not be functionally complete, and Red Hat does not recommend using them for production. These previews provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. See Technology Preview Features Support Scope on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

Syntax

```
ostreecontainer [--stateroot STATEROOT] --url URL [--transport TRANSPORT] [--remote
REMOTE] [--no-signature-verification]
```

Options

- **--no-signature-verification**: Use this option to disable verification for an ostree remote.
- **--stateroot**: Name of the state directory, also known as "osname". Default value is **default**.
- **--url**: Name of the container image for the registry transport. For example, **quay.io/exampleos/foo:latest**.
- **--transport**: The transport, for example, registry or oci. The default value is **registry**.
  When performing Kickstart installations, the following commands are available together with **ostreecontainer**:
- **graphical**, **text**, or **cmdline**
- **clearpart** and **zerombr**
- **autopart**
- **part**
- **logvol** and **volgroup**
- **reboot** and **shutdown**
- **lang**
- **rootpw**
- **sshkey**
- **bootloader** (available only with the **--append** optional parameter)
- **user** - When you specify a group within the user command, the user account can be assigned only to a group that already exists in the container image.
  The following Kickstart commands are unsupported together with **ostreecontainer**:
- **authconfig** and **authselect** (provide relevant configuration in the container image instead)
- **ostreesetup**
- **liveimg**
- **module**
- **%packages** (any necessary packages must be already available in the container image)
- **repo**
- **url** (if there is a need to fetch a **stage2** image for installation, for example, PXE installations, use **inst.stage2=** on the kernel command line instead of providing a url for **stage2** inside the Kickstart file)
- **zfcp**
- **zipl**

[illegible]

Kickstart commands not listed here are allowed to be used with the **ostreecontainer** command, however, they are not guaranteed to work as expected in the same way as with package-based installations.

Note

- The **ostreecontainer** option cannot be used with the **ostreesetup** command.

22.2.15. poweroff

The **poweroff** Kickstart command is optional. It shuts down and powers off the system after the installation has successfully completed. Normally during a manual installation, Anaconda displays a message and waits for the user to press a key before rebooting. Use this command only once. This command has no options.

Syntax

```
poweroff
```

Notes

- The **poweroff** option is equivalent to the **shutdown -P** command. For more details, see the *shutdown(8)* man page on your system.
- For other completion methods, see the **halt**, **reboot**, and **shutdown** Kickstart commands. The **halt** option is the default completion method if no other methods are explicitly specified in the Kickstart file.
- The **poweroff** command is highly dependent on the system hardware in use. Specifically, certain hardware components such as the BIOS, APM (advanced power management), and ACPI (advanced configuration and power interface) must be able to interact with the system kernel. Consult your hardware documentation for more information about your system’s APM/ACPI abilities.

22.2.16. reboot

The **reboot** Kickstart command is optional. It instructs the installation program to reboot after the installation is successfully completed. Normally, Kickstart displays a message and waits for the user to press a key before rebooting. Use this command only once.

Syntax

```
reboot OPTIONS
```

Options

- **--eject** - Attempt to eject the bootable media (DVD, USB, or other media) before rebooting.
- **--kexec** - Uses the **kexec** system call instead of performing a full reboot, which immediately loads the installed system into memory, bypassing the hardware initialization normally performed by the BIOS or firmware.

IMPORTANT

This option is deprecated and available as a Technology Preview only. For information about Red Hat scope of support for Technology Preview features, see the Technology Preview Features Support Scope document.

When **kexec** is used, device registers (which would normally be cleared during a full system reboot) might stay filled with data, which could potentially create issues for some device drivers.

Notes

- Use of the **reboot** option *might* result in an endless installation loop, depending on the installation media and method.
- The **reboot** option is equivalent to the **shutdown -r** command. For more details, see the *shutdown(8)* man page on your system.
- Specify **reboot** to automate installation fully when installing in command line mode on 64-bit IBM Z.
- For other completion methods, see the **halt**, **poweroff**, and **shutdown** Kickstart options. The **halt** option is the default completion method if no other methods are explicitly specified in the Kickstart file.

22.2.17. rhsm

The **rhsm** Kickstart command is optional. It instructs the installation program to register and install RHEL from the CDN. It can also connect the installed system to Red Hat Lightspeed. Use this command only once.

The **rhsm** Kickstart command eliminates the requirement of custom **%post** scripts when registering the system.

Options

- **--organization=** - Uses the organization ID to register and install RHEL from the CDN.
- **--activation-key=** - Uses the activation key to register and install RHEL from the CDN. Option can be used multiple times, once per activation key, as long as the activation keys used are registered to your subscription.
- **--connect-to-insights** - Connects the target system to Red Hat Lightspeed.
- **--proxy=** - Sets the HTTP proxy.
- **--server-hostname=** - Sets the Satellite instance hostname for registration.

IMPORTANT

- To switch the installation source repository to the CDN by using the **rhsm** Kickstart command, you must meet the following conditions:
  - On the kernel command line, you have used **inst.stage2=*<URL>*** to fetch the installation image but have not specified an installation source using **inst.repo=**.
  - In the Kickstart file, you have not specified an installation source by using the **url**, **cdrom**, **harddrive**, **liveimg**, **nfs** and **ostree** setup commands.
- An installation source URL specified using a boot option or included in a Kickstart file takes precedence over the CDN, even if the Kickstart file contains the **rhsm** command with valid credentials. The system is registered, but it is installed from the URL installation source. This ensures that earlier installation processes operate as normal.

22.2.18. shutdown

The **shutdown** Kickstart command is optional. It shuts down the system after the installation has successfully completed. Use this command only once. This command has no options.

Syntax

```
shutdown
```

Notes

- The **shutdown** Kickstart option is equivalent to the **shutdown** command. For more details, see the *shutdown(8)* man page on your system.
- For other completion methods, see the **halt**, **poweroff**, and **reboot** Kickstart options. The **halt** option is the default completion method if no other methods are explicitly specified in the Kickstart file.

22.2.19. sshpw

The **sshpw** Kickstart command is optional. During the installation, you can interact with the installation program and monitor its progress over an **SSH** connection. Use the **sshpw** command to create temporary accounts through which to log on. Each instance of the command creates a separate account that exists only in the installation environment. These accounts are not transferred to the installed system.

Syntax

```
sshpw --username=name [OPTIONS] password
```

Mandatory options

- **--username**=*name* - Provides the name of the user. This option is required.
- ***password*** - The password to use for the user. This argument is required.

Optional options

- **--iscrypted** - If this option is present, the password argument is assumed to already be encrypted. This option is mutually exclusive with **--plaintext**. To create an encrypted password, you can use Python:

```
$ python3 -c 'import crypt,getpass;pw=getpass.getpass();print(crypt.crypt(pw) if
(pw==getpass.getpass("Confirm: ")) else exit())'
```

This generates a sha512 crypt-compatible hash of your password by using a random salt.

- **--plaintext** - If this option is present, the password argument is assumed to be in plain text. This option is mutually exclusive with **--iscrypted**
- **--lock** - If this option is present, this account is locked by default. This means that the user will not be able to log in from the console.
- **--sshkey** - If this option is present, then the *<password>* string is interpreted as an ssh key value.

Notes

- By default, the **ssh** server is not started during the installation. To make **ssh** available during the installation, boot the system with the kernel boot option **inst.sshd**.
- If you want to disable root **ssh** access, while allowing another user **ssh** access, use the following:

  ```
  sshpw --username=example_username example_password --plaintext
  sshpw --username=root example_password --lock
  ```

- To simply disable root **ssh** access, use the following:

  ```
  sshpw --username=root example_password --lock
  ```

22.2.20. text

The **text** Kickstart command is optional. It performs the Kickstart installation in text mode. Kickstart installations are performed in graphical mode by default. Use this command only once.

Syntax

```
text [--non-interactive]
```

Options

- **--non-interactive** - Performs the installation in a completely non-interactive mode. This mode will terminate the installation when user interaction is required.

Note

- For a fully automatic installation, you must either specify one of the available modes (**graphical**, **text**, or **cmdline**) in the Kickstart file, or you must use the **console=** boot option.

If no mode is specified, the system will use graphical mode if possible, or prompt you to choose from RDP and text mode.

22.2.21. url

The **url** Kickstart command is optional. It is used to install from an installation tree image on a remote server by using the FTP, HTTP, or HTTPS protocol. You can only specify one URL. Use this command only once.

You must specify one of the **--url**, **--metalink** or **--mirrorlist** options.

Syntax

```
url --url=FROM [OPTIONS]
```

Options

- **--url=*FROM*** - Specifies the **HTTP**, **HTTPS**, **FTP**, or **file** location to install from.
- **--mirrorlist=** - Specifies the mirror URL to install from.
- **--proxy=** - Specifies an **HTTP**, **HTTPS**, or **FTP** proxy to use during the installation.
- **--noverifyssl** - Disables SSL verification when connecting to an **HTTPS** server.
- **--metalink=*URL*** - Specifies the metalink URL to install from. Variable substitution is done for **$releasever** and **$basearch** in the *URL*.

Examples

- To install from a FTP server:

  ```
  url --url=ftp://username:password@server/path
  ```

- To install from a HTTP server:

  ```
  url --url=http://server/path
  ```

Note

- To actually run the installation, you must specify one of **cdrom**, **harddrive**, **hmc**, **nfs**, **liveimg**, **ostreesetup**, **rhsm**, or **url** unless the **inst.repo** option is specified on the kernel command line.

22.2.22. hmc

The hmc kickstart command is optional. Use it to install from an installation medium by using SE/HMC on IBM Z. This command does not have any options.

Syntax

```
hmc
```

22.2.23. %include

The **%include** Kickstart command is optional. Use the **%include** command to include the contents of another file in the Kickstart file as if the contents were at the location of the **%include** command in the Kickstart file.

This inclusion is evaluated only after the **%pre** script sections and can thus be used to include files generated by scripts in the **%pre** sections. To include files before evaluation of **%pre** sections, use the **%ksappend** command.

Syntax

```
%include path/to/file
```

22.2.24. %ksappend

The **%ksappend** Kickstart command is optional.

Use the **%ksappend** command to include the contents of another file in the Kickstart file as if the contents were at the location of the **%ksappend** command in the Kickstart file.

This inclusion is evaluated before the **%pre** script sections, unlike inclusion with the **%include** command.

Syntax

```
%ksappend path/to/file
```

22.3. KICKSTART COMMANDS FOR SYSTEM CONFIGURATION

The Kickstart commands in this list configure further details on the resulting system such as users, repositories, or services.

22.3.1. authselect

The **authselect** Kickstart command is optional. It sets up the authentication options for the system using the **authselect** command, which can also be run on the command line after the installation finishes. Use this command only once.

Syntax

```
authselect [OPTIONS]
```

Notes

- This command passes all options to the **authselect** command. Refer to the *authselect(8)* manual page and the **authselect --help** command for more details.

- Passwords are shadowed by default.
- When using OpenLDAP with the **SSL** protocol for security, ensure that the **SSLv2** and **SSLv3** protocols are disabled in the server configuration. This is due to the POODLE SSL vulnerability (CVE-2014-3566). For more information, see the Red Hat Knowledgebase solution Resolution for POODLE SSLv3.0 vulnerability.

22.3.2. firewall

The **firewall** Kickstart command is optional. It specifies the firewall configuration for the installed system.

Syntax

```
firewall --enabled|--disabled [incoming] [OPTIONS]
```

Mandatory options

- **--enabled** or **--enable** - Reject incoming connections that are not in response to outbound requests, such as DNS replies or DHCP requests. If access to services running on this machine is needed, you can choose to allow specific services through the firewall.
- **--disabled** or **--disable** - Do not configure any iptables rules.

Optional options

- **--trust** - Listing a device here, such as **em1**, allows all traffic coming to and from that device to go through the firewall. To list more than one device, use the option more times, such as **--trust em1 --trust em2**. Do not use a comma-separated format such as **--trust em1, em2**.
- **--remove-service** - Close ports for comma-separated list of services.
- *incoming* - Replace with one or more of the following to allow the specified services through the firewall.
  - **--ssh**
  - **--smtp**
  - **--http**
  - **--ftp**
- **--port=** - You can specify that ports be allowed through the firewall using the port:protocol format. For example, to allow IMAP access through your firewall, specify **imap:tcp**. Numeric ports can also be specified explicitly; for example, to allow UDP packets on port 1234 through, specify **1234:udp**. To specify multiple ports, separate them by commas.
- **--service=** - This option provides a higher-level way to allow services through the firewall. Some services (such as **cups**, **avahi**, and so on) require multiple ports to be open or other special configuration in order for the service to work. You can specify each individual port with the **--port** option, or specify **--service=** and open them all at once.

Valid options are anything recognized by the **firewall-offline-cmd** program in the **firewalld** package. If the **firewalld** service is running, **firewall-cmd --get-services** provides a list of known service names.

- **--use-system-defaults** - Do not configure the firewall at all. This option instructs anaconda to do nothing and allows the system to rely on the defaults that were provided with the package or ostree. If this option is used with other options then all other options will be ignored.

22.3.3. group

The **group** Kickstart command is optional. It creates a new user group on the system.

Syntax

```
group --name=name [--gid=gid]
```

Mandatory options

- **--name=** - Provides the name of the group.

Optional options

- **--gid=** - The group’s GID. If not provided, defaults to the next available non-system GID.

Notes

- If a group with the given name or GID already exists, this command fails.
- The **user** command can be used to create a new group for the newly created user.

22.3.4. keyboard

The **keyboard** Kickstart command is required. It sets one or more available keyboard layouts for the system. Use this command only once.

Syntax

```
keyboard --vckeymap|--xlayouts OPTIONS
```

Options

- **--vckeymap=** - Specify a **VConsole** keymap which should be used. Valid names correspond to the list of files in the **/usr/lib/kbd/keymaps/xkb/** directory, without the **.map.gz** extension.
- **--xlayouts=** - Specify a list of X layouts that should be used as a comma-separated list without spaces. Accepts values in the same format as **setxkbmap(1)**, either in the ***layout*** format (such as **cz**), or in the ***layout* (*variant*)** format (such as **cz (qwerty)**).
All available layouts can be viewed on the **xkeyboard-config(7)** man page under **Layouts**.

- **--switch=** - Specify a list of layout-switching options (shortcuts for switching between multiple keyboard layouts). Multiple options must be separated by commas without spaces. Accepts values in the same format as **setxkbmap(1)**.
  Available switching options can be viewed on the **xkeyboard-config(7)** man page under **Options**.

Example

The following example sets up two keyboard layouts (**English (US)** and **Czech (qwerty)**) using the **--xlayouts=** option, and allows to switch between them using **Alt**+**Shift**:

```
keyboard --xlayouts=us,'cz (qwerty)' --switch=grp:alt_shift_toggle
```

Note

- Either the **--vckeymap=** or the **--xlayouts=** option must be used.

22.3.5. lang

The **lang** Kickstart command is required. It sets the language to use during installation and the default language to use on the installed system. Use this command only once.

Syntax

```
lang language [--addsupport=language,...]
```

Mandatory options

- ***language*** - Install support for this language and set it as system default.

