Red Hat Enterprise Linux for Real Time 9

Installing RHEL 9 for Real Time

Installing the RHEL for Real Time kernel on Red Hat Enterprise Linux

Last Updated: 2024-09-23

Red Hat Enterprise Linux for Real Time 9 Installing RHEL 9 for Real Time

Installing the RHEL for Real Time kernel on Red Hat Enterprise Linux

Legal Notice

Abstract

Install the RHEL for Real Time kernel on Red Hat Enterprise Linux to obtain low latency and make response times predictable. Learn how to install the real-time kernel, perform post-installation tasks, and configure a kernel variant, such as the real-time, stock, or the debug kernel to boot with.

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. INSTALLING RHEL FOR REAL TIME

Many industries and organizations need extremely high performance computing and may require low and predictable latency, especially in the financial and telecommunications industries. Latency, or response time, is defined as the time between an event and system response and is generally measured in microseconds (µs).

For most applications running under a Linux environment, basic performance tuning can improve latency sufficiently. For those industries where latency not only needs to be low, but also accountable and predictable, Red Hat developed a 'drop-in' kernel replacement that provides this. RHEL for Real Time is distributed as part of RHEL 9 and provides seamless integration with RHEL 9. RHEL for Real Time offers clients the opportunity to measure, configure, and record latency times within their organization.

WARNING

Before installing RHEL for Real Time, ensure that the base platform is properly tuned and the system BIOS parameters are adjusted. Failure to perform these tasks may prevent getting consistent performance from a RHEL Real Time deployment.

1.1. RHEL FOR REAL TIME FOR OPTIMIZING LATENCY

RHEL for Real Time is designed to be used on well-tuned systems for applications with extremely high determinism requirements. Kernel system tuning offers the vast majority of the improvement in determinism.

For example, in many workloads, thorough system tuning improves consistency of results by around 90%. This is why, before using RHEL for Real Time, we recommend that customers first perform system tuning of standard RHEL to see if it meets their objectives.

System tuning is just as important when using the Real Time kernel as it is for the standard kernel. Installing the Real Time kernel on an untuned system running the standard kernel supplied as part of RHEL is not likely to result in any noticeable benefit. Tuning the standard kernel will yield 90% of the possible latency gains. The Real Time kernel provides the last 10% of latency reduction required by the most demanding workloads.

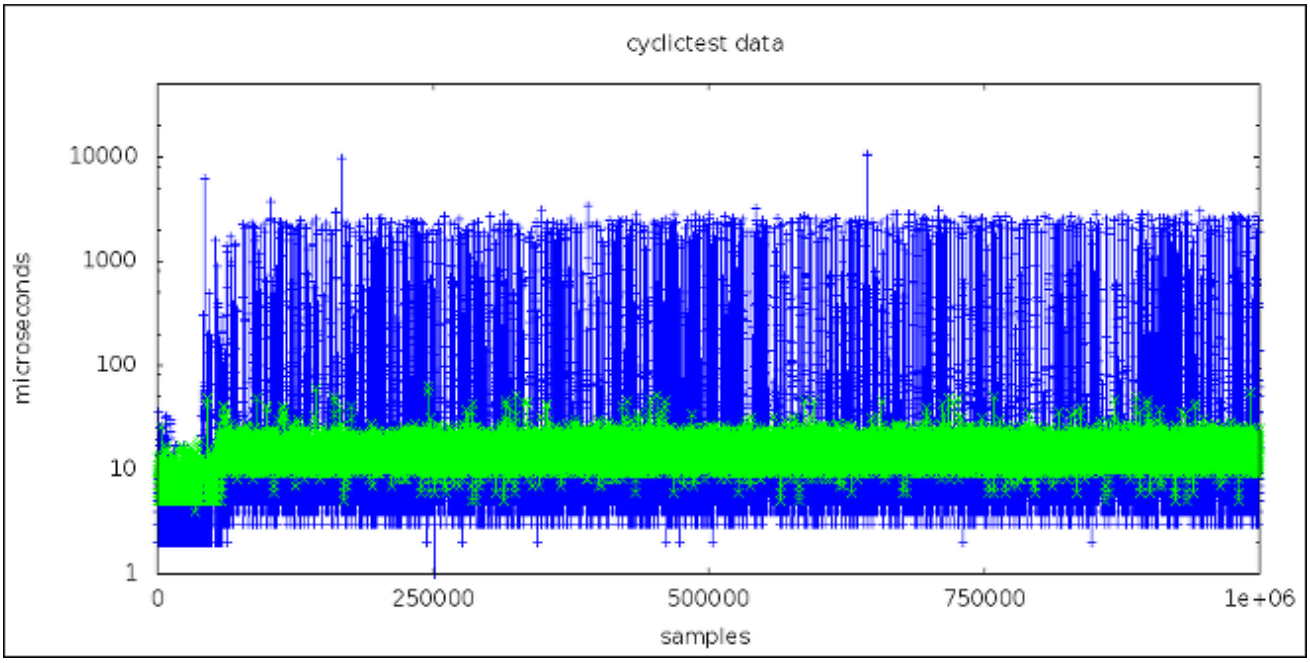
WARNING

Before tuning Real Time kernel systems, ensure that the base platform is properly tuned and the system BIOS parameters are adjusted. Failure to perform these tasks may prevent getting consistent performance from a RHEL Real Time deployment.

The objective of the Real Time kernel consistent, low-latency determinism offering predictable response times. There is some additional kernel overhead associated with the real time kernel. This is due primarily to handling hardware interrupts in separately scheduled threads. The increased overhead

in some workloads results in some degradation in overall throughput. The exact amount is very workload dependent, ranging from 0% to 30%.

For typical workloads with kernel latency requirements in the millisecond (ms) range, the standard RHEL kernel is sufficient. However,if your workload has stringent low-latency determinism requirements for core kernel features such as interrupt handling and process scheduling in the microsecond (µs) range, then the Real Time kernel is for you.



This graph compares a million samples of machines that use the RHEL and the RHEL for Real Time kernel, respectively.

- The blue points in this graph represent the system response time (in microseconds) of machines running a tuned RHEL kernel.
- The green points in the graph represent the system response time of machines running a tuned real-time kernel.

It is clear from this graph that the response time of the Real Time kernel is very consistent, in contrast to the standard kernel, which has greater variability.

