Red Hat Enterprise Linux 7
Kernel Crash Dump Guide

Kernel Crash Dump Configuration and Analysis

Mark Flitter Jaromír Hradílek Petr Bokoč

Red Hat Enterprise Linux 7 Kernel Crash Dump Guide

Kernel Crash Dump Configuration and Analysis

Mark Flitter
Red Hat Customer Content Services
mflitter@redhat.com

Jaromír Hradílek
Red Hat Customer Content Services

Petr Bokoč
Red Hat Customer Content Services

Legal Notice

Abstract

The Kernel Crash Dump Guide documents how to configure, test, and use the kdump crash collection service on Red Hat Enterprise Linux 7, and provides a brief overview of how to analyze the resulting core dump using the crash debugging utility. It is oriented towards system administrators with a basic understanding of the Red Hat Enterprise Linux system.

Table of Contents

Chapter 1. Introduction to kdump

1.1. About kdump and kexec

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, bypass BIOS, and preserve the contents of the first kernel's memory that would otherwise be lost.

In case of a system crash, kdump uses kexec to boot into a second kernel (a *capture kernel*). This second kernel resides in a reserved part of the system memory that is inaccessible to the first kernel. The second kernel then captures the contents of the crashed kernel's memory (a *crash dump*) and saves it.

1.2. Memory Requirements

In order for kdump to be able to capture a kernel crash dump and save it for further analysis, a part of the system memory has to be permanently reserved for the capture kernel. When reserved, this part of the system memory is not available to main kernel.

The memory requirements vary based on certain system parameters. One of the major factors is the system's hardware architecture. To find out the exact name of the machine architecture (such as **x86_64**) and print it to standard output, type the following command at a shell prompt:

```
uname -m
```

Another factor which influences the amount of memory to be reserved is the total amount of installed system memory. For example, on the x86_64 architecture, the amount of reserved memory will be 160 MB + 2 bits for every 4 KB of RAM. On a system with 1 TB of total physical memory installed, this means 224 MB (160 MB + 64 MB). For a complete list of memory requirements for kdump based on the system architecture and the amount of physical memory, see Section B.1, “Memory Requirements for kdump”.

On many systems, kdump can estimate the amount of required memory and reserve it automatically. This behavior is enabled by default, but only works on systems that have more than a certain amount of total available memory, which varies based on the system architecture. See Section B.2, “Minimum Threshold for Automatic Memory Reservation” for a list of minimum requirements for automatic memory reservation based on the system architecture.

If the system has less than the minimum amount of memory required for the automatic allocation to work or if your use case requires a different value, you can configure the amount of reserved memory manually. For information on how to do so on the command line, see Section 2.2.1, “Configuring the Memory Usage”. For information on how to configure the amount of reserved memory in the graphical user interface, see Section 2.3.1, “Configuring the Memory Usage”.

Important

It is highly recommended to test the configuration after setting up the kdump service, even when using the automatic memory reservation. For instructions on how to test your configuration, see Section 2.4, “Testing the kdump Configuration”.

Chapter 2. Installing and Configuring kdump

2.1. Installing kdump

In many cases, the **kdump** service is installed and activated by default on new Red Hat Enterprise Linux 7 installations. The **Anaconda** installer provides a screen for kdump configuration when performing an interactive installation using the graphical or text interface. The installer screen is titled **Kdump** and is available from the main **Installation Summary** screen, and only allows limited configuration - you can only select whether kdump will be enabled and how much memory will be reserved. Information about memory requirements for kdump is available in Section B.1, "Memory Requirements for kdump". The Kdump configuration screen in the installer is documented in the *Red Hat Enterprise Linux 7 Installation Guide*.

Note

In previous releases of Red Hat Enterprise Linux, kdump configuration was available in the **Firstboot** utility which was automatically executed *after* the installation finished and the system rebooted for the first time. Starting with Red Hat Enterprise Linux 7.1, kdump configuration has been moved into the installer.

Some installation options, such as custom Kickstart installations, may not install or enable kdump by default. If this is the case on your system, and you want to install kdump additionally, execute the following command as **root** at a shell prompt:

```
# yum install kexec-tools
```

This will install kdump and all other necessary packages, assuming your system has an active subscription or a custom repository containing the *kexec-tools* package for your system's architecture.

Note

If you do not know whether kdump is installed on your system, you can check using **rpm**:

```
$ rpm -q kexec-tools
```

Additionally, a graphical configuration tool is available, but not installed by default if you use the command described above. To install this utility, which is described in Section 2.3, "Configuring kdump in the Graphical User Interface", use the following command as **root**:

```
# yum install system-config-kdump
```

For more information on how to install new packages in Red Hat Enterprise Linux 7 using the **Yum** package manager, see the *Red Hat Enterprise Linux 7 System Administrator's Guide*.