Optional options

- **--addsupport=** - Add support for additional languages. Takes the form of comma-separated list without spaces. For example:

```
lang en_US --addsupport=cs_CZ,de_DE,en_UK
```

Notes

- The **locale -a | grep _** or **localectl list-locales | grep _** commands return a list of supported locales.
- Certain languages (for example, Chinese, Japanese, Korean, and Indic languages) are not supported during text-mode installation. If you specify one of these languages with the **lang** command, the installation process continues in English, but the installed system uses your selection as its default language.

Example

To set the language to English, the Kickstart file should contain the following line:

```
lang en_US
```

22.3.6. module

The **module** Kickstart command is optional. Use this command to enable a package module stream within the kickstart script.

Syntax

```
module --name=NAME [--stream=STREAM]
```

Mandatory options

- **--name=** : Specifies the name of the module to enable. Replace *NAME* with the actual name.

Optional options

- **--stream=** : Specifies the name of the module stream to enable. Replace *STREAM* with the actual name.

You do not need to specify this option for modules with a default stream defined. For modules without a default stream, this option is mandatory and leaving it out results in an error. Enabling a module multiple times with different streams is not possible.

Notes

- Using a combination of this command and the **%packages** section allows you to install packages provided by the enabled module and stream combination, without specifying the module and stream explicitly. Modules must be enabled before package installation. After enabling a module with the **module** command, you can install the packages enabled by this module by listing them in the **%packages** section.
- A single **module** command can enable only a single module and stream combination. To enable multiple modules, use multiple **module** commands. Enabling a module multiple times with different streams is not possible.
- In Red Hat Enterprise Linux 10, modules are present only in the AppStream repository. To list available modules, use the **dnf module list** command on an installed Red Hat Enterprise Linux Red Hat Enterprise Linux system with a valid subscription.

Additional resources

- Managing Software with the DNF Tool

22.3.7. repo

The **repo** Kickstart command is optional. It configures additional dnf repositories that can be used as sources for package installation. You can add multiple **repo** lines.

Syntax

```
repo --name=repoid [--baseurl=url|--mirrorlist=url|--metalink=url] [OPTIONS]
```

Mandatory options

- **--name=** - The repository id. This option is required. If a repository has a name which conflicts with another previously added repository, it is ignored. Because the installation program uses a list of preset repositories, this means that you cannot add repositories with the same names as the preset ones.

URL options

These options are mutually exclusive and optional. The variables that can be used in dnf repository configuration files are not supported here. You can use the strings **$releasever** and **$basearch** which are replaced by the respective values in the URL.

- **--baseurl=** - The URL to the repository.
- **--mirrorlist=** - The URL pointing at a list of mirrors for the repository.
- **--metalink=** - The URL with metalink for the repository.

Optional options

- **--install** - Save the provided repository configuration on the installed system in the **/etc/yum.repos.d/** directory. Without using this option, a repository configured in a Kickstart file will only be available during the installation process, not on the installed system.
- **--cost=** - An integer value to assign a cost to this repository. If multiple repositories provide the same packages, this number is used to prioritize which repository will be used before another. Repositories with a lower cost take priority over repositories with higher cost.
- **--excludepkgs=** - A comma-separated list of package names that must *not* be pulled from this repository. This is useful if multiple repositories provide the same package and you want to make sure it comes from a particular repository. Both full package names (such as **publican**) and globs (such as **gnome-***) are accepted.
- **--includepkgs=** - A comma-separated list of package names and globs that are allowed to be pulled from this repository. Any other packages provided by the repository will be ignored. This is useful if you want to install just a single package or set of packages from a repository while excluding all other packages the repository provides.
- **--proxy=[*protocol*://][*username*[:*password*]@]*host*[:*port*]** - Specify an HTTP/HTTPS/FTP proxy to use just for this repository. This setting does not affect any other repositories, nor how the **install.img** is fetched on HTTP installations.
- **--noverifyssl** - Disable SSL verification when connecting to an **HTTPS** server.

Note

- Repositories used for installation must be stable. The installation can fail if a repository is modified before the installation concludes.

22.3.8. rootpw

The **rootpw** Kickstart command is required. It sets the system's root password to the *password* argument. Use this command only once.

Syntax

```
rootpw [--iscrypted|--plaintext] [--lock] password
```

Mandatory options

- *password* - Password specification. Either plain text or encrypted string. See **--iscrypted** and **--plaintext** below.

Optional options

- **--iscrypted** - If this option is present, the password argument is assumed to already be encrypted. This option is mutually exclusive with **--plaintext**. To create an encrypted password, you can use python:

```
$ python -c 'import crypt,getpass;pw=getpass.getpass();print(crypt.crypt(pw) if
(pw==getpass.getpass("Confirm: ")) else exit())'
```

This generates a sha512 crypt-compatible hash of your password by using a random salt.

- **--plaintext** - If this option is present, the password argument is assumed to be in plain text. This option is mutually exclusive with **--iscrypted**.
- **--lock** - If this option is present, the root account is locked by default. This means that the root user will not be able to log in from the console. This option will also disable the **Root Password** screens in both the graphical and text-based manual installation.
- **--allow-ssh** - If this option is present, the root user can login to the system by using SSH with a password.

22.3.9. selinux

The **selinux** Kickstart command is optional. It sets the state of SELinux on the installed system. The default SELinux policy is **enforcing**. Use this command only once.

Syntax

```
selinux [--disabled|--enforcing|--permissive]
```

Options

- **--enforcing** - Enables SELinux with the default targeted policy being **enforcing**.
- **--permissive** - Outputs warnings based on the SELinux policy, but does not actually enforce the policy.
- **--disabled** - Disables SELinux completely on the system.

Additional resources

- Using SElinux

22.3.10. services

The **services** Kickstart command is optional. It modifies the default set of services that will run under the default systemd target. The list of disabled services is processed before the list of enabled services. Therefore, if a service appears on both lists, it will be enabled.

Syntax

```
services [--disabled=list] [--enabled=list]
```

Options

- **--disabled=** - Disable the services given in the comma separated list.
- **--enabled=** - Enable the services given in the comma separated list.

Notes

- When using the **services** element to enable **systemd** services, ensure you include packages containing the specified service file in the **%packages** section.
- Multiple services should be included separated by comma, without any spaces. For example, to disable four services, enter:

  ```
  services --disabled=auditd,cups,smartd,nfslock
  ```

  If you include any spaces, Kickstart enables or disables only the services up to the first space. For example:

  ```
  services --disabled=auditd, cups, smartd, nfslock
  ```

  That disables only the **auditd** service. To disable all four services, this entry must include no spaces.

22.3.11. skipx

The **skipx** Kickstart command is optional. If present, the system is configured to boot in text mode. Use this command only once. This command has no options.

Syntax

```
skipx
```

22.3.12. sshkey

The **sshkey** Kickstart command is optional. It adds a SSH public key to the **authorized_keys** file of the specified user on the installed system.

Syntax

```
sshkey --username=user "ssh_key"
```

Mandatory options

- **--username=** - The user for which the key will be installed.
- ***ssh_key*** - The complete SSH public key. It must be wrapped with quotes.

22.3.13. syspurpose

The **syspurpose** Kickstart command is optional. Use it to set the system purpose which describes how the system will be used after installation. This information helps apply the correct subscription entitlement to the system. Use this command only once.

NOTE

Red Hat Enterprise Linux 9.0 and later enables you to manage and display system purpose attributes with a single module by making the **role**, **service-level**, **usage**, and **addons** subcommands available under one **subscription-manager syspurpose** module. Previously, system administrators used one of four standalone **syspurpose** commands to manage each attribute. This standalone **syspurpose** command is deprecated starting with RHEL 9.0 and is planned to be removed in post RHEL 9. Red Hat will provide bug fixes and support for this feature during the current release lifecycle, but this feature will no longer receive enhancements. Starting with RHEL 9, the single **subscription-manager syspurpose** command and its associated subcommands is the only way to use system purpose. Note that these changes only apply to the command line tools for system purpose configuration used on installed systems and do not influence the functionality of the syspurpose kickstart command.

Syntax

```
syspurpose [OPTIONS]
```

Options

- **--role=** - Set the intended system role. Available values are:
  - Red Hat Enterprise Linux Server
  - Red Hat Enterprise Linux Workstation
  - Red Hat Enterprise Linux Compute Node
- **--sla=** - Set the Service Level Agreement. Available values are:
  - Premium
  - Standard
  - Self-Support
- **--usage=** - The intended usage of the system. Available values are:
  - Production
  - Disaster Recovery
  - Development/Test

Notes

- Enter the values with spaces and enclose them in double quotes:

```
syspurpose --role="Red Hat Enterprise Linux Server"
```

- While it is strongly recommended that you configure System Purpose, it is an optional feature of the Red Hat Enterprise Linux installation program.
- Additional values might be used based on how subscriptions are configured for the given account

22.3.14. timezone

The **timezone** Kickstart command is required. It sets the system time zone. Use this command only once.

Syntax

```
timezone timezone [OPTIONS]
```

Mandatory options

- *timezone* - the time zone to set for the system.

Optional options

- **--utc** - If present, the system assumes the hardware clock is set to UTC (Greenwich Mean) time.

Note

In Red Hat Enterprise Linux 10, time zone names are validated using the **pytz.all_timezones** list, provided by the **pytz** package. In previous releases, the names were validated against **pytz.common_timezones**, which is a subset of the currently used list. Note that the graphical and text mode interfaces still use the more restricted **pytz.common_timezones** list; you must use a Kickstart file to use additional time zone definitions.

22.3.15. timesource

The **timesource** kickstart command is optional. Use it to set NTP, NTS servers, and pools that provide time data, as well as control whether NTP services are enabled or disabled on the system.

Syntax

```
timesource [--ntp-server NTP_SERVER | --ntp-pool NTP_POOL | --ntp-disable] [--nts]
```

Mandatory options

It is mandatory to specify one of the following options when you use the **timesource** command:

- **--ntp-server** - adds one NTP server as a time source. This option can be added only once to a single command in order to add a one NTP time source server. To add multiple sources, add multiple timesource commands each with a single --ntp-server or --ntp-pool option each time. For example, to add multiple sources for *Europe* timezone

```
timezone Europe
timesource --ntp-server 0.rhel.pool.ntp.org
timesource --ntp-server 1.rhel.pool.ntp.org
timesource --ntp-server 2.rhel.pool.ntp.org
```

- **--ntp-pool** - adds a NTP server pool as a time source. This option can be added only once to add a single NTP time source pool. Repeat the timesource command to add multiple sources.

- **--ntp-disable** - disables NTP time sources for the installed system.

Optional options

- **--nts** - the server or pool added with this command uses the NTS protocol. Note that this option can be added even with --ntp-disable, but it has no effect.

22.3.16. user

The **user** Kickstart command is optional. It creates a new user on the system.

Syntax

```
user --name=username [OPTIONS]
```

Mandatory options

- **--name=** - Provides the name of the user. This option is required.

Optional options

- **--gecos=** - Provides the GECOS information for the user. This is a string of various system-specific fields separated by a comma. It is frequently used to specify the user’s full name, office number, and so on. See the **passwd(5)** man page for more details.

- **--groups=** - In addition to the default group, a comma separated list of group names the user should belong to. The groups must exist before the user account is created. See the **group** command.

- **--homedir=** - The home directory for the user. If not provided, this defaults to **/home/*username***.

- **--lock** - If this option is present, this account is locked by default. This means that the user will not be able to log in from the console. This option will also disable the **Create User** screens in both the graphical and text-based manual installation.

- **--password=** - The new user’s password. If not provided, the account will be locked by default.

- **--iscrypted** - If this option is present, the password argument is assumed to already be encrypted. This option is mutually exclusive with **--plaintext**. To create an encrypted password, you can use python:

```
$ python -c 'import crypt,getpass;pw=getpass.getpass();print(crypt.crypt(pw) if
(pw==getpass.getpass("Confirm: ")) else exit())'
```

This generates a sha512 crypt-compatible hash of your password by using a random salt.

- **--plaintext** - If this option is present, the password argument is assumed to be in plain text. This option is mutually exclusive with **--iscrypted**
- **--shell=** - The user's login shell. If not provided, the system default is used.
- **--uid=** - The user's UID (User ID). If not provided, this defaults to the next available non-system UID.
- **--gid=** - The GID (Group ID) to be used for the user's group. If not provided, this defaults to the next available non-system group ID.

Notes

- Consider using the **--uid** and **--gid** options to set IDs of regular users and their default groups at range starting at **5000** instead of **1000**. That is because the range reserved for system users and groups, **0**-**999**, might increase in the future and thus overlap with IDs of regular users.
- Files and directories are created with various permissions, dictated by the application used to create the file or directory. For example, the **mkdir** command creates directories with all permissions enabled. However, applications are prevented from granting certain permissions to newly created files, as specified by the **user file-creation mask** setting.
  The **user file-creation mask** can be controlled with the **umask** command. The default setting of the **user file-creation mask** for new users is defined by the **UMASK** variable in the **/etc/login.defs** configuration file on the installed system. If unset, it defaults to **022**. This means that by default when an application creates a file, it is prevented from granting write permission to users other than the owner of the file. However, this can be overridden by other settings or scripts.

22.3.17. xconfig

The **xconfig** Kickstart command is optional. If used with the **--startxonboot** option it configures the installed system to start in graphical mode. Use this command only once.

Syntax

```
xconfig [--startxonboot]
```

Options

- **--startxonboot** - Use a graphical login on the installed system.

Note

- Because Red Hat Enterprise Linux 10 does not include the KDE Desktop Environment, do not use the **--defaultdesktop=** documented upstream.

22.4. KICKSTART COMMANDS FOR NETWORK CONFIGURATION

The Kickstart commands in this list let you configure networking on the system.

22.4.1. network

Use the optional **network** Kickstart command to configure network information for the target system and activate the network devices in the installation environment. The device specified in the first **network** command is activated automatically. You can also explicitly require a device to be activated by using the **--activate** option.

> **WARNING**
>
> Re-configuration of already active network devices that are in use by the running installer may lead to an installation failure or freeze. In such a case, avoid re-configuration of network devices used to access the installer runtime image (*stage2*) over NFS.

Syntax

```
network OPTIONS
```

Options

- **--activate** - activate this device in the installation environment.

If you use the **--activate** option on a device that has already been activated (for example, an interface you configured with boot options so that the system could retrieve the Kickstart file) the device is reactivated to use the details specified in the Kickstart file.

Use the **--nodefroute** option to prevent the device from using the default route.

- **--no-activate** - do not activate this device in the installation environment.
  By default, Anaconda activates the first network device in the Kickstart file regardless of the **--activate** option. You can disable the default setting by using the **--no-activate** option.

- **--bootproto=** - One of **dhcp**, **bootp**, **ibft**, or **static**. The default option is **dhcp**; the **dhcp** and **bootp** options are treated the same. To disable **ipv4** configuration of the device, use **--noipv4** option.

  > **NOTE**
  >
  > This option configures the ipv4 configuration of the device. For ipv6 configuration use **--ipv6** and **--ipv6gateway** options.