Additional resources

- Optimizing RHEL 8 for Real Time for low latency operation

1.2. INSTALLING RHEL FOR REAL TIME USING DNF

In addition to installing the real-time kernel with **dnf**, an ISO image containing RHEL for Real Time is also available for download from the Download Red Hat Enterprise Linux portal. You can use this ISO image to obtain all the RPM packages included with RHEL for Real Time. However, because this is not a bootable ISO image, you cannot use it to create a bootable USB or CD medium.

Prerequisites

- The latest version of RHEL 9 is installed on an AMD64 or Intel64 system. The real-time kernel runs on AMD64 and Intel 64 (also known as x86_64) server platforms that are certified to run Red Hat Enterprise Linux.

- Your machine is registered and RHEL is attached to a RHEL for Real Time subscription.
- Ensure that the base platform is properly tuned and the system BIOS parameters are adjusted.

WARNING

Failure to perform the prerequisite tasks before installing the real-time kernel might prevent a consistent performance from a RHEL for Real Time kernel deployment.

Procedure

1. Enable the RHEL for Real Time repository.

```
# subscription-manager repos --enable rhel-9-for-x86_64-rt-rpms
```

2. Install the RHEL for Real Time package group.

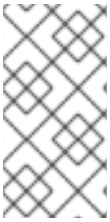
```
# dnf groupinstall RT
```

This group installs several packages:

- **kernel-rt** includes the RHEL for Real Time kernel package.
- **kernel-rt-core** includes the core RHEL for Real Time kernel package.
- **kernel-rt-devel** includes the RHEL for Real Time kernel development package.
- **kernel-rt-modules** includes the RHEL for Real Time kernel modules package.
- **kernel-rt-modules-extra** includes the RHEL for Real Time kernel extra modules package.
- **realtime-setup** sets up the basic environment required by RHEL for Real Time.
- **rteval** evaluates system suitability for RHEL for Real Time.
- **rteval-loads** provides source code for **rteval** loads.
- **tuned-profiles-realtime** includes the additional **TuneD** profiles targeted to real-time.

3. Optional: Additionally, the **tuna** package contains a tool that helps tune the real-time kernel workload, greatly automating CPU isolation and thread affinity operations from the command line or the GUI. This package is available in the base RHEL 9 repository.

```
# dnf install tuna
```



NOTE

When the RHEL for Real Time kernel is installed, it is automatically set to be the default kernel and is used on the next boot. You can also configure other existing kernels variants, such as, **kernel**, **kernel-debug**, or **kernel-rt-debug** as the default boot kernel. For more information, see Configuring kernel-rt as the default boot kernel .

Verification

- Check the installation location and verify that the components have been installed successfully.

```
# rpm -ql realtime-setup
/etc/security/limits.d/realtime.conf
/etc/sysconfig/realtime-setup
/etc/udev/rules.d/99-rhel-rt.rules
/usr/bin/realtime-setup
/usr/bin/rt-setup-kdump
/usr/bin/slub_cpu_partial_off
/usr/lib/.build-id
/usr/lib/.build-id/a4
/usr/lib/.build-id/a4/da77908aa4c6f048939f3267f1c552c456d117
/usr/lib/systemd/system/rt-entsk.service
/usr/lib/systemd/system/rt-setup.service
/usr/sbin/kernel-is-rt
/usr/sbin/rt-entsk
```

Additional resources

- Can KVM guests be run on real-time(RT) kernel?

1.3. AVAILABLE RPM PACKAGES IN THE RHEL FOR REAL TIME REPOSITORY

The Red Hat Package Manager (RPM) for RHEL for Real Time repository includes the following packages:

- **kernel-rt** package, which is the RHEL for Real Time kernel package.
- RHEL for Real Time kernel test packages, which contains test programs for the real-time kernel.
- RHEL for Real Time debugging packages, which are for debugging and code tracing.

Table 1.1. Basic RHEL for Real Time kernel packages

RPM package name	Description	RT-specific	Required
kernel-rt	Low latency and preemption functionality	Yes	Yes

Table 1.2. RHEL for Real Time kernel test packages

RPM package name	Description	RT-specific	Required
kernel-rt-devel	Headers and libraries for kernel development	Yes	No
kernel-rt-debug	RHEL for Real Time kernel with debugging functions compiled in (slow)	Yes	No
kernel-rt-debug-devel	Headers and libraries for development on debug kernel	Yes	No
realtime-tests	Utilities for measuring system latencies and for proving that priority-inheritance mutex functions properly	No	No

The debugging packages are provided to use with the **perf**, **trace-cmd**, and **crash** utilities for analyzing kernel crash dumps. The debugging packages include symbol tables and are quite large. For this reason, the debugging packages are separately delivered from the other RHEL for Real Time packages. You can download the debugging packages from **RHEL for Real Time - Debug RPMs** repository.

Table 1.3. Basic RHEL for Real Time debugging packages

RPM package name	Description	RT-specific	Required
kernel-rt-debuginfo	Symbols for profiling and debugging use, such as **perf** or **trace-cmd**	Yes	No
kernel-rt-debug-debuginfo	Symbols for profiling and tracing	Yes	No

1.4. POST INSTALLATION INSTRUCTIONS

After you install the real-time kernel ensure that:

- To achieve optimal low-latency determinism, you perform RHEL for Real Time specific system tuning.
- You know about the module compatibility of the real-time kernel and the standard kernel.
- To enable **kdump**, you must configure RHEL for Real Time to provide crash dump information by enabling **kexec/kdump**.
- Verify that the Real Time kernel is the default kernel.

Module compatibility of the real-time kernel and the standard kernel

The real-time kernel differs substantially from the standard Red Hat Enterprise Linux 9 kernel. As a consequence, third-party kernel modules are incompatible with RHEL for Real Time.

Kernel modules are inherently specific to the kernel they are built for. The real time kernel is substantially different from the standard kernel, and so are the modules. Therefore, you cannot take third-party modules from Red Hat Enterprise Linux 9 and use them as-is on the real-time kernel.

If you must use a third-party module, you must recompile it with the RHEL for Real Time header files, which are available in the RHEL for Real Time development and test packages.

The third-party drivers for the standard Red Hat Enterprise Linux 9 but do not currently have a custom build for RHEL for Real Time are:

- EMC Powerpath
- NVidia graphics
- Advanced storage adapter configuration utilities from Qlogic

The user space **syscall** interface is compatible with RHEL for Real Time.

Additional resources

- Specifying the RHEL kernel to run
- Improving latency using the tuna CLI

CHAPTER 2. SPECIFYING THE RHEL KERNEL TO RUN

You can boot any installed kernel, standard or Real Time by selecting the required kernel manually in the GRUB menu during booting. You can also configure the kernel to boot by default.

When the RHEL for Real Time kernel is installed, it is automatically set to be the default kernel and is used on the next boot.

2.1. DISPLAYING THE DEFAULT KERNEL

You can display the kernel configured to boot by default.

Procedure

- To view the default kernel:

```
# grubby --default-kernel

/boot/vmlinuz-kernel-rt-5.14.0-70.13.1.rt21.83.el9_0
```

The **rt** in the output of the command shows that the default kernel is a real time kernel.

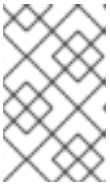
2.2. DISPLAYING THE RUNNING KERNEL

You can display the currently running kernel

Procedure

- To show which kernel the system is currently running.

```
~]# uname -a
Linux rt-server.example.com 4.18.0-80.rt9.138.el8.x86_64 …
```



NOTE

When the system receives a minor update, for example, from 8.3 to 8.4, the default kernel might automatically change from the Real Time kernel back to the standard kernel.

2.3. CONFIGURING KERNEL-RT AS THE DEFAULT BOOT KERNEL

On a newly installed system, the stock RHEL **kernel** is set as the default boot kernel and is used as the default kernel on the next boot and subsequent system updates. You can change this configuration and set **kernel-rt** as the default kernel to boot with and also make this configuration persistent across the system updates. Configuring **kernel-rt** is a one-time procedure, which you can change or revert to another kernel if necessary. You can also configure other existing kernels variants, such as, **kernel**, **kernel-debug**, or **kernel-rt-debug**, as the default boot kernel.

Procedure

1. To configure **kernel-rt** as the default boot kernel, enter the following command:

```
# grubby --set-default=<RT_VMLINUZ>
```

RT_VMLINUZ is the name of the **vmlinux** file that is associated with the **kernel-rt** kernel. For example:

```
# grubby --set-default=/boot/vmlinuz-5.14.0-284.11.1.rt14.296.el9_2.x86_64+rt
```

2. To configure **kernel-rt** as default boot kernel on system updates, enter the following command:

```
# sed -i ‘s/UPDATEDEFAULT=.*/UPDATEDEFAULT=yes/g’/etc/sysconfig/kernel
# sed -i 's/DEFAULTKERNEL=.*/DEFAULTKERNEL=kernel-rt-core/g'/etc/sysconfig/kernel
```

The **UPDATEDEFAULT** variable when specified as **yes**, sets the default kernel to change with system updates.

In the example output, the path for the default kernel is specific to the **kernel-rt-core** package installed. You can determine the path to the kernel from a package by using the **rpm -q kernel-rt-core** command.

a. Optional: If you need to determine the path to the kernel from a package, first list the installed packages:

```
# rpm -q kernel-rt-core
  kernel-rt-core-5.14.0-284.11.1.rt14.296.el9_2.x86_64
  kernel-rt-core-5.14.0-284.10.1.rt14.295.el9_2.x86_64
  kernel-rt-core-5.14.0-284.9.1.rt14.294.el9_2.x86_64
```

b. To use the latest installed package as the default, enter the following command to find the path to the boot image from that package:

```
# rpm -ql kernel-rt-core-5.14.0-284.11.1.rt14.296.el9_2.x86_64|grep‘^/boot/vmlinu’
/boot/vmlinuz-5.14.0-284.11.1.rt14.296.el9_2.x86_64.x86_64+rt
```

c. To configure **kernel-rt** as the default boot kernel, enter the following command:

```
# grubby --set-default=/boot/vmlinuz-5.14.0-284.11.1.rt14.296.el9_2.x86_64.x86_64+rt
```

Verification

- To verify **kernel-rt** is the default kernel, enter the following command:

```
# grubby --default-kernel
/boot/vmlinuz-5.14.0-284.11.1.rt14.296.el9_2.x86_64.x86_64+rt
```

CHAPTER 3. INSTALLING KDUMP

The **kdump** service is installed and activated by default on the new versions of Red Hat Enterprise Linux 9 installations. With the provided information and procedures, learn what **kdump** is and how to install if **kdump** is not enabled by default.

3.1. WHAT IS KDUMP

kdump is a service which provides a crash dumping mechanism and generates a dump file, known as crash dump or a **vmcore** file. The **vmcore** file has the contents of the system memory that helps in analysis and troubleshooting. **kdump** uses the **kexec** system call to boot into the second kernel, a *capture kernel* without a reboot and then captures the contents of the crashed kernel's memory and saves it into a file. The second kernel is available in a reserved part of the system memory.

IMPORTANT

A kernel crash dump can be the only information available if a system failure occur. Therefore, operational **kdump** is important in mission-critical environments. Red Hat advises to regularly update and test **kexec-tools** in your normal kernel update cycle. This is especially important when you install new kernel features.

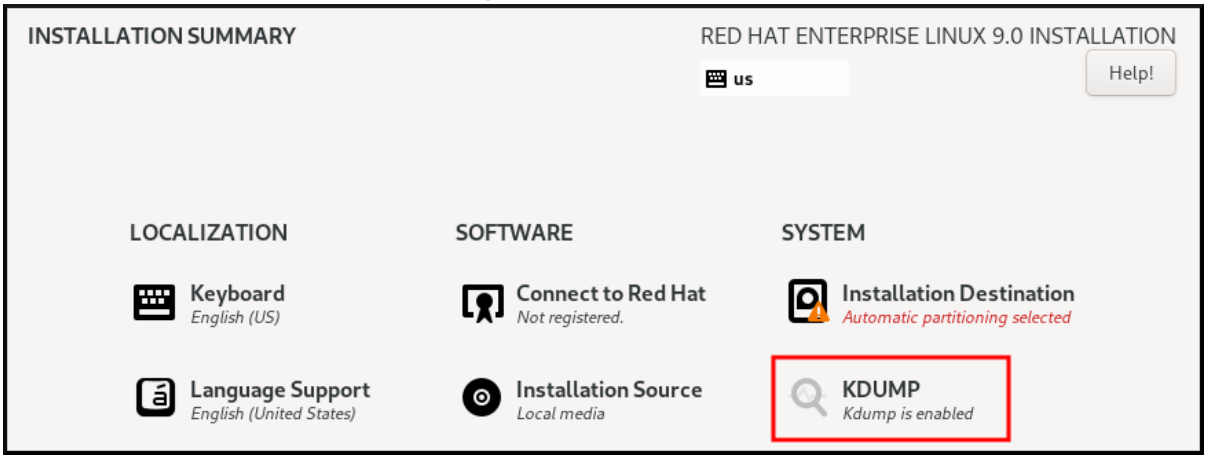
You can enable **kdump** for all installed kernels on a machine or only for specified kernels. This is useful when there are multiple kernels used on a machine, some of which are stable enough that there is no concern that they could crash. When you install **kdump**, a default **/etc/kdump.conf** file is created. The **/etc/kdump.conf** file includes the default minimum **kdump** configuration, which you can edit to customize the **kdump** configuration.