  The DHCP method uses a DHCP server system to obtain its networking configuration. The BOOTP method is similar, requiring a BOOTP server to supply the networking configuration. To direct a system to use DHCP:

  –

```
network --bootproto=dhcp
```

To direct a machine to use BOOTP to obtain its networking configuration, use the following line in the Kickstart file:

```
network --bootproto=bootp
```

To direct a machine to use the configuration specified in iBFT, use:

```
network --bootproto=ibft
```

The **static** method requires that you specify at least the IP address and netmask in the Kickstart file. This information is static and is used during and after the installation.

All static networking configuration information must be specified on *one* line; you cannot wrap lines by using a backslash (\) as you can on a command line.

```
network --bootproto=static --ip=10.0.2.15 --netmask=255.255.255.0 --
gateway=10.0.2.254 --nameserver=10.0.2.1
```

You can also configure multiple nameservers at the same time. To do so, use the **--nameserver=** option once, and specify each of their IP addresses, separated by commas:

```
network --bootproto=static --ip=10.0.2.15 --netmask=255.255.255.0 --
gateway=10.0.2.254 --nameserver=192.168.2.1,192.168.3.1
```

- **--device=** - specifies the device to be configured (and eventually activated in Anaconda) with the **network** command.
  If the **--device=** option is missing on the *first* use of the **network** command, the value of the **inst.ks.device=** Anaconda boot option is used, if available. This is considered deprecated behavior; in most cases, you should always specify a **--device=** for every **network** command.

  The behavior of any subsequent **network** command in the same Kickstart file is unspecified if its **--device=** option is missing. Verify you specify this option for any **network** command beyond the first.

  You can specify a device to be activated in any of the following ways:

  - the device name of the interface, for example, **em1**
  - the MAC address of the interface, for example, **01:23:45:67:89:ab**
  - the keyword **link**, which specifies the first interface with its link in the **up** state
  - the keyword **bootif**, which uses the MAC address that pxelinux set in the **BOOTIF** variable. Set **IPAPPEND 2** in your **pxelinux.cfg** file to have pxelinux set the **BOOTIF** variable.

  For example:

  ```
  network --bootproto=dhcp --device=em1
  ```

- **--ipv4-dns-search**/**--ipv6-dns-search** - Set the DNS search domains manually. You must use these options together with **--device** options and mirror their respective NetworkManager properties, for example:

  ```
  network --device ens3 --ipv4-dns-search domain1.example.com,domain2.example.com
  ```

- **--ipv4-ignore-auto-dns**/**--ipv6-ignore-auto-dns** - Set to ignore the DNS settings from DHCP. You must use these options together with **--device** options and these options do not require any arguments.

- **--ip=** - IP address of the device.

- **--ipv6=** - IPv6 address of the device, in the form of *address*[/*prefix length*] - for example, **3ffe:ffff:0:1::1/128**. If the *prefix* is omitted, **64** is used. You can also use **auto** for automatic configuration, or **dhcp** for DHCPv6-only configuration (no router advertisements).

- **--gateway=** - Default gateway as a single IPv4 address.

- **--ipv6gateway=** - Default gateway as a single IPv6 address.

- **--nodefroute** - Prevents the interface being set as the default route. Use this option when you activate additional devices with the **--activate=** option, for example, a NIC on a separate subnet for an iSCSI target.

- **--nameserver=** - DNS name server, as an IP address. To specify more than one name server, use this option once, and separate each IP address with a comma.

- **--netmask=** - Network mask for the installed system.

- **--hostname=** - Used to configure the target system's host name. The host name can either be a fully qualified domain name (FQDN) in the format **hostname.domainname**, or a short host name without the domain. Many networks have a Dynamic Host Configuration Protocol (DHCP) service that automatically supplies connected systems with a domain name. To allow the DHCP service to assign the domain name to this machine, specify only the short host name.
  When using static IP and host name configuration, it depends on the planned system use case whether to use a short name or FQDN. Red Hat Identity Management configures FQDN during provisioning but some 3rd party software products may require a short name. In either case, to ensure availability of both forms in all situations, add an entry for the host in **/etc/hosts** in the format **IP FQDN short-alias**.

  Host names can only contain alphanumeric characters and **-** or **.**. Host name should be equal to or less than 64 characters. Host names cannot start or end with **-** and **.**. To be compliant with DNS, each part of a FQDN should be equal to or less than 63 characters and the FQDN total length, including dots, should not exceed 255 characters.

  If you only want to configure the target system's host name, use the **--hostname** option in the **network** command and do not include any other option.

  If you provide additional options when configuring the host name, the **network** command configures a device by using the options specified. If you do not specify which device to configure by using the **--device** option, the default **--device link** value is used. Additionally, if you do not specify the protocol by using the **--bootproto** option, the device is configured to use DHCP by default.

- **--ethtool=** - Specifies additional low-level settings for the network device which will be passed to the ethtool program.
- **--onboot=** - Whether or not to enable the device at boot time.
- **--dhcpclass=** - The DHCP class.
- **--mtu=** - The MTU of the device.
- **--noipv4** - Disable IPv4 on this device.
- **--noipv6** - Disable IPv6 on this device.
- **--bondslaves=** - When this option is used, the bond device specified by the **--device=** option is created using secondary devices defined in the **--bondslaves=** option. For example:

  ```
  network --device=bond0 --bondslaves=em1,em2
  ```

  The above command creates a bond device named **bond0** by using the **em1** and **em2** interfaces as its secondary devices.

- **--bondopts=** - a list of optional parameters for a bonded interface, which is specified by using the **--bondslaves=** and **--device=** options. Options in this list must be separated by commas (",") or semicolons (";"). If an option itself contains a comma, use a semicolon to separate the options. For example:

  ```
  network --bondopts=mode=active-backup,balance-rr;primary=eth1
  ```

  IMPORTANT

  The **--bondopts=mode=** parameter only supports full mode names such as **balance-rr** or **broadcast**, not their numerical representations such as **0** or **3**. For the list of available and supported modes, see the Configuring and Managing Networking Guide.

- **--vlanid=** - Specifies virtual LAN (VLAN) ID number (802.1q tag) for the device created by using the device specified in **--device=** as a parent. For example, **network --device=em1 --vlanid=171** creates a virtual LAN device **em1.171**.
- **--interfacename=** - Specify a custom interface name for a virtual LAN device. This option should be used when the default name generated by the **--vlanid=** option is not desirable. This option must be used along with **--vlanid=**. For example:

  ```
  network --device=em1 --vlanid=171 --interfacename=vlan171
  ```

  The above command creates a virtual LAN interface named **vlan171** on the **em1** device with an ID of **171**.

  The interface name can be arbitrary (for example, **my-vlan**), but in specific cases, the following conventions must be followed:

- If the name contains a dot (**.**), it must take the form of ***NAME.ID***. The *NAME* is arbitrary, but the *ID* must be the VLAN ID. For example: **em1.171** or **my-vlan.171**.
- Names starting with **vlan** must take the form of **vlan*ID*** - for example, **vlan171**.

- **--bridgeslaves=** - When this option is used, the network bridge with device name specified by using the **--device=** option will be created and devices defined in the **--bridgeslaves=** option will be added to the bridge. For example:

  ```
  network --device=bridge0 --bridgeslaves=em1
  ```

- **--bridgeopts=** - An optional comma-separated list of parameters for the bridged interface. Available values are **stp**, **priority**, **forward-delay**, **hello-time**, **max-age**, and **ageing-time**. For information about these parameters, see the *bridge setting* table in the **nm-settings(5)** man page or at Network Configuration Setting Specification.
  Also see the Configuring and managing networking document for general information about network bridging.
- **--bindto=mac** - Bind the device configuration file on the installed system to the device MAC address (**HWADDR**) instead of the default binding to the interface name (**DEVICE**). This option is independent of the **--device=** option - **--bindto=mac** will be applied even if the same **network** command also specifies a device name, **link**, or **bootif**.

Notes

- The **ethN** device names such as **eth0** are no longer available in Red Hat Enterprise Linux due to changes in the naming scheme. For more information about the device naming scheme, see the upstream document Predictable Network Interface Names .
- If you used a Kickstart option or a boot option to specify an installation repository on a network, but no network is available at the start of the installation, the installation program displays the **Network Configuration** window to set up a network connection prior to displaying the **Installation Summary** window. For more details, see Configuring network and host name options.

22.4.2. realm

The **realm** Kickstart command is optional. Use it to join an Active Directory or IPA domain. For more information about this command, see the **join** section of the **realm(8)** man page on your system.

Syntax

```
realm join [OPTIONS] domain
```

Mandatory options

- ***domain*** - The domain to join.

Options

- **--computer-ou=OU=** - Provide the distinguished name of an organizational unit in order to create the computer account. The exact format of the distinguished name depends on the client software and membership software. The root DSE portion of the distinguished name can usually be left out.

- **--no-password** - Join automatically without a password.
- **--one-time-password=** - Join using a one-time password. This is not possible with all types of realm.
- **--client-software=** - Only join realms which can run this client software. Valid values include **sssd** and **winbind**. Not all realms support all values. By default, the client software is chosen automatically.
- **--server-software=** - Only join realms which can run this server software. Possible values include **active-directory** or **freeipa**.
- **--membership-software=** - Use this software when joining the realm. Valid values include **samba** and **adcli**. Not all realms support all values. By default, the membership software is chosen automatically.

22.5. KICKSTART COMMANDS FOR HANDLING STORAGE

The Kickstart commands in this section configure aspects of storage such as devices, disks, partitions, LVM, and filesystems.

IMPORTANT

The **sdX** (or **/dev/sdX**) format does not guarantee consistent device names across reboots, which can complicate the usage of some Kickstart commands. When a command requires a device node name, you can use any item from **/dev/disk** as an alternative. For example, instead of using the following device name:

```
part / --fstype=xfs --onpart=sda1
```

You can use an entry similar to one of the following:

```
part / --fstype=xfs --onpart=/dev/disk/by-path/pci-0000:00:05.0-scsi-0:0:0:0-part1
```

```
part / --fstype=xfs --onpart=/dev/disk/by-id/ata-ST3160815AS_6RA0C882-part1
```

By using this approach, the command always targets the same storage device. This is especially useful in large storage environments. To explore the available device names on the system, you can use the **ls -lR /dev/disk** command during the interactive installation. For more information about different ways to consistently refer to storage devices, see Persistent naming attributes.

22.5.1. ignoredisk

The **ignoredisk** Kickstart command is optional. It causes the installation program to ignore the specified disks.

This is useful if you use automatic partitioning and want to be sure that some disks are ignored. For example, without **ignoredisk**, attempting to deploy on a SAN-cluster the Kickstart would fail, as the installation program detects passive paths to the SAN that return no partition table. Use this command only once.

Syntax

```
ignoredisk --drives=drive1,drive2,... | --only-use=drive
```

Options

- **--drives=*driveN*,...** - Replace *driveN* with one of **sda**, **sdb**,..., **hda**,... and so on.
- **--only-use=*driveN*,...** - Specifies a list of disks for the installation program to use. All other disks are ignored. For example, to use disk **sda** during installation and ignore all other disks:

  ```
  ignoredisk --only-use=sda
  ```

  To include a multipath device that does not use LVM:

  ```
  ignoredisk --only-use=disk/by-id/dm-uuid-mpath-
  2416CD96995134CA5D787F00A5AA11017
  ```

  To include a multipath device that uses LVM:

  ```
  ignoredisk --only-use=disk/by-id/scsi-58095BEC5510947BE8C0360F604351918
  ```

  You must specify only one of the **--drives** or **--only-use**.

Notes

- To specify a multipath device, use the format **disk/by-id/scsi-WWID**, where WWID is the world-wide identifier for the device. For example, to specify a disk with WWID **58095BEC5510947BE8C0360F604351918**, use:

```
ignoredisk --only-use=disk/by-id/scsi-58095BEC5510947BE8C0360F604351918
```

This format is preferable for all multipath devices, but if errors arise, multipath devices that do not use logical volume manager (LVM) can also be specified by using the format **disk/by-id/dm-uuid-mpath-WWID**, where WWID is the world-wide identifier for the device. For example, to specify a disk with WWID **2416CD96995134CA5D787F00A5AA11017**, use:

```
ignoredisk --only-use=disk/by-id/dm-uuid-mpath-2416CD96995134CA5D787F00A5AA11017
```

Never specify multipath devices by device names like mpatha. Device names such as this are not specific to a particular disk. The disk named /dev/mpatha during installation might not be the one that you expect it to be. Therefore, the clearpart command could target the wrong disk.

- The **sdX** (or **/dev/sdX**) format does not guarantee consistent device names across reboots, which can complicate the usage of some Kickstart commands. When a command requires a device node name, you can use any item from **/dev/disk** as an alternative. For example, instead of using the following device name:

  ```
  ignoredisk --only-use=sda
  ```

  You can use an entry similar to one of the following:

  ```
  ignoredisk --only-use=/dev/disk/by-path/pci-0000:00:05.0-scsi-0:0:0:0
  ```

```
ignoredisk --only-use=/dev/disk/by-id/ata-ST3160815AS_6RA0C882
```

By using this approach, the command always targets the same storage device. This is especially useful in large storage environments. To explore the available device names on the system, you can use the **ls -lR /dev/disk** command during the interactive installation. For more information about different ways to consistently refer to storage devices, see Persistent naming attributes.

22.5.2. clearpart

The **clearpart** Kickstart command is optional. It removes partitions from the system, prior to creation of new partitions. By default, no partitions are removed. Use this command only once.

Syntax

```
clearpart OPTIONS
```

Options

- **--all** - Erases all partitions from the system.

This option will erase all disks which can be reached by the installation program, including any attached network storage. Use this option with caution.

You can prevent **clearpart** from wiping storage you want to preserve by using the **--drives=** option and specifying only the drives you want to clear, by attaching network storage later (for example, in the **%post** section of the Kickstart file), or by blocklisting the kernel modules used to access network storage.

- **--drives=** - Specifies which drives to clear partitions from. For example, the following clears all the partitions on the first two drives on the primary IDE controller:

  ```
  clearpart --drives=hda,hdb --all
  ```

  To clear a multipath device, use the format **disk/by-id/scsi-*WWID***, where *WWID* is the world-wide identifier for the device. For example, to clear a disk with WWID **58095BEC5510947BE8C0360F604351918**, use:

  ```
  clearpart --drives=disk/by-id/scsi-58095BEC5510947BE8C0360F604351918
  ```

  This format is preferable for all multipath devices, but if errors arise, multipath devices that do not use logical volume manager (LVM) can also be cleared using the format **disk/by-id/dm-uuid-mpath-*WWID***, where *WWID* is the world-wide identifier for the device. For example, to clear a disk with WWID **2416CD96995134CA5D787F00A5AA11017**, use:

  ```
  clearpart --drives=disk/by-id/dm-uuid-mpath-2416CD96995134CA5D787F00A5AA11017
  ```

  Never specify multipath devices by device names like **mpatha**. Device names such as this are not specific to a particular disk. The disk named **/dev/mpatha** during installation might not be the one that you expect it to be. Therefore, the **clearpart** command could target the wrong disk.

- **--initlabel** - Initializes a disk (or disks) by creating a default disk label for all disks in their respective architecture that have been designated for formatting (for example, gpt for x86). Because **--initlabel** can see all disks, it is important to ensure only those drives that are to be formatted are connected. Disks cleared by **clearpart** will have the label created even in case the **--initlabel** is not used.

  ```
  clearpart --initlabel --drives=names_of_disks
  ```

  For example:

  ```
  clearpart --initlabel --drives=dasda,dasdb,dasdc
  ```

- **--list=** - Specifies which partitions to clear. This option overrides the **--all** and **--linux** options if used. Can be used across different drives. For example:

  ```
  clearpart --list=sda2,sda3,sdb1
  ```

- **--disklabel=*LABEL*** - Set the default disklabel to use. Only disklabels supported for the platform will be accepted. For example, on the 64-bit Intel and AMD architectures, the **msdos** and **gpt** disklabels are accepted, but **dasd** is not accepted.
- **--linux** - Erases all Linux partitions.
- **--none** (default) - Do not remove any partitions.
- **--cdl** - Reformat any LDL DASDs to CDL format.

Notes

- On the **x86_64** systems running in Legacy BIOS mode, the GPT disklabel is the default. To ensure these systems can boot properly with GPT, add a dedicated BIOS boot partition. For automated installations, consider using the **reqpart** kickstart command to create this BIOS boot partition when required automatically. Optionally, use the **part** kickstart command, for example: **part biosboot --size=1**. If you prefer to stick with the older MBR partitioning scheme instead of GPT, you have two options:
  - During a kickstart installation, use the **--disklabel msdos** option with the **clearpart** command.
  - Alternatively, add the **inst.disklabel=mbr** boot option at the installer's boot prompt.
- To specify a multipath device, use the format **disk/by-id/scsi-WWID**, where WWID is the world-wide identifier for the device. For example, to specify a disk with WWID **58095BEC5510947BE8C0360F604351918**, use:

  ```
  ignoredisk --only-use=disk/by-id/scsi-58095BEC5510947BE8C0360F604351918
  ```

  This format is preferable for all multipath devices, but if errors arise, multipath devices that do not use logical volume manager (LVM) can also be specified by using the format **disk/by-id/dm-uuid-mpath-WWID**, where WWID is the world-wide identifier for the device. For example, to specify a disk with WWID **2416CD96995134CA5D787F00A5AA11017**, use:

```
ignoredisk --only-use=disk/by-id/dm-uuid-mpath-
2416CD96995134CA5D787F00A5AA11017
```

Never specify multipath devices by device names like mpatha. Device names such as this are not specific to a particular disk. The disk named /dev/mpatha during installation might not be the one that you expect it to be. Therefore, the **ignoredisk** command could target the wrong disk.

- The **sdX** (or **/dev/sdX**) format does not guarantee consistent device names across reboots, which can complicate the usage of some Kickstart commands. When a command requires a device node name, you can use any item from **/dev/disk** as an alternative. For example, instead of using the following device name:

  ```
  clearpart --drives=sda
  ```

  You can use an entry similar to one of the following:

  ```
  clearpart --drives=/dev/disk/by-path/pci-0000:00:05.0-scsi-0:0:0:0
  clearpart --drives=/dev/disk/by-id/ata-ST3160815AS_6RA0C882
  ```

  By using this approach, the command always targets the same storage device. This is especially useful in large storage environments. To explore the available device names on the system, you can use the **ls -lR /dev/disk** command during the interactive installation. For more information about different ways to consistently refer to storage devices, see Persistent naming attributes.

- If the **clearpart** command is used, then the **part --onpart** command cannot be used on a logical partition.

22.5.3. zerombr

The **zerombr** Kickstart command is optional. The **zerombr** initializes any invalid partition tables that are found on disks and destroys all of the contents of disks with invalid partition tables.

This command is required when performing an installation on an 64-bit IBM Z system with unformatted Direct Access Storage Device (DASD) disks, otherwise the unformatted disks are not formatted and used during the installation. Use this command only once. This command has no options.

Syntax

```
zerombr
```

Notes

- On 64-bit IBM Z, if **zerombr** is specified, any Direct Access Storage Device (DASD) visible to the installation program which is not already low-level formatted is automatically low-level formatted with dasdfmt. The command also prevents user choice during interactive installations.

- If **zerombr** is not specified and there is at least one unformatted DASD visible to the installation program, a non-interactive Kickstart installation exits unsuccessfully.

- If **zerombr** is not specified and there is at least one unformatted DASD visible to the installation program, an interactive installation exits if the user does not agree to format all visible and unformatted DASDs. To circumvent this, only activate those DASDs that you will use during installation. You can always add more DASDs after installation is complete.

22.5.4. bootloader

The **bootloader** Kickstart command is required. It specifies how the boot loader should be installed. Use this command only once.

Syntax

```
bootloader [OPTIONS]
```

Options

- **--append=** - Specifies additional kernel parameters. To specify multiple parameters, separate them with spaces. For example:

```
bootloader --location=mbr --append="hdd=ide-scsi ide=nodma"
```

The **rhgb** and **quiet** parameters are automatically added when the **plymouth** package is installed, even if you do not specify them here or do not use the **--append=** command at all. To disable this behavior, explicitly disallow installation of **plymouth**:

```
%packages
-plymouth
%end
```

This option is useful for disabling mechanisms which were implemented to mitigate the Meltdown and Spectre speculative execution vulnerabilities found in most modern processors (CVE-2017-5754, CVE-2017-5753, and CVE-2017-5715). In some cases, these mechanisms may be unnecessary, and keeping them enabled causes decreased performance with no improvement in security. To disable these mechanisms, add the options to do so into your Kickstart file - for example, **bootloader --append="nopti noibrs noibpb"** on AMD64/Intel 64 systems.

WARNING

Ensure your system is not at risk of attack before disabling any of the vulnerability mitigation mechanisms. See the Red Hat vulnerability response article for information about the Meltdown and Spectre vulnerabilities.

- **--boot-drive=** - Specifies which drive the boot loader should be written to, and therefore which drive the computer will boot from. If you use a multipath device as the boot drive, specify the device by using its disk/by-id/dm-uuid-mpath-WWID name.

IMPORTANT

The **--boot-drive=** option is currently being ignored in Red Hat Enterprise Linux installations on 64-bit IBM Z systems by using the **zipl** boot loader. When **zipl** is installed, it determines the boot drive on its own.

- **--leavebootorder** - This option is applicable for Power and UEFI systems. The installation program adds Red Hat Enterprise Linux 10 to the list of the installed systems in UEFI. It does not add the installed system to the boot order. All existing boot entries as well as their order are preserved.

- **--driveorder=** - Specifies which drive is first in the BIOS boot order. For example:

```
bootloader --driveorder=sda,hda
```

- **--location=** - Specifies where the boot record is written. Valid values are the following:

  - **mbr** - The default option. Depends on whether the drive uses the Master Boot Record (MBR) or GUID Partition Table (GPT) scheme:
    On a GPT-formatted disk, this option installs stage 1.5 of the boot loader into the BIOS boot partition.

    On an MBR-formatted disk, stage 1.5 is installed into the empty space between the MBR and the first partition.

  - **partition** - Install the boot loader on the first sector of the partition containing the kernel.

  - **none** - Do not install the boot loader.

  In most cases, this option does not need to be specified.

- **--nombr** - Do not install the boot loader to the MBR.

- **--password=** - If using GRUB2, set the boot loader password to the one specified with this option. This should be used to restrict access to the GRUB2 shell, where arbitrary kernel options can be passed.
  If a password is specified, GRUB2 also asks for a user name. The user name is always **root**.

- **--iscrypted** - Normally, when you specify a boot loader password by using the **--password=** option, it is stored in the Kickstart file in plain text. If you want to encrypt the password, use this option and an encrypted password.
  To generate an encrypted password, use the **grub2-mkpasswd-pbkdf2** command, enter the password you want to use, and copy the command’s output (the hash starting with **grub.pbkdf2**) into the Kickstart file. An example **bootloader** Kickstart entry with an encrypted password looks similar to the following:

```
bootloader --iscrypted --
password=grub.pbkdf2.sha512.10000.5520C6C9832F3AC3D149AC0B24BE69E2D4FB0D
BEEDBD29CA1D30A044DE2645C4C7A291E585D4DC43F8A4D82479F8B95CA4BA4381
F8550510B75E8E0BB2938990.C688B6F0EF935701FF9BD1A8EC7FE5BD2333799C98F
28420C5CC8F1A2A233DE22C83705BB614EA17F3FDFDF4AC2161CEA3384E56EB38A
2E39102F5334C47405E
```

- **--timeout=** - Specifies the amount of time the boot loader waits before booting the default option (in seconds).
- **--default=** - Sets the default boot image in the boot loader configuration.
- **--extlinux** - Use the extlinux boot loader instead of GRUB2. This option only works on systems supported by extlinux.
- **--disabled** - This option is a stronger version of **--location=none**. While **--location=none** simply disables boot loader installation, **--disabled** disables boot loader installation and also disables installation of the package containing the boot loader, thus saving space.

Notes

- Red Hat recommends setting up a boot loader password on every system. An unprotected boot loader can allow a potential attacker to modify the system's boot options and gain unauthorized access to the system.
- In some cases, a special partition is required to install the boot loader on AMD64, Intel 64, and 64-bit ARM systems. The type and size of this partition depends on whether the disk you are installing the boot loader to use the Master Boot Record (MBR) or a GUID Partition Table (GPT) schema. For more information, see the Configuring boot loader section.
- The **sdX** (or **/dev/sdX**) format does not guarantee consistent device names across reboots, which can complicate the usage of some Kickstart commands. When a command requires a device node name, you can use any item from **/dev/disk** as an alternative. For example, instead of using the following device name:

  ```
  bootloader --boot-drive=sda
  ```

  You can use an entry similar to one of the following:

  ```
  bootloader --boot-drive=/dev/disk/by-path/pci-0000:00:05.0-scsi-0:0:0:0
  bootloader --boot-drive=/dev/disk/by-id/ata-ST3160815AS_6RA0C882
  ```

  By using this approach, the command always targets the same storage device. This is especially useful in large storage environments. To explore the available device names on the system, you can use the **ls -lR /dev/disk** command during the interactive installation. For more information about different ways to consistently refer to storage devices, see Persistent naming attributes.

22.5.5. autopart

The **autopart** Kickstart command is optional. It automatically creates partitions.

The automatically created partitions are: a root (**/**) partition (1 GiB or larger), a **swap** partition, and an appropriate **/boot** partition for the architecture. On large enough drives (50 GiB and larger), this also creates a **/home** partition. Use this command only once.

Syntax

```
autopart OPTIONS
```

Options

- **--type=** - Selects one of the predefined automatic partitioning schemes you want to use. Accepts the following values:
  - **lvm**: The LVM partitioning scheme.
  - **plain**: Regular partitions with no LVM.
  - **thinp**: The LVM Thin Provisioning partitioning scheme.
- **--fstype=** - Selects one of the available file system types. The available values are **ext2**, **ext3**, **ext4**, **xfs**, and **vfat**. The default file system is **xfs**.
- **--nohome** - Disables automatic creation of the **/home** partition.
- **--nolvm** - Do not use LVM for automatic partitioning. This option is equal to **--type=plain**.
- **--noboot** - Do not create a **/boot** partition.
- **--noswap** - Do not create a swap partition.
- **--encrypted** - Encrypts all partitions with Linux Unified Key Setup (LUKS). This is equivalent to checking the **Encrypt partitions** check box on the initial partitioning screen during a manual graphical installation.

  NOTE

  When encrypting one or more partitions, Anaconda attempts to gather 256 bits of entropy to ensure the partitions are encrypted securely. Gathering entropy can take some time - the process will stop after a maximum of 10 minutes, regardless of whether sufficient entropy has been gathered.

  The process can be speed up by interacting with the installation system (typing on the keyboard or moving the mouse). If you are installing in a virtual machine, you can also attach a **virtio-rng** device (a virtual random number generator) to the guest.

- **--luks-version=*LUKS_VERSION*** - Specifies which version of LUKS format should be used to encrypt the filesystem. This option is only meaningful if **--encrypted** is specified.
- **--passphrase=** - Provides a default system-wide passphrase for all encrypted devices.
- **--escrowcert=*URL_of_X.509_certificate*** - Stores data encryption keys of all encrypted volumes as files in **/root**, encrypted by using the X.509 certificate from the URL specified with *URL_of_X.509_certificate*. The keys are stored as a separate file for each encrypted volume. This option is only meaningful if **--encrypted** is specified.
- **--backuppassphrase** - Adds a randomly-generated passphrase to each encrypted volume. Store these passphrases in separate files in **/root**, encrypted by using the X.509 certificate specified with **--escrowcert**. This option is only meaningful if **--escrowcert** is specified.
- **--cipher=** - Specifies the type of encryption to use if the Anaconda default aes-xts-plain64 is not satisfactory. You must use this option together with the **--encrypted** option; by itself it has no effect. Available types of encryption are listed in the Security hardening document.

For example, **aes-xts-plain64**.

- ***--pbkdf=PBKDF*** - Sets Password-Based Key Derivation Function (PBKDF) algorithm for LUKS keyslot. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.
- ***--pbkdf-memory=PBKDF_MEMORY*** - Sets the memory cost for PBKDF. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.
- ***--pbkdf-time=PBKDF_TIME*** - Sets the number of milliseconds to spend with PBKDF passphrase processing. See also **--iter-time** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-iterations**.
- ***--pbkdf-iterations=PBKDF_ITERATIONS*** - Sets the number of iterations directly and avoids PBKDF benchmark. See also **--pbkdf-force-iterations** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-time**.

Notes

- The **autopart** option cannot be used together with the **part**/**partition**, **raid**, **logvol**, or **volgroup** options in the same Kickstart file.
- The **autopart** command is not mandatory, but you must include it if there are no **part** or **mount** commands in your Kickstart script.
- It is recommended to use the **autopart --nohome** Kickstart option when installing on a single FBA DASD of the CMS type. This ensures that the installation program does not create a separate /**home** partition. The installation then proceeds successfully.
- If you lose the LUKS passphrase, any encrypted partitions and their data is completely inaccessible. There is no way to recover a lost passphrase. However, you can save encryption passphrases with the **--escrowcert** and create backup encryption passphrases with the **--backuppassphrase** options.
- Ensure that the disk sector sizes are consistent when using **autopart**, **autopart --type=lvm**, or **autopart=thinp**.

22.5.6. reqpart

The **reqpart** Kickstart command is optional. It automatically creates partitions required by your hardware platform. These include a /**boot**/**efi** partition for systems with UEFI firmware, a **biosboot** partition for systems with BIOS firmware and GPT, and a **PRePBoot** partition for IBM Power Systems. Use this command only once.

Syntax

```
reqpart [--add-boot]
```

Options

- **--add-boot** - Creates a separate /**boot** partition in addition to the platform-specific partition created by the base command.