3.2. INSTALLING KDUMP USING ANACONDA

The **Anaconda** installer provides a graphical interface screen for **kdump** configuration during an interactive installation. The installer screen is titled as **KDUMP** and is available from the main **Installation Summary** screen. You can enable **kdump** and reserve the required amount of memory.

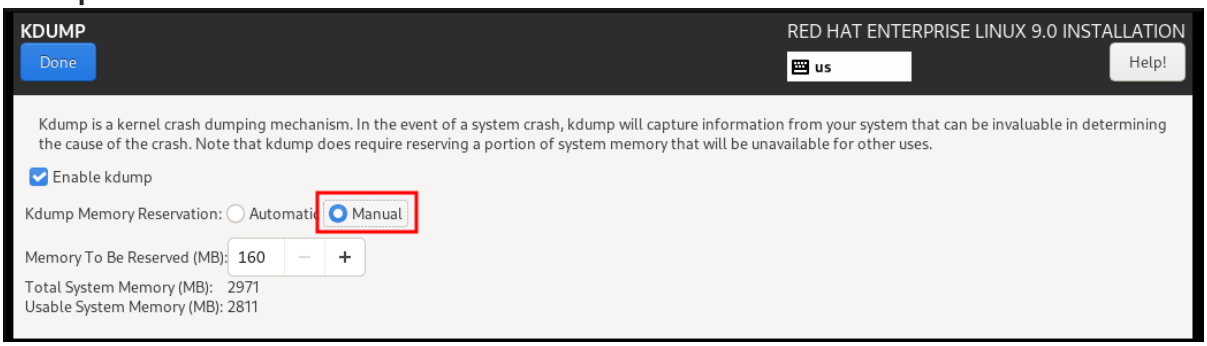
Procedure

1. Under the **KDUMP** field, enable **kdump** if not already enabled.



2. Under **Kdump Memory Reservation**, select **Manual`** if you must customize the memory reserve.

3. Under **KDUMP** field, in **Memory To Be Reserved (MB)**, set the required memory reserve for **kdump**.



3.3. INSTALLING KDUMP ON THE COMMAND LINE

Some installation options, such as custom **Kickstart** installations, in some cases do **not** install or enable **kdump** by default. If this is your case, follow the procedure below.

Prerequisites

- An active RHEL subscription.
- A repository containing the **kexec-tools** package for your system CPU architecture.
- Fulfilled requirements for **kdump** configurations and targets. For details, see Supported kdump configurations and targets.

Procedure

1. Check whether **kdump** is installed on your system:

```
# rpm -q kexec-tools
```

Output if the package is installed:

```
# kexec-tools-2.0.22-13.el9.x86_64
```

Output if the package is not installed:

```
package kexec-tools is not installed
```

2. Install **kdump** and other necessary packages by:

```
# dnf install kexec-tools
```

CHAPTER 4. CONFIGURING KDUMP ON THE COMMAND LINE

The memory for **kdump** is reserved during the system boot. The memory size is configured in the system's Grand Unified Bootloader (GRUB) configuration file. The memory size depends on the **crashkernel=** value specified in the configuration file and the size of the system's physical memory.

4.1. ESTIMATING THE KDUMP SIZE

When planning and building your **kdump** environment, it is important to know how much space the crash dump file requires.

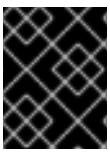
The **makedumpfile --mem-usage** command estimates how much space the crash dump file requires. It generates a memory usage report. The report helps you determine the dump level and which pages are safe to be excluded.

Procedure

- Execute the following command to generate a memory usage report:

```
# makedumpfile --mem-usage /proc/kcore


TYPE        PAGES    EXCLUDABLE    DESCRIPTION
----------------------------------------------------------------
ZERO          501635      yes        Pages filled with zero
CACHE         51657       yes        Cache pages
CACHE_PRIVATE 5442        yes        Cache pages + private
USER          16301       yes        User process pages
FREE          77738211    yes        Free pages
KERN_DATA     1333192     no         Dumpable kernel data
```



IMPORTANT

The **makedumpfile --mem-usage** command reports required memory in pages. This means that you must calculate the size of memory in use against the kernel page size.

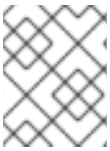
4.2. CONFIGURING KDUMP MEMORY USAGE ON RHEL 9

The **kexec-tools** package maintains the default **crashkernel=** memory reservation values. The **kdump** service uses the default value to reserve the crash kernel memory for each kernel. The default value can also serve as the reference base value to estimate the required memory size when you set the **crashkernel=** value manually. The minimum size of the crash kernel can vary depending on the hardware and machine specifications.

The automatic memory allocation for **kdump** also varies based on the system hardware architecture and available memory size. For example, on AMD and Intel 64-bit architectures, the default value for the **crashkernel=** parameter will work only when the available memory is more than 1 GB. The **kexec-tools** utility configures the following default memory reserves on AMD64 and Intel 64-bit architecture:

```
crashkernel=1G-4G:192M,4G-64G:256M,64G:512M
```

You can also run **kdumpctl estimate** to query a rough estimate value without triggering a crash. The estimated **crashkernel=** value might not be an accurate one but can serve as a reference to set an appropriate **crashkernel=** value.



NOTE

The **crashkernel=auto** option in the boot command line is no longer supported on RHEL 9 and later releases.

Prerequisites

- You have root permissions on the system.
- You have fulfilled **kdump** requirements for configurations and targets. For details, see Supported kdump configurations and targets .
- You have installed the **zipl** utility if it is the IBM Z system.

Procedure

1. Configure the default value for crash kernel.

```
# kdumpctl reset-crashkernel --kernel=ALL
```

When configuring the **crashkernel=** value, test the configuration by rebooting with **kdump** enabled. If the **kdump** kernel fails to boot, increase the memory size gradually to set an acceptable value.

2. To use a custom **crashkernel=** value:

   a. Configure the required memory reserve.

```
crashkernel=192M
```

Alternatively, you can set the amount of reserved memory to a variable depending on the total amount of installed memory using the syntax **crashkernel=*<range1>:<size1>,<range2>:<size2>***. For example:

```
crashkernel=1G-4G:192M,2G-64G:256M
```

The example reserves 192 MB of memory if the total amount of system memory is 1 GB or higher and lower than 4 GB. If the total amount of memory is more than 4 GB, 256 MB is reserved for **kdump**.

   b. Optional: Offset the reserved memory.
Some systems require to reserve memory with a certain fixed offset since crashkernel reservation is very early, and it wants to reserve some area for special usage. If the offset is set, the reserved memory begins there. To offset the reserved memory, use the following syntax:

```
crashkernel=192M@16M
```

The example reserves 192 MB of memory starting at 16 MB (physical address 0x01000000). If you offset to 0 or do not specify a value, **kdump** offsets the reserved memory

automatically. You can also offset memory when setting a variable memory reservation by specifying the offset as the last value. For example, **crashkernel=1G-4G:192M,2G-64G:256M@16M**.

c. Update the bootloader configuration.

```
# grubby --update-kernel ALL --args "crashkernel=<custom-value>"
```

The ***<custom-value>*** must contain the custom **crashkernel=** value that you have configured for the crash kernel.

3. Reboot for changes to take effect.

```
# reboot
```

Verification

Cause the kernel to crash by activating the **sysrq** key. The **address-YYYY-MM-DD-HH:MM:SS/vmcore** file is saved to the target location as specified in the **/etc/kdump.conf** file. If you choose the default target location, the **vmcore** file is saved in the partition mounted under **/var/crash/**.

WARNING

The commands to test **kdump** configuration will cause the kernel to crash with data loss. Follow the instructions with care and do not use an active production system to test the kdump configuration

1. Activate the **sysrq** key to boot into the **kdump** kernel.

```
# echo c > /proc/sysrq-trigger
```

The command causes the kernel to crash and reboots the kernel if required.

2. Display the **/etc/kdump.conf** file and check if the **vmcore** file is saved in the target destination.

Additional resources

- How to manually modify the boot parameter in grub before the system boots
- **grubby(8)** man page on your system

4.3. CONFIGURING THE KDUMP TARGET

The crash dump is usually stored as a file in a local file system, written directly to a device. Alternatively, you can set up for the crash dump to be sent over a network using the **NFS** or **SSH** protocols. Only one of these options to preserve a crash dump file can be set at a time. The default behavior is to store it in the **/var/crash/** directory of the local file system.

Prerequisites

- You have root permissions on the system.
- Fulfilled requirements for **kdump** configurations and targets. For details, see Supported kdump configurations and targets.

Procedure

- To store the crash dump file in **/var/crash/** directory of the local file system, edit the **/etc/kdump.conf** file and specify the path:

```
path /var/crash
```

The option **path /var/crash** represents the path to the file system in which **kdump** saves the crash dump file.

NOTE

- When you specify a dump target in the **/etc/kdump.conf** file, then the path is **relative** to the specified dump target.
- When you do not specify a dump target in the **/etc/kdump.conf** file, then the path represents the **absolute** path from the root directory.

Depending on what is mounted in the current system, the dump target and the adjusted dump path are taken automatically.

To secure the crash dump file and the accompanying files produced by **kdump**, you should set up proper attributes for the target destination directory, such as user permissions and SELinux contexts. Additionally, you can define a script, for example **kdump_post.sh** in the **kdump.conf** file as follows:

```
kdump_post <path_to_kdump_post.sh>
```

The **kdump_post** directive specifies a shell script or a command that is executed **after kdump** has completed capturing and saving a crash dump to the specified destination. You can use this mechanism to extend the functionality of **kdump** to perform actions including the adjustment of file permissions.

Example 4.1. The kdump target configuration

```
# grep -v ^# /etc/kdump.conf | grep -v ^$
ext4 /dev/mapper/vg00-varcrashvol
path /var/crash
core_collector makedumpfile -c --message-level 1 -d 31
```

Here, the dump target is specified (**ext4 /dev/mapper/vg00-varcrashvol**), and thus mounted at **/var/crash**. The **path** option is also set to **/var/crash**, so the **kdump** saves the **vmcore** file in the **/var/crash/var/crash** directory.

- To change the local directory in which the crash dump is to be saved, as **root**, edit the **/etc/kdump.conf** configuration file:
  a. Remove the hash sign (**#**) from the beginning of the **#path /var/crash** line.

b. Replace the value with the intended directory path. For example:

```
path /usr/local/cores
```

IMPORTANT

In Red Hat Enterprise Linux 9, the directory defined as the **kdump** target using the **path** directive must exist when the **kdump systemd** service starts to avoid failures. This behavior is different from versions earlier than RHEL, where the directory is created automatically if it did not exist when the service starts.

- To write the file to a different partition, edit the **/etc/kdump.conf** configuration file:

  a. Remove the hash sign (**#**) from the beginning of the **#ext4** line, depending on your choice.

  - device name (the **#ext4 /dev/vg/lv_kdump** line)
  - file system label (the **#ext4 LABEL=/boot** line)
  - UUID (the **#ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937** line)

  b. Change the file system type and the device name, label or UUID, to the required values. The correct syntax for specifying UUID values is both **UUID="correct-uuid"** and **UUID=correct-uuid**. For example:

```
ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937
```

IMPORTANT

It is recommended to specify storage devices using a **LABEL=** or **UUID=**. Disk device names such as **/dev/sda3** are not guaranteed to be consistent across reboot.

When you use Direct Access Storage Device (DASD) on IBM Z hardware, ensure the dump devices are correctly specified in **/etc/dasd.conf** before you proceed with **kdump**.

- To write the crash dump directly to a device, edit the **/etc/kdump.conf** configuration file:

  a. Remove the hash sign (**#**) from the beginning of the **#raw /dev/vg/lv_kdump** line.

  b. Replace the value with the intended device name. For example:

```
raw /dev/sdb1
```

- To store the crash dump to a remote machine using the **NFS** protocol:

  a. Remove the hash sign (**#**) from the beginning of the **#nfs my.server.com:/export/tmp** line.

  b. Replace the value with a valid hostname and directory path. For example:

```
nfs penguin.example.com:/export/cores
```

c. Restart the **kdump** service for the changes to take effect:

```
sudo systemctl restart kdump.service
```

NOTE

When using the NFS directive to specify the NFS target, **kdump.service** automatically attempts to mount the NFS target to check the disk space. There is no need to mount the NFS target beforehand. To prevent **kdump.service** from mounting the target, use the **dracut_args --mount** directive in **kdump.conf** so that **kdump.service** calls the **dracut** utility with the **--mount** argument to specify the NFS target.