Note

- This command cannot be used together with **autopart**, because **autopart** does everything the **reqpart** command does and, in addition, creates other partitions or logical volumes such as / and **swap**. In contrast with **autopart**, this command only creates platform-specific partitions and leaves the rest of the drive empty, allowing you to create a custom layout.

22.5.7. part or partition

The **part** or **partition** Kickstart command is required. It creates a partition on the system.

Syntax

```
part|partition mntpoint [OPTIONS]
```

Options

- *mntpoint* - Where the partition is mounted. The value must be of one of the following forms:
  - /***path***
    For example, /, /**usr**, /**home**
  - **swap**
    The partition is used as swap space.

    To determine the size of the swap partition automatically, use the **--recommended** option:

    ```
    swap --recommended
    ```

    The size assigned will be effective but not precisely calibrated for your system.

    To determine the size of the swap partition automatically but also allow extra space for your system to hibernate, use the **--hibernation** option:

    ```
    swap --hibernation
    ```

    The size assigned will be equivalent to the swap space assigned by **--recommended** plus the amount of RAM on your system. For the swap sizes assigned by these commands, see Recommended Partitioning Scheme for AMD64, Intel 64, and 64-bit ARM systems.
  - **raid.*id***
    The partition is used for software RAID (see **raid**).
  - **pv.*id***
    The partition is used for LVM (see **logvol**).
  - **biosboot**
    The partition will be used for a BIOS Boot partition. A 1 MiB BIOS boot partition is necessary on BIOS-based AMD64 and Intel 64 systems using a GUID Partition Table (GPT); the boot loader will be installed into it. It is not necessary on UEFI systems. See also the **bootloader** command.

   - **/boot/efi**
     An EFI System Partition. A 50 MiB EFI partition is necessary on UEFI-based AMD64, Intel 64, and 64-bit ARM; the recommended size is 200 MiB. It is not necessary on BIOS systems. See also the **bootloader** command.

- **--size=** - The minimum partition size in MiB. Specify an integer value here such as **500** (do not include the unit). Installation fails if size specified is too small. Set the **--size** value as the minimum amount of space you require. For size recommendations, see Recommended Partitioning Scheme.

- **--grow** - Specifies the partition to grow to fill available space (if any), or up to the maximum size setting, if one is specified. If you use **--grow** without setting **--maxsize=** on a swap partition, Anaconda limits the maximum size of the swap partition. For systems that have less than 2 GiB of physical memory, the imposed limit is twice the amount of physical memory. For systems with more than 2 GiB, the imposed limit is the size of physical memory plus 2GiB.

- **--maxsize=** - The maximum partition size in MiB when the partition is set to grow. Specify an integer value here such as **500** (do not include the unit).

- **--noformat** - Specifies that the partition should not be formatted, for use with the **--onpart** command.

- **--onpart=** or **--usepart=** - Specifies the device on which to place the partition. Uses an existing blank device and formats it to the new specified type. For example:

  ```
  partition /home --onpart=hda1
  ```

  puts /**home** on /**dev/hda1**.

  These options can also add a partition to a logical volume. For example:

  ```
  partition pv.1 --onpart=hda2
  ```

  The device must already exist on the system; the **--onpart** option will not create it.

  It is also possible to specify an entire drive, rather than a partition, in which case Anaconda will format and use the drive without creating a partition table. However, installation of GRUB2 is not supported on a device formatted in this way, and must be placed on a drive with a partition table.

  ```
  partition pv.1 --onpart=hdb
  ```

- **--ondisk=** or **--ondrive=** - Creates a partition (specified by the **part** command) on an existing disk.This command always creates a partition. Forces the partition to be created on a particular disk. For example, **--ondisk=sdb** puts the partition on the second SCSI disk on the system.

  NOTE

  To specify a multipath device, use the format **disk/by-id/scsi-WWID**, where WWID is the world-wide identifier for the device. For example, to specify a disk with WWID **58095BEC5510947BE8C0360F604351918**, use:

```
part / --size=5000 --ondisk=disk/by-id/scsi-58095BEC5510947BE8C0360F604351918
```

This format is preferable for all multipath devices, but if errors arise, multipath devices that do not use logical volume manager (LVM) can also be specified by using the format **disk/by-id/dm-uuid-mpath-WWID**, where WWID is the world-wide identifier for the device. For example, to specify a disk with WWID **2416CD96995134CA5D787F00A5AA11017**, use:

```
part / --size=5000 --ondisk=disk/by-id/dm-uuid-mpath-
2416CD96995134CA5D787F00A5AA11017
```

Never specify multipath devices by device names like mpatha. Device names such as this are not specific to a particular disk. The disk named /dev/mpatha during installation might not be the one that you expect it to be. Therefore, the **part** command could target the wrong disk.

- **--asprimary** - Forces the partition to be allocated as a *primary* partition. If the partition cannot be allocated as primary (usually due to too many primary partitions being already allocated), the partitioning process fails. This option only makes sense when the disk uses a Master Boot Record (MBR); for GUID Partition Table (GPT)-labeled disks this option has no meaning.

- **--fsprofile=** - Specifies a usage type to be passed to the program that makes a filesystem on this partition. A usage type defines a variety of tuning parameters to be used when making a filesystem. For this option to work, the filesystem must support the concept of usage types and there must be a configuration file that lists valid types. For **ext2**, **ext3**, **ext4**, this configuration file is **/etc/mke2fs.conf**.

- **--mkfsoptions=** - Specifies additional parameters to be passed to the program that makes a filesystem on this partition. This is similar to **--fsprofile** but works for all filesystems, not just the ones that support the profile concept. No processing is done on the list of arguments, so they must be supplied in a format that can be passed directly to the mkfs program. This means multiple options should be comma-separated or surrounded by double quotes, depending on the filesystem. For example,

```
part /opt/foo1 --size=512 --fstype=ext4 --mkfsoptions="-O
^has_journal,^flex_bg,^metadata_csum"

part /opt/foo2 --size=512 --fstype=xfs --mkfsoptions="-m bigtime=0,finobt=0"
```

For details, see the man pages of the filesystems you are creating. For example, **mkfs.ext4** or **mkfs.xfs**.

- **--fstype=** - Sets the file system type for the partition. Valid values are **xfs**, **ext2**, **ext3**, **ext4**, **swap**, **vfat**, **efi** and **biosboot**.

- **--fsoptions** - Specifies a free form string of options to be used when mounting the filesystem. This string will be copied into the **/etc/fstab** file of the installed system and should be enclosed in quotes.

NOTE

In the EFI system partition (**/boot/efi**), anaconda hard codes the value and ignores the users specified **--fsoptions** values.

- **--label=** - assign a label to an individual partition.
- **--recommended** - Determine the size of the partition automatically. For details about the recommended scheme, see Recommended Partitioning Scheme for AMD64, Intel 64, and 64-bit ARM. This option can only be used for partitions which result in a file system such as the /**boot** partition and **swap** space. It cannot be used to create LVM physical volumes or RAID members.
- **--onbiosdisk** - Forces the partition to be created on a particular disk as discovered by the BIOS.
- **--encrypted** - Specifies that this partition should be encrypted with Linux Unified Key Setup (LUKS) by using the passphrase provided in the **--passphrase** option. If you do not specify a passphrase, Anaconda uses the default, system-wide passphrase set with the **autopart --passphrase** command, or stops the installation and prompts you to provide a passphrase if no default is set.

  NOTE

  When encrypting one or more partitions, Anaconda attempts to gather 256 bits of entropy to ensure the partitions are encrypted securely. Gathering entropy can take some time - the process will stop after a maximum of 10 minutes, regardless of whether sufficient entropy has been gathered. The process can be sped up by interacting with the installation system (typing on the keyboard or moving the mouse). If you are installing in a virtual machine, you can also attach a **virtio-rng** device (a virtual random number generator) to the guest.

- **--luks-version=*LUKS_VERSION*** - Specifies which version of LUKS format should be used to encrypt the filesystem. This option is only meaningful if **--encrypted** is specified.
- **--passphrase=** - Specifies the passphrase to use when encrypting this partition. You must use this option together with the **--encrypted** option; by itself it has no effect.
- **--cipher=** - Specifies the type of encryption to use if the Anaconda default aes-xts-plain64 is not satisfactory. You must use this option together with the **--encrypted** option; by itself it has no effect. Available types of encryption are listed in the Security hardening document. For example, **aes-xts-plain64**.
- **--escrowcert=*URL_of_X.509_certificate*** - Store data encryption keys of all encrypted partitions as files in /**root**, encrypted by using the X.509 certificate from the URL specified with *URL_of_X.509_certificate*. The keys are stored as a separate file for each encrypted partition. This option is only meaningful if **--encrypted** is specified.
- **--backuppassphrase** - Add a randomly-generated passphrase to each encrypted partition. Store these passphrases in separate files in /**root**, encrypted by using the X.509 certificate specified with **--escrowcert**. This option is only meaningful if **--escrowcert** is specified.
- **--pbkdf=*PBKDF*** - Sets Password-Based Key Derivation Function (PBKDF) algorithm for LUKS keyslot. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.
- **--pbkdf-memory=*PBKDF_MEMORY*** - Sets the memory cost for PBKDF. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.

- **--pbkdf-time=*PBKDF_TIME*** - Sets the number of milliseconds to spend with PBKDF passphrase processing. See also **--iter-time** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-iterations**.
- **--pbkdf-iterations=*PBKDF_ITERATIONS*** - Sets the number of iterations directly and avoids PBKDF benchmark. See also **--pbkdf-force-iterations** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-time**.
- **--resize** - Resize an existing partition. When using this option, specify the target size (in MiB) by using the **--size=** option and the target partition by using the **--onpart=** option.

Notes

- The **part** command is not mandatory, but you must include either **part**, **autopart** or **mount** in your Kickstart script.
- If partitioning fails for any reason, diagnostic messages appear on virtual console 3.
- All partitions created are formatted as part of the installation process unless **--noformat** and **--onpart** are used.
- The **sdX** (or **/dev/sdX**) format does not guarantee consistent device names across reboots, which can complicate the usage of some Kickstart commands. When a command requires a device node name, you can use any item from **/dev/disk** as an alternative. For example, instead of using the following device name:

  ```
  part / --fstype=xfs --onpart=sda1
  ```

  You can use an entry similar to one of the following:

  ```
  part / --fstype=xfs --onpart=/dev/disk/by-path/pci-0000:00:05.0-scsi-0:0:0:0-part1
  part / --fstype=xfs --onpart=/dev/disk/by-id/ata-ST3160815AS_6RA0C882-part1
  ```

  By using this approach, the command always targets the same storage device. This is especially useful in large storage environments. To explore the available device names on the system, you can use the **ls -lR /dev/disk** command during the interactive installation. For more information about different ways to consistently refer to storage devices, see Persistent naming attributes.
- If you lose the LUKS passphrase, any encrypted partitions and their data is completely inaccessible. There is no way to recover a lost passphrase. However, you can save encryption passphrases with the **--escrowcert** and create backup encryption passphrases with the **--backuppassphrase** options.

22.5.8. raid

The **raid** Kickstart command is optional. It assembles a software RAID device.

Syntax

```
raid mntpoint --level=level --device=device-name partitions*
```

Options

- *mntpoint* - Location where the RAID file system is mounted. If it is **/**, the RAID level must be 1 unless a boot partition (**/boot**) is present. If a boot partition is present, the **/boot** partition must be level 1 and the root (**/**) partition can be any of the available types. The *partitions** (which denotes that multiple partitions can be listed) lists the RAID identifiers to add to the RAID array.

IMPORTANT

- On IBM Power Systems, if a RAID device has been prepared and has not been reformatted during the installation, ensure that the RAID metadata version is **0.90** or **1.0** if you intend to put the **/boot** and PReP partitions on the RAID device. The **mdadm** metadata versions **1.1** and **1.2** are not supported for the **/boot** and PReP partitions.
- The **PReP** Boot partitions are not required on PowerNV systems.

- **--level=** - RAID level to use (0, 1, 4, 5, 6, or 10).
- **--device=** - Name of the RAID device to use - for example, **--device=root**.

IMPORTANT

Do not use **mdraid** names in the form of **md0** - these names are not guaranteed to be persistent. Instead, use meaningful names such as **root** or **swap**. Using meaningful names creates a symbolic link from **/dev/md/*name*** to whichever **/dev/md*X*** node is assigned to the array.

If you have an old (v0.90 metadata) array that you cannot assign a name to, you can specify the array by a filesystem label or UUID. For example, **--device=LABEL=root** or **--device=UUID=93348e56-4631-d0f0-6f5b-45c47f570b88**.

You can use the UUID of the file system on the RAID device or UUID of the RAID device itself. The UUID of the RAID device should be in the **8-4-4-4-12** format. UUID reported by mdadm is in the **8:8:8:8** format which needs to be changed. For example **93348e56:4631d0f0:6f5b45c4:7f570b88** should be changed to **93348e56-4631-d0f0-6f5b-45c47f570b88**.

- **--chunksize=** - Sets the chunk size of a RAID storage in KiB. In certain situations, using a different chunk size than the default (**512 Kib**) can improve the performance of the RAID.
- **--spares=** - Specifies the number of spare drives allocated for the RAID array. Spare drives are used to rebuild the array in case of drive failure.
- **--fsprofile=** - Specifies a usage type to be passed to the program that makes a filesystem on this partition. A usage type defines a variety of tuning parameters to be used when making a filesystem. For this option to work, the filesystem must support the concept of usage types and there must be a configuration file that lists valid types. For ext2, ext3, and ext4, this configuration file is **/etc/mke2fs.conf**.

- **--fstype=** - Sets the file system type for the RAID array. Valid values are **xfs**, **ext2**, **ext3**, **ext4**, **swap**, and **vfat**.

- **--fsoptions=** - Specifies a free form string of options to be used when mounting the filesystem. This string will be copied into the **/etc/fstab** file of the installed system and should be enclosed in quotes. In the EFI system partition (**/boot/efi**), anaconda hard codes the value and ignores the users specified **--fsoptions** values.

- **--mkfsoptions=** - Specifies additional parameters to be passed to the program that makes a filesystem on this partition. No processing is done on the list of arguments, so they must be supplied in a format that can be passed directly to the mkfs program. This means multiple options should be comma-separated or surrounded by double quotes, depending on the filesystem. For example,

  ```
  part /opt/foo1 --size=512 --fstype=ext4 --mkfsoptions="-O
  ^has_journal,^flex_bg,^metadata_csum"

  part /opt/foo2 --size=512 --fstype=xfs --mkfsoptions="-m bigtime=0,finobt=0"
  ```

  For details, see the man pages of the filesystems you are creating. For example, **mkfs.ext4** or **mkfs.xfs**.

- **--label=** - Specify the label to give to the filesystem to be made. If the given label is already in use by another filesystem, a new label will be created.

- **--noformat** - Use an existing RAID device and do not format the RAID array.

- **--useexisting** - Use an existing RAID device and reformat it.

- **--encrypted** - Specifies that this RAID device should be encrypted with Linux Unified Key Setup (LUKS) by using the passphrase provided in the **--passphrase** option. If you do not specify a passphrase, Anaconda uses the default, system-wide passphrase set with the **autopart --passphrase** command, or stops the installation and prompts you to provide a passphrase if no default is set.