- To store the crash dump to a remote machine using the SSH protocol:

  a. Remove the hash sign (**#**) from the beginning of the **#ssh user@my.server.com** line.

  b. Replace the value with a valid username and hostname.

  c. Include your SSH key in the configuration.

    i. Remove the hash sign from the beginning of the **#sshkey /root/.ssh/kdump_id_rsa** line.

    ii. Change the value to the location of a key valid on the server you are trying to dump to. For example:

```
ssh john@penguin.example.com
sshkey /root/.ssh/mykey
```

Additional resources

Section 4.8, “Files produced by kdump after system crash”

4.4. CONFIGURING THE KDUMP CORE COLLECTOR

The **kdump** service uses a **core_collector** program to capture the crash dump image. In RHEL, the **makedumpfile** utility is the default core collector. It helps shrink the dump file by:

- Compressing the size of a crash dump file and copying only necessary pages using various dump levels.
- Excluding unnecessary crash dump pages.
- Filtering the page types to be included in the crash dump.

Syntax

```
core_collector makedumpfile -l --message-level 1 -d 31
```

Options

- **-c**, **-l** or **-p**: specify compress dump file format by each page using either, **zlib** for **-c** option, **lzo** for **-l** option or **snappy** for **-p** option.

- **-d (dump_level)**: excludes pages so that they are not copied to the dump file.
- **--message-level** : specify the message types. You can restrict outputs printed by specifying **message_level** with this option. For example, specifying 7 as **message_level** prints common messages and error messages. The maximum value of **message_level** is 31

Prerequisites

- You have root permissions on the system.
- Fulfilled requirements for **kdump** configurations and targets. For details, see Supported kdump configurations and targets.

Procedure

1. As **root**, edit the **/etc/kdump.conf** configuration file and remove the hash sign ("#") from the beginning of the **#core_collector makedumpfile -l --message-level 1 -d 31**.
2. To enable crash dump file compression, execute:

```
core_collector makedumpfile -l --message-level 1 -d 31
```

The **-l** option specifies the **dump** compressed file format. The **-d** option specifies dump level as 31. The **--message-level** option specifies message level as 1.

Also, consider following examples with the **-c** and **-p** options:

- To compress a crash dump file using **-c**:

```
core_collector makedumpfile -c -d 31 --message-level 1
```

- To compress a crash dump file using **-p**:

```
core_collector makedumpfile -p -d 31 --message-level 1
```

Additional resources

- **makedumpfile(8)** man page on your system
- Configuration file for kdump

4.5. CONFIGURING THE KDUMP DEFAULT FAILURE RESPONSES

By default, when **kdump** fails to create a crash dump file at the configured target location, the system reboots and the dump is lost in the process. You can change the default failure response and configure **kdump** to perform a different operation in case it fails to save the core dump to the primary target. The additional actions are:

dump_to_rootfs

Saves the core dump to the **root** file system.

reboot

Reboots the system, losing the core dump in the process.

halt

Stops the system, losing the core dump in the process.

poweroff

Power the system off, losing the core dump in the process.

shell

Runs a shell session from within the **initramfs**, you can record the core dump manually.

final_action

Enables additional operations such as **reboot**, **halt**, and **poweroff** after a successful **kdump** or when shell or **dump_to_rootfs** failure action completes. The default is **reboot**.

failure_action

Specifies the action to perform when a dump might fail in a kernel crash. The default is **reboot**.

Prerequisites

- Root permissions.
- Fulfilled requirements for **kdump** configurations and targets. For details, see Supported kdump configurations and targets.

Procedure

1. As **root**, remove the hash sign (**#**) from the beginning of the **#failure_action** line in the **/etc/kdump.conf** configuration file.
2. Replace the value with a desired action.

```
failure_action poweroff
```

Additional resources

- Configuring the kdump target

4.6. CONFIGURATION FILE FOR KDUMP

The configuration file for **kdump** kernel is **/etc/sysconfig/kdump**. This file controls the **kdump** kernel command line parameters. For most configurations, use the default options. However, in some scenarios you might need to modify certain parameters to control the **kdump** kernel behavior. For example, modifying the **KDUMP_COMMANDLINE_APPEND** option to append the **kdump** kernel command-line to obtain a detailed debugging output or the **KDUMP_COMMANDLINE_REMOVE** option to remove arguments from the **kdump** command line.

KDUMP_COMMANDLINE_REMOVE

This option removes arguments from the current **kdump** command line. It removes parameters that may cause **kdump** errors or **kdump** kernel boot failures. These parameters may have been parsed from the previous **KDUMP_COMMANDLINE** process or inherited from the **/proc/cmdline** file. When this variable is not configured, it inherits all values from the **/proc/cmdline** file. Configuring this option also provides information that is helpful in debugging an issue.

To remove certain arguments, add them to **KDUMP_COMMANDLINE_REMOVE** as follows:

```
KDUMP_COMMANDLINE_REMOVE="hugepages hugepagesz slub_debug quiet log_buf_len
swiotlb"
```

KDUMP_COMMANDLINE_APPEND

This option appends arguments to the current command line. These arguments may have been parsed by the previous **KDUMP_COMMANDLINE_REMOVE** variable.
For the **kdump** kernel, disabling certain modules such as **mce**, **cgroup**, **numa**, **hest_disable** can help prevent kernel errors. These modules may consume a significant portion of the kernel memory reserved for **kdump** or cause **kdump** kernel boot failures.

To disable memory **cgroups** on the **kdump** kernel command line, run the command as follows:

```
KDUMP_COMMANDLINE_APPEND="cgroup_disable=memory"
```

Additional resources

- The **Documentation/admin-guide/kernel-parameters.txt** file
- The **/etc/sysconfig/kdump** file

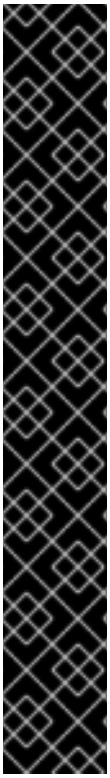
4.7. TESTING THE KDUMP CONFIGURATION

After configuring **kdump**, you must manually test a system crash and ensure that the **vmcore** file is generated in the defined **kdump** target. The **vmcore** file is captured from the context of the freshly booted kernel and therefore has critical information to help debug a kernel crash.