  NOTE

  When encrypting one or more partitions, Anaconda attempts to gather 256 bits of entropy to ensure the partitions are encrypted securely. Gathering entropy can take some time - the process will stop after a maximum of 10 minutes, regardless of whether sufficient entropy has been gathered.

  The process can be sped up by interacting with the installation system (typing on the keyboard or moving the mouse). If you are installing in a virtual machine, you can also attach a **virtio-rng** device (a virtual random number generator) to the guest.

- **--luks-version=*LUKS_VERSION*** - Specifies which version of LUKS format should be used to encrypt the filesystem. This option is only meaningful if **--encrypted** is specified.

- **--cipher=** - Specifies the type of encryption to use if the Anaconda default aes-xts-plain64 is not satisfactory. You must use this option together with the **--encrypted** option; by itself it has no effect. Available types of encryption are listed in the Security hardening document. For example, **aes-xts-plain64**.

- **--passphrase=** - Specifies the passphrase to use when encrypting this RAID device. You must use this option together with the **--encrypted** option; by itself it has no effect.
- **--escrowcert=*URL_of_X.509_certificate*** - Store the data encryption key for this device in a file in /**root**, encrypted by using the X.509 certificate from the URL specified with *URL_of_X.509_certificate*. This option is only meaningful if **--encrypted** is specified.
- **--backuppassphrase** - Add a randomly-generated passphrase to this device. Store the passphrase in a file in /**root**, encrypted by using the X.509 certificate specified with **--escrowcert**. This option is only meaningful if **--escrowcert** is specified.
- **--pbkdf=*PBKDF*** - Sets Password-Based Key Derivation Function (PBKDF) algorithm for LUKS keyslot. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.
- **--pbkdf-memory=*PBKDF_MEMORY*** - Sets the memory cost for PBKDF. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.
- **--pbkdf-time=*PBKDF_TIME*** - Sets the number of milliseconds to spend with PBKDF passphrase processing. See also **--iter-time** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-iterations**.
- **--pbkdf-iterations=*PBKDF_ITERATIONS*** - Sets the number of iterations directly and avoids PBKDF benchmark. See also **--pbkdf-force-iterations** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-time**.

Example

The following example shows how to create a RAID level 1 partition for /, and a RAID level 5 for /**home**, assuming there are three SCSI disks on the system. It also creates three swap partitions, one on each drive.

```
part raid.01 --size=6000 --ondisk=sda
part raid.02 --size=6000 --ondisk=sdb
part raid.03 --size=6000 --ondisk=sdc
part swap --size=512 --ondisk=sda
part swap --size=512 --ondisk=sdb
part swap --size=512 --ondisk=sdc
part raid.11 --size=1 --grow --ondisk=sda
part raid.12 --size=1 --grow --ondisk=sdb
part raid.13 --size=1 --grow --ondisk=sdc
raid / --level=1 --device=rhel8-root --label=rhel8-root raid.01 raid.02 raid.03
raid /home --level=5 --device=rhel8-home --label=rhel8-home raid.11 raid.12 raid.13
```

Note

- If you lose the LUKS passphrase, any encrypted partitions and their data is completely inaccessible. There is no way to recover a lost passphrase. However, you can save encryption passphrases with the **--escrowcert** and create backup encryption passphrases with the **--backuppassphrase** options.

22.5.9. volgroup

The **volgroup** Kickstart command is optional. It creates a Logical Volume Manager (LVM) group.

Syntax

```
volgroup name [OPTIONS] [partition*]
```

Mandatory options

- *name* - Name of the new volume group.

Options

- *partition* - Physical volume partitions to use as backing storage for the volume group. At least one partition has to be specified when creating a new volume group.
- **--noformat** - Use an existing volume group and do not format it.
- **--useexisting** - Use an existing volume group and reformat it. If you use this option, do not specify a *partition*. For example:

  ```
  volgroup rhel00 --useexisting --noformat
  ```

- **--pesize=** - Set the size of the volume group's physical extents in KiB. The default value is 4096 (4 MiB), and the minimum value is 1024 (1 MiB).
- **--reserved-space=** - Specify an amount of space to leave unused in a volume group in MiB. Applicable only to newly created volume groups.
- **--reserved-percent=** - Specify a percentage of total volume group space to leave unused. Applicable only to newly created volume groups.

Notes

- Create the partition first, then create the logical volume group, and then create the logical volume. For example:

```
part pv.01 --size 10000
volgroup my_volgrp pv.01
logvol / --vgname=my_volgrp --size=2000 --name=root
```

- Do not use the dash (**-**) character in logical volume and volume group names when installing Red Hat Enterprise Linux by using Kickstart. If this character is used, the installation finishes normally, but the **/dev/mapper/** directory will list these volumes and volume groups with every dash doubled. For example, a volume group named **volgrp-01** containing a logical volume named **logvol-01** will be listed as **/dev/mapper/volgrp--01-logvol--01**.
  This limitation only applies to newly created logical volume and volume group names. If you are reusing existing ones by using the **--noformat** option, their names will not be changed.

22.5.10. logvol

The **logvol** Kickstart command is optional. It creates a logical volume for Logical Volume Manager (LVM).

Syntax

```
logvol mntpoint --vgname=name --name=name [OPTIONS]
```

Mandatory options

- ***mntpoint***: The mount point where the partition is mounted. Must be of one of the following forms:
  - /***path***
    For example, / or /**home**
  - **swap**
    The partition is used as swap space.

    To determine the size of the swap partition automatically, use the **--recommended** option:

    ```
    swap --recommended
    ```

    To determine the size of the swap partition automatically and also allow extra space for your system to hibernate, use the **--hibernation** option:

    ```
    swap --hibernation
    ```

    The size assigned will be equivalent to the swap space assigned by **--recommended** plus the amount of RAM on your system. For the swap sizes assigned by these commands, see Recommended Partitioning Scheme for AMD64, Intel 64, and 64-bit ARM systems.
- **--vgname=*name***: Name of the volume group.
- **--name=*name***: Name of the logical volume.

Optional options

- **--noformat**: Use an existing logical volume and do not format it.
- **--useexisting**: Use an existing logical volume and reformat it.
- **--fstype=**: Sets the file system type for the logical volume. Valid values are **xfs**, **ext2**, **ext3**, **ext4**, **swap**, and **vfat**.
- **--fsoptions=**: Specifies a free form string of options to be used when mounting the filesystem. This string will be copied into the /**etc**/**fstab** file of the installed system and should be enclosed in quotes.

  NOTE

  In the EFI system partition (/**boot**/**efi**), anaconda hard codes the value and ignores the users specified **--fsoptions** values.
- **--mkfsoptions=**: Specifies additional parameters to be passed to the program that makes a filesystem on this partition. No processing is done on the list of arguments, so they must be

supplied in a format that can be passed directly to the mkfs program. This means multiple options should be comma-separated or surrounded by double quotes, depending on the filesystem. For example,

```
part /opt/foo1 --size=512 --fstype=ext4 --mkfsoptions="-O
^has_journal,^flex_bg,^metadata_csum"

part /opt/foo2 --size=512 --fstype=xfs --mkfsoptions="-m bigtime=0,finobt=0"
```

For details, see the man pages of the filesystems you are creating. For example, **mkfs.ext4** or **mkfs.xfs**.

- **--fsprofile=**: Specifies a usage type to be passed to the program that makes a filesystem on this partition. A usage type defines a variety of tuning parameters to be used when making a filesystem. For this option to work, the filesystem must support the concept of usage types and there must be a configuration file that lists valid types. For **ext2**, **ext3**, and **ext4**, this configuration file is **/etc/mke2fs.conf**.
- **--label=**: Sets a label for the logical volume.
- **--grow**: Extends the logical volume to occupy the available space (if any), or up to the maximum size specified, if any. The option must be used only if you have pre-allocated a minimum storage space in the disk image, and would want the volume to grow and occupy the available space. In a physical environment, this is a one-time-action. However, in a virtual environment, the volume size increases as and when the virtual machine writes any data to the virtual disk.
- **--size=**: The size of the logical volume in MiB. This option cannot be used together with the **--percent=** option.
- **--percent=**: The size of the logical volume, as a percentage of the free space in the volume group after any statically-sized logical volumes are taken into account. This option cannot be used together with the **--size=** option.

IMPORTANT

When creating a new logical volume, you must either specify its size statically by using the **--size=** option, or as a percentage of remaining free space by using the **--percent=** option. You cannot use both of these options on the same logical volume.

- **--maxsize=**: The maximum size in MiB when the logical volume is set to grow. Specify an integer value here such as **500** (do not include the unit).
- **--recommended**: Use this option when creating a logical volume to determine the size of this volume automatically, based on your system’s hardware. For details about the recommended scheme, see Recommended Partitioning Scheme for AMD64, Intel 64, and 64-bit ARM systems.
- **--resize**: Resize a logical volume. If you use this option, you must also specify **--useexisting** and **--size**.
- **--encrypted**: Specifies that this logical volume should be encrypted with Linux Unified Key Setup (LUKS) by using the passphrase provided in the **--passphrase=** option. If you do not

specify a passphrase, the installation program stops the installation and prompts you to provide a passphrase if no default is set.

NOTE

When encrypting one or more partitions, Anaconda attempts to gather 256 bits of entropy to ensure the partitions are encrypted securely. Gathering entropy can take some time - the process will stop after a maximum of 10 minutes, regardless of whether sufficient entropy has been gathered. The process can be sped up by interacting with the installation system (typing on the keyboard or moving the mouse). If you are installing in a virtual machine, you can also attach a **virtio-rng** device (a virtual random number generator) to the guest.

- **--passphrase=**: Specifies the passphrase to use when encrypting this logical volume. You must use this option together with the **--encrypted** option; it has no effect by itself.
- **--cipher=**: Specifies the type of encryption to use if the Anaconda default aes-xts-plain64 is not satisfactory. You must use this option together with the **--encrypted** option; by itself it has no effect. Available types of encryption are listed in Security hardening. For example, **aes-xts-plain64**.
- **--escrowcert=*URL_of_X.509_certificate***: Store data encryption keys of all encrypted volumes as files in **/root**, encrypted by using the X.509 certificate from the URL specified with *URL_of_X.509_certificate*. The keys are stored as a separate file for each encrypted volume. This option is only meaningful if **--encrypted** is specified.
- **--luks-version=*LUKS_VERSION***: Specifies which version of LUKS format should be used to encrypt the filesystem. This option is only meaningful if **--encrypted** is specified.
- **--backuppassphrase**: Add a randomly-generated passphrase to each encrypted volume. Store these passphrases in separate files in **/root**, encrypted by using the X.509 certificate specified with **--escrowcert**. This option is only meaningful if **--escrowcert** is specified.
- **--pbkdf=*PBKDF***: Sets Password-Based Key Derivation Function (PBKDF) algorithm for LUKS keyslot. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.
- **--pbkdf-memory=*PBKDF_MEMORY***: Sets the memory cost for PBKDF. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.
- **--pbkdf-time=*PBKDF_TIME***: Sets the number of milliseconds to spend with PBKDF passphrase processing. See also **--iter-time** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-iterations**.
- **--pbkdf-iterations=*PBKDF_ITERATIONS***: Sets the number of iterations directly and avoids PBKDF benchmark. See also **--pbkdf-force-iterations** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-time**.
- **--thinpool**: Creates a thin pool of logical volume. (Use a mount point of **none**)
- **--metadatasize=*size***: Specify the metadata area size (in MiB) for a new thin pool device.

- **--chunksize=*size***: Specify the chunk size (in KiB) for a new thin pool device.
- **--thin**: Create a thin logical volume. (Requires use of **--poolname**)
- **--poolname=*name***: Specify the name of the thin pool in which to create a thin logical volume. Requires the **--thin** option.
- **--profile=*name***: Specify the configuration profile name to use with thin logical volumes. If used, the name will also be included in the metadata for the given logical volume. By default, the available profiles are **default** and **thin-performance** and are defined in the **/etc/lvm/profile/** directory. See the **lvm(8)** man page for additional information.
- **--cachepvs=**: A comma-separated list of physical volumes which should be used as a cache for this volume.
- **--cachemode=**: Specify which mode should be used to cache this logical volume - either **writeback** or **writethrough**.

NOTE

For more information about cached logical volumes and their modes, see the **lvmcache(7)** man page on your system.

- **--cachesize=**: Size of cache attached to the logical volume, specified in MiB. This option requires the **--cachepvs=** option.

Notes

- Do not use the dash (**-**) character in logical volume and volume group names when installing Red Hat Enterprise Linux by using Kickstart. If this character is used, the installation finishes normally, but the **/dev/mapper/** directory will list these volumes and volume groups with every dash doubled. For example, a volume group named **volgrp-01** containing a logical volume named **logvol-01** will be listed as **/dev/mapper/volgrp—01-logvol—01**. This limitation only applies to newly created logical volume and volume group names. If you are reusing existing ones by using the **--noformat** option, their names will not be changed.
- If you lose the LUKS passphrase, any encrypted partitions and their data is completely inaccessible. There is no way to recover a lost passphrase. However, you can save encryption passphrases with the **--escrowcert** and create backup encryption passphrases with the **--backuppassphrase** options.

Examples

- Create the partition first, create the volume group, and then create the logical volume:

```
part pv.01 --size 3000
volgroup myvg pv.01
logvol / --vgname=myvg --size=2000 --name=rootvol
```

- Create the partition first, create the volume group, and then create the logical volume to occupy 90% of the remaining space in the volume group:

```
part pv.01 --size 1 --grow
volgroup myvg pv.01
logvol / --vgname=myvg --name=rootvol --percent=90
```

Additional resources

- Configuring and managing logical volumes

22.5.11. snapshot

The **snapshot** Kickstart command is optional. Use it to create LVM thin volume snapshots during the installation process. This enables you to back up a logical volume before or after the installation.

To create multiple snapshots, add the **snaphost** Kickstart command multiple times.

Syntax

```
snapshot vg_name/lv_name --name=snapshot_name --when=pre-install|post-install
```

Options

- ***vg_name*/*lv_name*** - Sets the name of the volume group and logical volume to create the snapshot from.
- **--name=*snapshot_name*** - Sets the name of the snapshot. This name must be unique within the volume group.
- **--when=*pre-install|post-install*** - Sets if the snapshot is created before the installation begins or after the installation is completed.

22.5.12. mount

The **mount** Kickstart command is optional. It assigns a mount point to an existing block device, and optionally reformats it to a given format.

Syntax

```
mount [OPTIONS] device mountpoint
```

Mandatory options

- ***device*** - The block device to mount.
- ***mountpoint*** - Where to mount the ***device***. It must be a valid mount point, such as **/** or **/usr**, or **none** if the device is unmountable (for example **swap**).

Optional options

- **--reformat=** - Specifies a new format (such as **ext4**) to which the device should be reformatted.
- **--mkfsoptions=** - Specifies additional options to be passed to the command which creates

the new file system specified in **--reformat=**. The list of options provided here is not processed, so they must be specified in a format that can be passed directly to the **mkfs** program. The list of options should be either comma-separated or surrounded by double quotes, depending on the file system. See the **mkfs** man page for the file system you want to create (for example **mkfs.ext4(8)** or **mkfs.xfs(8)**) for specific details.