WARNING

Do not test **kdump** on active production systems. The commands to test **kdump** will cause the kernel to crash with loss of data. Depending on your system architecture, ensure that you schedule significant maintenance time because **kdump** testing might require several reboots with a long boot time.

If the **vmcore** file is not generated during the **kdump** test, identify and fix issues before you run the test again for a successful **kdump** testing.



IMPORTANT

Ensure that you schedule significant maintenance time, because **kdump** testing might require several reboots with a long boot time.

If you make any manual system modifications, you must test the **kdump** configuration at the end of any system modification. For example, if you make any of the following changes, ensure that you test the **kdump** configuration for an optimal **kdump** performance:

- Package upgrades.
- Hardware level changes, for example, storage or networking changes.
- Firmware and BIOS upgrades.
- New installation and application upgrades that include third party modules.
- If you use the hot-plugging mechanism to add more memory on hardware that support this mechanism.
- After you make changes in the **/etc/kdump.conf** or **/etc/sysconfig/kdump** file.

Prerequisites

- You have root permissions on the system.
- You have saved all important data. The commands to test **kdump** cause the kernel to crash with loss of data.
- You have scheduled significant machine maintenance time depending on the system architecture.

Procedure

1. Enable the **kdump** service:

```
# kdumpctl restart
```

2. Check the status of the **kdump** service. With the **kdumpctl** command, you can print the output at the console.

```
# kdumpctl status
  kdump:Kdump is operational
```

   Alternatively, if you use the **systemctl** command, the output prints in the **systemd** journal.

3. Initiate a kernel crash to test the **kdump** configuration. The **sysrq-trigger** key combination causes the kernel to crash and might reboot the system if required.

```
# echo c > /proc/sysrq-trigger
```

   On a kernel reboot, the ***address-YYYY-MM-DD-HH:MM:SS*/vmcore** file is created at the location you have specified in the **/etc/kdump.conf** file. The default is **/var/crash/**.

Additional resources

Additional resources

- Configuring the kdump target

4.8. FILES PRODUCED BY KDUMP AFTER SYSTEM CRASH

After your system crashes, the **kdump** service captures the kernel memory in a dump file (**vmcore**) and it also generates additional diagnostic files to aid in troubleshooting and post-mortem analysis.

Files produced by **kdump**:

- **vmcore** - main kernel memory dump file containing system memory at the time of the crash. It includes data as per the configuration of the **core_collector** program specified in **kdump** configuration. By default the kernel data structures, process information, stack traces, and other diagnostic information.
- **vmcore-dmesg.txt** - contents of the kernel ring buffer log (**dmesg**) from the primary kernel that panicked.
- **kexec-dmesg.log** - contains kernel and system log messages from the execution of the secondary **kexec** kernel that collects the **vmcore** data.

Additional resources

- What is the kernel ring buffer
- What is kdump

4.9. ENABLING AND DISABLING THE KDUMP SERVICE

You can configure to enable or disable the **kdump** functionality on a specific kernel or on all installed kernels. You must routinely test the **kdump** functionality and validate that it is working properly.

Prerequisites

- You have root permissions on the system.
- You have completed **kdump** requirements for configurations and targets. See Supported kdump configurations and targets.
- All configurations for installing **kdump** are set up as required.

Procedure

- Enable the **kdump** service for **multi-user.target**:

```
# systemctl enable kdump.service
```

- Start the service in the current session:

```
# systemctl start kdump.service
```

- Stop the **kdump** service:

```
# systemctl stop kdump.service
```

-

- Disable the **kdump** service:

```
# systemctl disable kdump.service
```

WARNING

It is recommended to set **kptr_restrict=1** as default. When **kptr_restrict** is set to (1) as default, the **kdumpctl** service loads the crash kernel even if Kernel Address Space Layout (KASLR) is enabled or not enabled.

If **kptr_restrict** is not set to **1** and KASLR is enabled, the contents of **/proc/kore** file are generated as all zeros. The **kdumpctl** service fails to access the **/proc/kcore** file and load the crash kernel. The **kexec-kdump-howto.txt** file displays a warning message, which recommends you to set **kptr_restrict=1**. Verify for the following in the **sysctl.conf** file to ensure that **kdumpctl** service loads the crash kernel:

- Kernel **kptr_restrict=1** in the **sysctl.conf** file.

4.10. PREVENTING KERNEL DRIVERS FROM LOADING FOR KDUMP

You can control the capture kernel from loading certain kernel drivers by adding the **KDUMP_COMMANDLINE_APPEND=** variable in the **/etc/sysconfig/kdump** configuration file. By using this method, you can prevent the **kdump** initial RAM disk image **initramfs** from loading the specified kernel module. This helps to prevent the out-of-memory (OOM) killer errors or other crash kernel failures.

You can append the **KDUMP_COMMANDLINE_APPEND=** variable using one of the following configuration options:

- **rd.driver.blacklist=*<modules>***
- **modprobe.blacklist=*<modules>***

Prerequisites

- You have root permissions on the system.

Procedure

1. Display the list of modules that are loaded to the currently running kernel. Select the kernel module that you intend to block from loading.

```
$ lsmod

Module                  Size  Used by
fuse                  126976  3
xt_CHECKSUM            16384  1
```

```
ipt_MASQUERADE         16384  1
uinput                 20480  1
xt_conntrack           16384  1
```

2. Update the **KDUMP_COMMANDLINE_APPEND=** variable in the **/etc/sysconfig/kdump** file. For example:

```
KDUMP_COMMANDLINE_APPEND="rd.driver.blacklist=hv_vmbus,hv_storvsc,hv_utils,hv_netvsc,hid-hyperv"
```

Also, consider the following example using the **modprobe.blacklist=<modules>** configuration option:

```
KDUMP_COMMANDLINE_APPEND="modprobe.blacklist=emcp modprobe.blacklist=bnx2fc modprobe.blacklist=libfcoe modprobe.blacklist=fcoe"
```

3. Restart the **kdump** service:

```
# systemctl restart kdump
```

Additional resources

- **dracut.cmdline** man page on your system

4.11. RUNNING KDUMP ON SYSTEMS WITH ENCRYPTED DISK

When you run a LUKS encrypted partition, systems require certain amount of available memory. If the system has less than the required amount of available memory, the **cryptsetup** utility fails to mount the partition. As a result, capturing the **vmcore** file to an encrypted target location fails in the second kernel (capture kernel).

The **kdumpctl estimate** command helps you estimate the amount of memory you need for **kdump**. **kdumpctl estimate** prints the recommended **crashkernel** value, which is the most suitable memory size required for **kdump**.