- **--mountoptions=** - Specifies a free form string that contains options to be used when mounting the file system. The string will be copied to the **/etc/fstab** file on the installed system and should be enclosed in double quotes. See the **mount(8)** man page for a full list of mount options, and **fstab(5)** for basics.

Notes

- Unlike most other storage configuration commands in Kickstart, **mount** does not require you to describe the entire storage configuration in the Kickstart file. You only need to ensure that the described block device exists on the system. However, if you want to *create* the storage stack with all the devices mounted, you must use other commands such as **part** to do so.

- You cannot use **mount** together with other storage-related commands such as **part**, **logvol**, or **autopart** in the same Kickstart file.

22.5.13. zipl

The **zipl** Kickstart command is optional. It specifies the ZIPL configuration for 64-bit IBM Z. Use this command only once.

Options

- **--secure-boot** - Enables secure boot if it is supported by the installing system.

- **--no-secure-boot** - Disables secure boot.

- **--force-secure-boot** - Enables secure boot unconditionally.

Notes

- When installed on a system that is later than IBM z14, the installed system cannot be booted from an IBM z14 or earlier model.

- Installation is not supported on IBM z14 and earlier models.

- Secure Boot is not supported on IBM z14 and earlier models. Use **--no-secure-boot** if you intend to boot the installed system on IBM z14 and earlier models.

22.5.14. fcoe

The **fcoe** Kickstart command is optional. It specifies which FCoE devices should be activated automatically in addition to those discovered by Enhanced Disk Drive Services (EDD).

Syntax

```
fcoe --nic=name [OPTIONS]
```

Options

- **--nic=** (required) - The name of the device to be activated.
- **--dcb=** - Establish Data Center Bridging (DCB) settings.
- **--autovlan** - Discover VLANs automatically. This option is enabled by default.

22.5.15. iscsi

The **iscsi** Kickstart command is optional. It specifies additional iSCSI storage to be attached during installation.

Syntax

```
iscsi --ipaddr=address [OPTIONS]
```

Mandatory options

- **--ipaddr=** (required) - the IP address of the target to connect to.

Optional options

- **--port=** (required) - the port number. If not present, **--port=3260** is used automatically by default.
- **--target=** - the target IQN (iSCSI Qualified Name).
- **--iface=** - bind the connection to a specific network interface instead of using the default one determined by the network layer. Once used, it must be specified in all instances of the **iscsi** command in the entire Kickstart file.
- **--user=** - the user name required to authenticate with the target
- **--password=** - the password that corresponds with the user name specified for the target
- **--reverse-user=** - the user name required to authenticate with the initiator from a target that uses reverse CHAP authentication
- **--reverse-password=** - the password that corresponds with the user name specified for the initiator

Notes

- If you use the **iscsi** command, you must also assign a name to the iSCSI node by using the **iscsiname** command. The **iscsiname** command must appear before the **iscsi** command in the Kickstart file.
- Wherever possible, configure iSCSI storage in the system BIOS or firmware (iBFT for Intel systems) rather than use the **iscsi** command. Anaconda automatically detects and uses disks configured in BIOS or firmware and no special configuration is necessary in the Kickstart file.
- [illegible]

- If you must use the **iscsi** command, ensure that networking is activated at the beginning of the installation, and that the **iscsi** command appears in the Kickstart file *before* you refer to iSCSI disks with commands such as **clearpart** or **ignoredisk**.

22.5.16. iscsiname

The **iscsiname** Kickstart command is optional. It assigns a name to an iSCSI node specified by the **iscsi** command. Use this command only once.

Syntax

```
iscsiname iqname
```

Options

- ***iqname*** - Name to assign to the iSCSI node.

Note

- If you use the **iscsi** command in your Kickstart file, you must specify **iscsiname** *earlier* in the Kickstart file.

22.5.17. zfcp

The **zfcp** Kickstart command is optional. It defines a Fibre channel device.

This option only applies on 64-bit IBM Z.

Syntax

```
zfcp --devnum=devnum [--wwpn=wwpn --fcplun=lun]
```

Options

- **--devnum=** - The device number (zFCP adapter device bus ID).
- **--wwpn=** - The device’s World Wide Port Name (WWPN). Takes the form of a 16-digit number, preceded by **0x**.
- **--fcplun=** - The device’s Logical Unit Number (LUN). Takes the form of a 16-digit number, preceded by **0x**.

Example

```
zfcp --devnum=0.0.4000 --wwpn=0x5005076300C213e9 --fcplun=0x5022000000000000
zfcp --devnum=0.0.4000
```

22.6. KICKSTART COMMANDS FOR ADDONS SUPPLIED WITH THE RHEL INSTALLATION PROGRAM

The Kickstart commands in this section are related to add-ons supplied by default with the Red Hat Enterprise Linux installation program.

22.6.1. %addon com_redhat_kdump

The **%addon com_redhat_kdump** Kickstart command is optional. This command configures the kdump kernel crash dumping mechanism.

Syntax

```
%addon com_redhat_kdump [OPTIONS]
%end
```

NOTE

The syntax for this command is different because it is an add-on rather than a built-in Kickstart command.

Kdump is a kernel crash dumping mechanism that allows you to save the contents of the system's memory for later analysis. It relies on **kexec**, which can be used to boot a Linux kernel from the context of another kernel without rebooting the system, and preserve the contents of the first kernel's memory that would otherwise be lost.

In case of a system crash, **kexec** boots into a second kernel (a capture kernel). This capture kernel resides in a reserved part of the system memory. Kdump then captures the contents of the crashed kernel's memory (a crash dump) and saves it to a specified location. The location cannot be configured using this Kickstart command; it must be configured after the installation by editing the **/etc/kdump.conf** configuration file.

For more information about Kdump, see the Installing kdump.

Options

- **--enable** - Enable kdump on the installed system.
- **--disable** - Disable kdump on the installed system.
- **--reserve-mb=** - The amount of memory you want to reserve for kdump, in MiB. For example:

  ```
  %addon com_redhat_kdump --enable --reserve-mb=128
  %end
  ```

  You can also specify **auto** instead of a numeric value. In that case, the installation program will determine the amount of memory automatically based on the criteria described in the Memory requirements for kdump section of the *Managing, monitoring and updating the kernel* document.

  If you enable kdump and do not specify a **--reserve-mb=** option, the value **auto** will be used.
- **--enablefadump** - Enable firmware-assisted dumping on systems which allow it (notably, IBM Power Systems servers).

22.7. KICKSTART COMMANDS FOR SYSTEM RECOVERY

The Kickstart command in this section repairs an installed system.

22.7.1. rescue

The **rescue** Kickstart command is optional. It provides a shell environment with root privileges and a set of system management tools to repair the installation and to troubleshoot the issues like:

- Mount file systems as read-only
- Blocklist or add a driver provided on a driver disc
- Install or upgrade system packages
- Manage partitions

NOTE

The Kickstart rescue mode is different from the rescue mode and emergency mode, which are provided as part of the systemd and service manager.

The **rescue** command does not modify the system on its own. It only sets up the rescue environment by mounting the system under /mnt/sysroot in a read-write mode. You can choose not to mount the system, or to mount it in read-only mode. Use this command only once.

Syntax

```
rescue [--nomount|--romount]
```

Options

- **--nomount** or **--romount** - Controls how the installed system is mounted in the rescue environment. By default, the installation program finds your system and mount it in read-write mode, telling you where it has performed this mount. You can optionally select to not mount anything (the **--nomount** option) or mount in read-only mode (the **--romount** option). Only one of these two options can be used.

Notes

To run a rescue mode, make a copy of the Kickstart file, and include the **rescue** command in it. Using the **rescue** command causes the installer to perform the following steps:

1. Run the **%pre** script.
2. Set up the environment for rescue mode.
   The following kickstart commands take effect:
   a. updates
   b. sshpw
   c. logging
   d. lang

   e. network

3. Set up an advanced storage environment.
   The following kickstart commands take effect:
   a. fcoe
   b. iscsi
   c. iscsiname
   d. nvdimm
   e. zfcp
4. Mount the system.

   ```
   rescue [--nomount|--romount]
   ```

5. Run %post script
   This step is run only if the installed system is mounted in read-write mode.
6. Start shell.
7. Reboot system.

CHAPTER 23. BOOT OPTIONS REFERENCE

You can use the boot options to modify the default behavior of the installation program.

23.1. INSTALLATION SOURCE BOOT OPTIONS

You can configure installation source boot options to specify where RHEL installation files are located. These options enable installations from various sources including local media, network servers, and additional repositories, providing flexibility in deployment scenarios.

inst.repo=

The **inst.repo=** boot option specifies the installation source, that is, the location providing the package repositories and a valid **.treeinfo** file that describes them. For example: **inst.repo=cdrom**. The target of the **inst.repo=** option must be one of the following installation media:

- an installable tree, which is a directory structure containing the installation program images, packages, and repository data as well as a valid **.treeinfo** file
- a DVD (a physical disk present in the system DVD drive)
- an ISO image of the full Red Hat Enterprise Linux installation DVD, placed on a disk or a network location accessible to the system.
  Use the **inst.repo=** boot option to configure different installation methods by using different formats. The following table contains details of the **inst.repo=** boot option syntax:

Table 23.1. Types and format for the inst.repo= boot option and installation source

Source type	Boot option format	Source format
CD/DVD drive	**inst.repo=cdrom:*<device>***	Installation DVD as a physical disk. [a]
Mountable device (HDD and USB stick)	**inst.repo=hd:*<device>*:/*<path>***	Image file of the installation DVD.
NFS Server	**inst.repo=nfs:[*options*:]*<server>*:/*<path>***	Image file of the installation DVD, or an installation tree, which is a complete copy of the directories and files on the installation DVD. [b]
HTTP Server	**inst.repo=http://*<host>*/*<path>***	Installation tree that is a complete copy of the directories and files on the installation DVD.
HTTPS Server	**inst.repo=https://*<host>*/*<path>***	
FTP Server	**inst.repo=ftp://*<username>*:*<password>*@*<host>*/*<path>***	

Source type	Boot option format	Source format
HMC	**inst.repo=hmc**	

[a] If *device* is left out, installation program automatically searches for a drive containing the installation DVD.

[b] The NFS Server option uses NFS protocol version 3 by default. To use a different version, add **nfsvers=*X*** to *options*, replacing *X* with the version number that you want to use.

Set disk device names with the following formats:

- Kernel device name, for example **/dev/sda1** or **sdb2**
- File system label, for example **LABEL=Flash** or **LABEL=RHEL8**
- File system UUID, for example **UUID=8176c7bf-04ff-403a-a832-9557f94e61db**
  Non-alphanumeric characters must be represented as **\xNN**, where *NN* is the hexadecimal representation of the character. For example, **\x20** is a white space **(" ")**.

inst.addrepo=

Use the **inst.addrepo=** boot option to add an additional repository that you can use as another installation source along with the main repository (**inst.repo=**). You can use the **inst.addrepo=** boot option multiple times during one boot. The following table contains details of the **inst.addrepo=** boot option syntax.

NOTE

The **REPO_NAME** is the name of the repository and is required in the installation process. These repositories are only used during the installation process; they are not installed on the installed system.

Table 23.2. Installation sources and boot option format

Installation source	Boot option format	Additional information
Installable tree at a URL	**inst.addrepo=REPO_NAME,[http,https,ftp]://*<host>*/*<path>***	Looks for the installable tree at a given URL.
Installable tree at an NFS path	**inst.addrepo=REPO_NAME,nfs://*<server>*:/*<path>***	Looks for the installable tree at a given NFS path. A colon is required after the host. The installation program passes everything after **nfs://** directly to the mount command instead of parsing URLs according to RFC 2224.

Installation source	Boot option format	Additional information
Installable tree in the installation environment	**inst.addrepo=REPO_NAME,file://*<path>***	Looks for the installable tree at the given location in the installation environment. To use this option, the repository must be mounted before the installation program attempts to load the available software groups. The benefit of this option is that you can have multiple repositories on one bootable ISO, and you can install both the main repository and additional repositories from the ISO. The path to the additional repositories is **/run/install/source/REPO_ISO_PATH**. Additionally, you can mount the repository directory in the **%pre** section in the Kickstart file. The path must be absolute and start with /, for example **inst.addrepo=REPO_NAME,file:///*<path>***
Disk	**inst.addrepo=REPO_NAME,hd:*<device>*:*<path>***	Mounts the given *<device>* partition and installs from the ISO that is specified by the *<path>*. If the *<path>* is not specified, the installation program looks for a valid installation ISO on the *<device>*. This installation method requires an ISO with a valid installable tree.

inst.stage2=

The **inst.stage2=** boot option specifies the location of the installation program’s runtime image. This option expects the path to a directory that contains a valid **.treeinfo** file and reads the runtime image location from the **.treeinfo** file. If the **.treeinfo** file is not available, the installation program attempts to load the image from **images/install.img**.
When you do not specify the **inst.stage2** option, the installation program attempts to use the location specified with the **inst.repo** option.

You can use this option when you want to manually specify the installation source in the installation program at a later time. For example, when you want to select the Content Delivery Network (CDN) as an installation source. The installation DVD and Boot ISO already contain a suitable **inst.stage2** option to boot the installation program from the respective ISO.

If you want to specify an installation source, use the **inst.repo=** option instead.

> **NOTE**
>
> By default, the **inst.stage2=** boot option is used on the installation media and is set to a specific label; for example, **inst.stage2=hd:LABEL=RHEL-x-0-0-BaseOS-x86_64**. If you modify the default label of the file system that contains the runtime image, or if you use a customized procedure to boot the installation system, verify that the **inst.stage2=** boot option is set to the correct value.

inst.noverifyssl

Use the **inst.noverifyssl** boot option to prevent the installer from verifying SSL certificates for all HTTPS connections with the exception of additional Kickstart repositories, where **--noverifyssl** can be set per repository.
For example, if your remote installation source is using self-signed SSL certificates, the **inst.noverifyssl** boot option enables the installer to complete the installation without verifying the SSL certificates. For example, when specifying the source by using **inst.stage2=**:

```
inst.stage2=https://hostname/path_to_install_image/ inst.noverifyssl
```

And for example when specifying the source by using **inst.repo=**:

```
inst.repo=https://hostname/path_to_install_repository/ inst.noverifyssl
```

inst.stage2.all

Use the **inst.stage2.all** boot option to specify several HTTP, HTTPS, or FTP sources. You can use the **inst.stage2=** boot option multiple times with the **inst.stage2.all** option to fetch the image from the sources sequentially until one succeeds. For example:

```
inst.stage2.all
inst.stage2=http://hostname1/path_to_install_tree/
inst.stage2=http://hostname2/path_to_install_tree/
inst.stage2=http://hostname3/path_to_install_tree/
```

inst.dd=

The **inst.dd=** boot option is used to perform a driver update during the installation. For more information about how to update drivers during installation, see the *Updating drivers during installation*.

inst.repo=hmc

This option eliminates the requirement of an external network setup and expands the installation options. When booting from a Binary DVD, the installation program prompts you to enter additional kernel parameters. To set the DVD as an installation source, append the **inst.repo=hmc** option to the kernel parameters. The installation program then enables support element (SE) and hardware management console (HMC) file access, fetches the images for stage2 from the DVD, and provides access to the packages on the DVD for software selection.

IMPORTANT

To use the **inst.repo** boot option, ensure the user is configured with a **minimum of Privilege Class B**. For more information about the user configuration, see IBM documentation.

inst.proxy=

This boot option is used when performing an installation from a HTTP, HTTPS, and FTP protocol in the following form:

```
[PROTOCOL://][USERNAME[:PASSWORD]@]HOST[:PORT]
For example:
http://proxyuser:proxypassword@10.1.2.3:3128
```

inst.nosave=

Use the **inst.nosave=** boot option to control the installation logs and related files that are not saved to the installed system, for example **input_ks**, **output_ks**, **all_ks**, **logs** and **all**. You can combine multiple values separated by a comma. For example,

```
inst.nosave=input_ks,logs
```

NOTE

The **inst.nosave** boot option is used for excluding files from the installed system that cannot be removed by a Kickstart %post script, such as logs and input/output Kickstart results.

input_ks

Disables the ability to save the input Kickstart results.

output_ks

Disables the ability to save the output Kickstart results generated by the installation program.

all_ks

Disables the ability to save the input and output Kickstart results.

logs

Disables the ability to save all installation logs.

all

Disables the ability to save all Kickstart results, and all logs.

inst.multilib

Use the **inst.multilib** boot option to set DNF's **multilib_policy** to **all**, instead of **best**.

inst.memcheck

The **inst.memcheck** boot option performs a check to verify that the system has enough RAM to complete the installation. If there is not enough RAM, the installation process is stopped. The system check is approximate and memory usage during installation depends on the package selection, user interface, for example graphical or text, and other parameters.

inst.nomemcheck

The **inst.nomemcheck** boot option does not perform a check to verify if the system has enough RAM to complete the installation. Any attempt to perform the installation with less than the minimum amount of memory is unsupported, and might result in the installation process failing.

23.2. NETWORK BOOT OPTIONS

You can configure network boot options to enable RHEL installation over network connections. These options allow you to set up network interfaces, specify IP addresses, configure bonding, bridging, and VLANs for network-based installations and remote system deployment.

+

NOTE

Initialize the network with the **dracut** tool. For a complete list of **dracut** options, see the **dracut.cmdline(7)** man page on your system.

ip=

Use the **ip=** boot option to configure one or more network interfaces. To configure multiple interfaces, use one of the following methods

- use the **ip** option multiple times, once for each interface; to do so, use the **rd.neednet=1** option, and specify a primary boot interface using the **bootdev** option.
- use the **ip** option once, and then use Kickstart to set up further interfaces. While the configuration with ip= option is applied early in the installation process, the configuration defined in kickstart is applied at a later stage of installation, after the start of the installation program.
  This option accepts several different formats. The following tables contain information about the most common options. In the following tables:
- The **ip** parameter specifies the client IP address and **IPv6** requires square brackets, for example 192.0.2.1 or [2001:db8::99].
- The **gateway** parameter is the default gateway. **IPv6** requires square brackets.
- The **netmask** parameter is the netmask to be used. This can be either a full netmask (for example, 255.255.255.0) or a prefix (for example, 64).
- The **hostname** parameter is the host name of the client system. This parameter is optional.

Table 23.3. Boot option formats to configure the network interface

Boot option format	Configuration method
ip=*method*	Automatic configuration of all interfaces.
ip=*interface:method*	Automatic configuration of a specific interface

Boot option format	Configuration method
ip=*ip::gateway:netmask:hostname:interface*:none	Static configuration, for example, IPv4: **ip=192.0.2.1::192.0.2.254:255.255.255.0:server.example.com:enp1s0:none** IPv6: **ip=[2001:db8::1]::[2001:db8::fffe]:64:server.example.com:enp1s0:none**
ip=*ip::gateway:netmask:hostname:interface:method:mtu*	Automatic configuration of a specific interface with an override

Configuration methods for the automatic interface

The method **automatic configuration of a specific interface with an override** opens the interface by using the specified method of automatic configuration, such as **dhcp**, but overrides the automatically obtained IP address, gateway, netmask, host name or other specified parameters. All parameters are optional, so specify only the parameters that you want to override. For the values of the **method** parameter, see the **dracut.cmdline(7)** man page on your system.

NOTE

- If you use a boot option that requires network access, such as **inst.ks=http://host/path**, without specifying the **ip** option, the default value of the **ip** option is **ip=dhcp**..
- To connect to an iSCSI target automatically, activate a network device for accessing the target by using the **ip=ibft** boot option.

- **nameserver=** - The **nameserver=** option specifies the address of the name server. You can use this option multiple times.

  NOTE

  The **ip=** parameter requires square brackets. However, an IPv6 address does not work with square brackets. An example of the correct syntax to use for an IPv6 address is **nameserver=2001:db8::1**.

- **bootdev=** - The **bootdev=** option specifies the boot interface. This option is mandatory if you use more than one **ip** option.
- **ifname=** - The **ifname=** options assigns an interface name to a network device with a given MAC address. You can use this option multiple times. The syntax is **ifname=interface:MAC**. For example:

```
ifname=eth0:01:23:45:67:89:ab
```

NOTE

The **ifname=** option is the only supported way to set custom network interface names during installation.

- **inst.dhcpclass=** - The **inst.dhcpclass=** option specifies the DHCP vendor class identifier. The **dhcpd** service recognizes this value as **vendor-class-identifier**. The default value is **anaconda-$(uname -srm)**. To ensure the **inst.dhcpclass=** option is applied correctly, request network activation during the early stage of installation by also adding the **ip** option.
- **inst.waitfornet=** - Using the **inst.waitfornet=SECONDS** boot option causes the installation system to wait for network connectivity before installation. The value given in the **SECONDS** argument specifies the maximum amount of time to wait for network connectivity before timing out and continuing the installation process even if network connectivity is not present.
- **vlan=**- Use the **vlan=** option to configure a Virtual LAN (VLAN) device on a specified interface with a given name. The syntax is **vlan=name:interface**. For example:

  ```
  vlan=vlan5:enp0s1
  ```

  This configures a VLAN device named **vlan5** on the **enp0s1** interface. The name can take the following forms:

  - VLAN_PLUS_VID: **vlan0005**
  - VLAN_PLUS_VID_NO_PAD: **vlan5**
  - DEV_PLUS_VID: **enp0s1.0005**
  - DEV_PLUS_VID_NO_PAD: **enp0s1.5**
- **bond=** - Use the **bond=** option to configure a bonding device with the following syntax: **bond=name[:interfaces][:options]**. Replace *name* with the bonding device name, *interfaces* with a comma-separated list of physical (Ethernet) interfaces, and *options* with a comma-separated list of bonding options. For example:

  ```
  bond=bond0:enp0s1,enp0s2:mode=active-backup,tx_queues=32,downdelay=5000
  ```

  For a list of available options, execute the **modinfo bonding** command.
- **bridge=** - Use the **bridge=** option to configure a bridge device with the following syntax: **bridge=name:interfaces**. Replace *name* with the desired name of the bridge device and *interfaces* with a comma-separated list of physical (Ethernet) devices to be used as underlying interfaces in the bridge device. For example:

  ```
  bridge=bridge0:enp0s1,enp0s2
  ```

Additional resources

- Configuring and managing networking

23.3. CONSOLE BOOT OPTIONS

You can configure boot options for your console, monitor display, and keyboard to customize the installation process.

console=

Use the **console=** option to specify a device that you want to use as the primary console. For example, to use a console on the first serial port, use **console=ttyS0**. When using the **console=** argument, the installation starts with a text UI. If you must use the **console=** option multiple times, the boot message is displayed on all specified consoles. However, the installation program uses only the last specified console. For example, if you specify **console=ttyS0 console=ttyS1**, the installation program uses **ttyS1**.

inst.lang=

Use the **inst.lang=** option to set the language that you want to use during the installation. To view the list of locales, enter the command **locale -a | grep _** or the **localectl list-locales | grep _** command.

inst.geoloc=

Use the **inst.geoloc=** option to configure geolocation usage in the installation program. Geolocation is used to preset the language and time zone, and uses the following syntax: **inst.geoloc=value**. The **value** can be any of the following parameters:

- Disable geolocation: **inst.geoloc=0**
- Use the Fedora GeoIP API: **inst.geoloc=provider_fedora_geoip**. This option is deprecated.
- Use the Hostip.info GeoIP API: **inst.geoloc=provider_hostip**. This option is deprecated.

inst.keymap=

Use the **inst.keymap=** option to specify the keyboard layout to use for the installation.

inst.cmdline

Use the **inst.cmdline** option to force the installation program to run in command-line mode. This mode does not allow any interaction, and you must specify all options in a Kickstart file or on the command line.

inst.graphical

Use the **inst.graphical** option to force the installation program to run in graphical mode. The graphical mode is the default.

inst.text

Use the **inst.text** option to force the installation program to run in text mode instead of graphical mode.

inst.noninteractive

Use the **inst.noninteractive** boot option to run the installation program in a non-interactive mode. User interaction is not permitted in the non-interactive mode, and **inst.noninteractive** you can use the **inst.nointeractive** option with a graphical or text installation. When you use the **inst.noninteractive** option in text mode, it behaves the same as the **inst.cmdline** option.

NOTE

It only makes sense to use the **inst.noninteractive** option when performing a Kickstart installation.

inst.resolution=

Use the **inst.resolution=** option to specify the screen resolution in graphical mode. The format is **NxM**, where *N* is the screen width and *M* is the screen height (in pixels). The recommended resolution is 1024x768.

inst.rdp

Use the **inst.rdp** option to run the graphical installation using remote desktop protocol. If no RDP username (by using **inst.rdpuser=**) or password (by using **inst.rdp.password=**) are specified, the installation program asks users to provide interactively. This option only applies if used together with the **inst.rdp** option.

inst.rdp.password=

Use the **inst.rdp.password=** option to set a password on the RDP server that is used by the installation program.

modprobe.blacklist=

Use the **modprobe.blacklist=** option to blocklist or completely disable one or more drivers. Drivers (modules) that you disable using this option cannot load when the installation starts. After the installation finishes, the installed system retains these settings. You can find a list of the blocklisted drivers in the **/etc/modprobe.d/** directory. Use a comma-separated list to disable multiple drivers. For example:

```
modprobe.blacklist=ahci,firewire_ohci
```

NOTE

You can use **modprobe.blacklist** in combination with the different command line options. For example, use it with the **inst.dd** option to ensure that an updated version of an existing driver is loaded from a driver update disc:

```
modprobe.blacklist=virtio_blk
```

inst.sshd

Use the **inst.sshd** option to start the **sshd** service during installation, so that you can connect to the system during the installation using SSH, and monitor the installation progress. For more information about SSH, see the **ssh(1)** man page on your system. By default, the **sshd** service is automatically started only on the 64-bit IBM Z architecture. On other architectures, **sshd** is not started unless you use the **inst.sshd** option.

NOTE

During installation, the root account has no password by default. You can set a root password during installation with the **sshpw** Kickstart command.

inst.kdump_addon=

Use the **inst.kdump_addon=** option to enable or disable the Kdump configuration screen (add-on) in the installation program. This screen is enabled by default; use **inst.kdump_addon=off** to disable it. Disabling the add-on disables the Kdump screens in both the graphical and text-based interface as well as the **%addon com_redhat_kdump** Kickstart command.

23.4. DEBUG BOOT OPTIONS

You can use the following options when debugging issues to troubleshoot and fix issues.

inst.rescue

Use the **inst.rescue** option to run the rescue environment for diagnosing and fixing systems. For more information, see the Red Hat Knowledgebase solution repair a filesystem in rescue mode .

inst.updates=

Use the **inst.updates=** option to specify the location of the **updates.img** file that you want to apply during installation. The **updates.img** file can be derived from one of several sources.

Table 23.4. **updates.img** file sources

Source	Description	Example
Updates from a network	Specify the network location of **updates.img**. This does not require any modification to the installation tree. To use this method, edit the kernel command line to include **inst.updates**.	**inst.updates=http://website.com/path/to/updates.img**.
Updates from a disk image	Save an **updates.img** on a USB key.	**inst.updates=sda1:/images/updates.img** for an updates image located in the **images** directory on the **sda1** device or **inst.updates=UUID=b4234403-dafb-44c1-b878-4d57b40c9843:/updates.img** for an update image located in the root directory of a partition identified by its UUID.
Updates from an installation tree	If you are using a CD, disk, HTTP, HTTPS, or FTP installation, save the **updates.img** in the installation tree so that all installations can detect the **.img** file. The file name must be **updates.img**.	For NFS installs, save the file in the **images/** directory, or in the **RHupdates/** directory.

inst.syslog=

Sends log messages to the **syslog** process on the specified host when the installation starts. You can use **inst.syslog=<host>[:port]** only if the remote **syslog** process is configured to accept incoming connections.

inst.virtiolog=

Use the **inst.virtiolog=<name>** option to specify which virtio port (a character device at **/dev/virtio-ports/<name>**) to use for forwarding logs. The default value is **org.fedoraproject.anaconda.log.0**.

rd.live.ram

Copies the stage 2 image in **images/install.img** into RAM. Note that this increases the memory required for installation by the size of the image, which may result in an additional requirement of approximately 1 GiB of RAM or more.

inst.nokill

Prevent the installation program from rebooting when a fatal error occurs, or at the end of the installation process. Use it capture installation logs which would be otherwise lost upon reboot.

inst.noshell

Prevent a shell on terminal session 2 (tty2) including tmux window 2, during installation.

inst.notmux

Prevent the use of tmux during installation. The output is generated without terminal control characters and is meant for non-interactive uses.

23.5. STORAGE BOOT OPTIONS

You can specify the following options to customize booting from a storage device.

inst.nompath

Disables support for multipath devices. Use this option only if your system has a false-positive that incorrectly identifies a normal block device as a multipath device.

WARNING

Use this option with caution. Do not use this option with multipath hardware. Using this option to install to a single path of a multipath device is not supported.

inst.gpt

Prefer creation of GPT disk labels. This option is deprecated and will be removed in future releases. Use **inst.disklabel=gpt** instead.

inst.disklabel=

Prefer creation of the specified disk label type. Specify **gpt** to prefer creation of GPT disk labels (default). Specify **mbr** to prefer creation of MBR disk labels if supported.

inst.wait_for_disks=

Use the **inst.wait_for_disks=** option to specify the number of seconds installation program to wait for disk devices to appear at the beginning of the installation. Use this option when you use the **OEMDRV-labeled** device to automatically load the Kickstart file or the kernel drivers but the device takes a longer time to appear during the boot process. By default, the installation program waits for **5** seconds. Use **0** seconds to minimize the delay.

inst.nonibftiscsiboot

The **inst.nonibftiscsiboot** option enables placement of the boot loader on **iSCSI** devices that are not configured through the **iSCSI** boot firmware table (**iBFT**).