The recommended **crashkernel** value is calculated based on the current kernel size, kernel module, initramfs, and the LUKS encrypted target memory requirement.

In case you are using the custom **crashkernel=** option, **kdumpctl estimate** prints the **LUKS required size** value. The value is the memory size required for LUKS encrypted target.

Procedure

1. Print the estimate **crashkernel=** value:

```
# *kdumpctl estimate*

Encrypted kdump target requires extra memory, assuming using the keyslot with minimum memory requirement
  Reserved crashkernel:    256M
  Recommended crashkernel: 652M

  Kernel image size:   47M
  Kernel modules size: 8M
```

```
Initramfs size:       20M
Runtime reservation: 64M
LUKS required size:  512M
Large modules: <none>
WARNING: Current crashkernel size is lower than recommended size 652M.
```

2. Configure the amount of required memory by increasing the **crashkernel=** value.

3. Reboot the system.

NOTE

If the **kdump** service still fails to save the dump file to the encrypted target, increase the **crashkernel=** value as required.

CHAPTER 5. ENABLING KDUMP

For your Red Hat Enterprise Linux 9 systems, you can configure to enable or disable the **kdump** functionality on a specific kernel or on all installed kernels. However, you must routinely test the **kdump** functionality and validate that it's working properly.

5.1. ENABLING KDUMP FOR ALL INSTALLED KERNELS

The **kdump** service starts by enabling **kdump.service** after the **kexec** tool is installed. You can enable and start the **kdump** service for all kernels installed on the machine.

Prerequisites

- You have administrator privileges.

Procedure

1. Add the **crashkernel=** command-line parameter to all installed kernels.

```
# grubby --update-kernel=ALL --args="crashkernel=xxM"
```

   xxM is the required memory in megabytes.

2. Enable the **kdump** service.

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

Verification

- Check that the **kdump** service is running.

```
# systemctl status kdump.service

○ kdump.service - Crash recovery kernel arming
     Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset:
disabled)
     Active: active (live)
```

5.2. ENABLING KDUMP FOR A SPECIFIC INSTALLED KERNEL

You can enable the **kdump** service for a specific kernel on the machine.

Prerequisites

- You have administrator privileges.

Procedure

1. List the kernels installed on the machine.

```
# ls -a /boot/vmlinuz-*
/boot/vmlinuz-0-rescue-2930657cd0dc43c2b75db480e5e5b4a9
```

```
/boot/vmlinuz-4.18.0-330.el8.x86_64
/boot/vmlinuz-4.18.0-330.rt7.111.el8.x86_64
```

2. Add a specific **kdump** kernel to the system's Grand Unified Bootloader (GRUB) configuration. For example:

```
# grubby --update-kernel=vmlinuz-4.18.0-330.el8.x86_64 --args="crashkernel=xxM"
```

xxM is the required memory reserve in megabytes.

3. Enable the **kdump** service.

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

Verification

- Check that the **kdump** service is running.

```
# systemctl status kdump.service

○ kdump.service - Crash recovery kernel arming
     Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset:
disabled)
     Active: active (live)
```

5.3. DISABLING THE KDUMP SERVICE

You can stop the **kdump.service** and disable the service from starting on your Red Hat Enterprise Linux 9 systems.

Prerequisites

- Fulfilled requirements for **kdump** configurations and targets. For details, see Supported kdump configurations and targets.
- All configurations for installing **kdump** are set up according to your needs. For details, see Installing kdump.

Procedure

1. To stop the **kdump** service in the current session:

```
# systemctl stop kdump.service
```

2. To disable the **kdump** service:

```
# systemctl disable kdump.service
```

WARNING

It is recommended to set **kptr_restrict=1** as default. When **kptr_restrict** is set to (1) as default, the **kdumpctl** service loads the crash kernel even if Kernel Address Space Layout (**KASLR**) is enabled or not enabled.

If **kptr_restrict** is not set to **1** and **KASLR** is enabled, the contents of **/proc/kore** file are generated as all zeros. The **kdumpctl** service fails to access the **/proc/kcore** file and load the crash kernel. The **kexec-kdump-howto.txt** file displays a warning message, which recommends you to set **kptr_restrict=1**. Verify for the following in the **sysctl.conf** file to ensure that **kdumpctl** service loads the crash kernel:

- Kernel **kptr_restrict=1** in the **sysctl.conf** file.

Additional resources

- Managing systemd

CHAPTER 6. REPORTING RHEL FOR REAL TIME BUGS

The preferred method for reporting a RHEL for Real Time bug is to submit a bug report on the Red Hat Bugzilla. Before filing a bug, it is useful to identify the source where the problem occurred, such as the standard kernel or the RHEL for Real Time kernel.

6.1. DIAGNOSING RHEL FOR REAL TIME BUGS

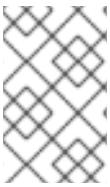
Identifying which kernel, the RHEL for Real Time or the standard kernel, is the source of the problem can increase the chances of having your bug fixed faster. By following the procedure steps, you can diagnose the source of the problem before submitting a bug report.

Prerequisite:

- The latest version of RHEL for Real Time kernel is installed.

Procedure:

1. Verify that you have the latest version of the RHEL for Real Time kernel.
2. Boot into RHEL for Real Time kernel using the **GRUB** menu.
3. If the problem occurs, report a bug against RHEL for Real Time.
4. Try to reproduce the problem with the standard kernel.
   This troubleshooting step assists in identifying the problem location.



NOTE

If the problem does not occur with the standard kernel, then the bug is probably the result of changes introduced in the RHEL for Real Time specific enhancements, which Red Hat has applied on top of the baseline (4.18.0) kernel.

6.2. SUBMITTING A BUG REPORT WITH BUGZILLA

After identifying the bug specific to RHEL for Real Time, use the following procedure steps to submit a bug report with Bugzilla.

Prerequisite:

- You have a Red Hat Bugzilla account.

Procedure

1. Log into your Bugzilla account.
2. Click **Enter A New Bug Report**.
3. Select **Red Hat classification**.
4. Select the **Red Hat Enterprise Linux** product.

5. Enter **Component**.
   For example, use **kernel-rt** if it is a kernel issue or the name of the affected user space component, such as **rteval**.

6. Provide a detailed description of the bug issue for RHEL for Real Time kernel.
   When entering the problem description you can also state if you were able to reproduce the problem on the standard RHEL 8 kernel.

Additional resources

- Red Hat Bugzilla – Create a new Red Hat Bugzilla account