Important

Starting with Red Hat Enterprise Linux 7.4 the **Intel IOMMU** driver is supported with **kdump**. When running kernels from version 7.3 or earlier, it is advised that **Intel IOMMU** support is disabled.

2.2. Configuring kdump on the Command Line

2.2.1. Configuring the Memory Usage

Memory reserved for the kdump kernel is always reserved during system boot, which means that the amount of memory is specified in the system's boot loader configuration. This section will explain how to change the amount of reserved memory on AMD64 and Intel 64 systems and IBM Power Systems servers using the **GRUB2** boot loader, and on IBM System z using **zipl**.

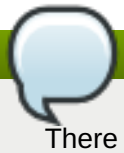
Procedure 2.1. Changing Memory Options in GRUB2 for AMD64 and Intel 64 systems and IBM Power Systems Hardware.

1. Open the **/etc/default/grub** configuration file as **root** using a plain text editor such as **vim** or **Gedit**.
2. In this file, locate the line beginning with **GRUB_CMDLINE_LINUX**. The line will look similar to the following:

   ```
   GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto
   rd.lvm.lv=rhel/root rhgb quiet"
   ```

   Note the highlighted **crashkernel=** option; this is where the reserved memory is configured.
3. Change the value of the **crashkernel=** option to the amount of memory you want to reserve. For example, to reserve 128 MB of memory, use the following:

   ```
   crashkernel=128M
   ```

   Note

   There are multiple ways to configure the memory reserved - for example, you can define an offset or multiple memory amounts based on how much RAM is available in the system at startup. This is described further in this section.

   Then, save the file and exit the editor.
4. Finally, regenerate the **GRUB2** configuration using the edited **default** file. If your system uses BIOS firmware, execute the following command:

   ```
   # grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

   On a system with UEFI firmware, execute the following instead:

   ```
   # grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
   ```

After finishing the procedure above, the boot loader is reconfigured and the amount of memory you have specified in its configuration file will be reserved after the next reboot.

Procedure 2.2. Changing Memory Options in zipl for IBM System z Hardware

1. Open the **/etc/zipl.conf** configuration file as **root** using a plain text editor such as **vim** or **Gedit**.
2. In this file, locate the **parameters=** section, and edit the **crashkernel=** parameter (or add it if not present). For example, to reserve 128 MB of memory, use the following:

```
crashkernel=128M
```

Note

There are multiple ways to configure the memory reserved - for example, you can define an offset or multiple memory amounts based on how much RAM is available in the system at startup. This is described further in this section.

Then, save the file and exit the editor.

3. Finally, regenerate the **zipl** configuration:

```
# zipl
```

Note

Executing only the **zipl** command with no additional options will use default values. See the **zipl(8)** man page for information about available options.

After finishing the procedure above, the boot loader is reconfigured and the amount of memory you have specified in its configuration file will be reserved after the next reboot.

The **crashkernel=** option can be defined in multiple ways. The **auto** value enables automatic configuration of reserved memory based on the total amount of memory in the system, following the guidelines described in Section B.1, “Memory Requirements for kdump”. Replace the **auto** value with a specific amount of memory to change this behavior. For example, to reserve 128 MB of memory, use the following:

```
crashkernel=128M
```

You can also set the amount of reserved memory to be variable, depending on the total amount of installed memory. The syntax is for variable memory reservation is **crashkernel=*<range1>*:*<size1>*,*<range2>*:*<size2>***. For example:

```
crashkernel=512M-2G:64M,2G-:128M
```

The above example will reserve 64 MB of memory if the total amount of system memory is 512 MB or higher and lower than 2 GB. If the total amount of memory is more than 2 GB, 128 MB will be reserved for kdump instead.

On some systems, it might be necessary to reserve memory with a certain fixed offset. If the offset is set, the reserved memory will begin there. To offset the reserved memory, use the following syntax:

```
crashkernel=128M@16M
```

The example above means that kdump will reserve 128 MB of memory starting at 16 MB (physical address 0x01000000). If the offset parameter is set to 0 or omitted entirely, kdump will offset the reserved memory automatically. This syntax can also be used when setting a variable memory reservation as described above; in this case, the offset is always specified last (for example, **crashkernel=512M-2G:64M,2G-:128M@16M**).

2.2.2. Configuring the kdump Type

When a kernel crash is captured, the core dump can be either stored as a file in a local file system, written directly to a device, or sent over a network using the **NFS** (Network File System) or **SSH** (Secure Shell) protocol. Only one of these options can be set at the moment, and the default option is to store the **vmcore** file in the **/var/crash/** directory of the local file system. To change this, as **root**, open the **/etc/kdump.conf** configuration file in a text editor and edit the options as described below.

To change the local directory in which the core dump is to be saved, remove the hash sign ("#") from the beginning of the **#path /var/crash** line, and replace the value with a desired directory path.

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

Important

In Red Hat Enterprise Linux 7, the directory defined as the kdump target using the **path** directive must exist when the **kdump** systemd service is started - otherwise the service will fail. This behavior is different from earlier releases of Red Hat Enterprise Linux, where the directory was being created automatically if it did not exist when starting the service.

Optionally, if you wish to write the file to a different partition, follow the same procedure with the one of the lines beginning with **#ext4**. Here, you can use either a device name (the **#ext4 /dev/vg/lv_kdump** line), a file system label (the **#ext4 LABEL=/boot** line) or a UUID (the **#ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937** line). Change the file system type as well as the device name, label or UUID to the desired values. For example:

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

Important

Specifying storage devices using a **LABEL=** or **UUID=** is recommended. Disk device names such as **/dev/sda3** are not guaranteed to be consistent across reboot. See the *Red Hat Enterprise Linux 7 Storage Administration Guide* for information about persistent disk device naming.

Important

When dumping to DASD on s390x hardware, it is essential that the dump devices are correctly specified in **/etc/dasd.conf** before proceeding.

To write the dump directly to a device, remove the hash sign ("#") from the beginning of the **#raw /dev/vg/lv_kdump** line, and replace the value with a desired device name. For example:

```
raw /dev/sdb1
```

To store the dump to a remote machine using the **NFS** protocol, remove the hash sign ("#") from the beginning of the **#nfs my.server.com:/export/tmp** line, and replace the value with a valid hostname and directory path. For example:

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

To store the dump to a remote machine using the **SSH** protocol, remove the hash sign ("#") from the beginning of the **#ssh user@my.server.com** line, and replace the value with a valid username and hostname. To include your SSH key in the configuration as well, remove the hash sign ("#") from the beginning of the **#sshkey /root/.ssh/kdump_id_rsa** line and 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
```

For information on how to configure an SSH server and set up a key-based authentication, see the *Red Hat Enterprise Linux 7 System Administrator's Guide*.

For a complete list of currently supported and unsupported targets sorted by type, see Table B.3, "Supported kdump Targets".

2.2.3. Configuring the Core Collector

To reduce the size of the **vmcore** dump file, **kdump** allows you to specify an external application (a *core collector*) to compress the data, and optionally leave out all irrelevant information. Currently, the only fully supported core collector is **makedumpfile**.

To enable the core collector, as **root**, open the **/etc/kdump.conf** configuration file in a text editor, remove the hash sign ("#") from the beginning of the **#core_collector makedumpfile -l --message-level 1 -d 31** line, and edit the command line options as described below.

To enable the dump file compression, add the **-c** parameter. For example:

```
core_collector makedumpfile -c
```

To remove certain pages from the dump, add the **-d *value*** parameter, where *value* is a sum of values of pages you want to omit as described in Table B.4, "Supported Filtering Levels". For example, to remove both zero and free pages, use the following:

```
core_collector makedumpfile -d 17 -c
```

See the **makedumpfile(8)** man page for a complete list of available options.

2.2.4. Configuring the Default Action

By default, when **kdump** fails to create a core dump at the target location specified in Section 2.2.2, "Configuring the kdump Type", the root file system is mounted and **kdump** attempts to save the core locally. To change this behavior, as **root**, open the **/etc/kdump.conf** configuration file in a text editor, remove the hash sign ("#") from the beginning of the **#default shell** line, and replace the value with a desired action as described in Table B.5, "Supported Default Actions".

For example:

```
default reboot
```

2.2.5. Enabling the Service

To start the **kdump** daemon at boot time, type the following at a shell prompt as **root**:

```
systemctl enable kdump.service
```

This will enable the service for **multi-user.target**. Similarly, typing **systemctl stop kdump** will disable it. To start the service in the current session, use the following command as **root**:

```
systemctl start kdump.service
```

Important

In Red Hat Enterprise Linux 7, the directory defined as the kdump target must exist when the **kdump** systemd service is started - otherwise the service will fail. This behavior is different from earlier releases of Red Hat Enterprise Linux, where the directory was being created automatically if it did not exist when starting the service.

For more information on systemd and configuring services in general, see the *Red Hat Enterprise Linux 7 System Administrator's Guide*.

2.3. Configuring kdump in the Graphical User Interface

To start the **Kernel Dump Configuration** utility, select **Activities** → **Other** → **Kernel crash dumps** from the panel, or type **system-config-kdump** at a shell prompt. You will be presented with a window as shown in Figure 2.1, "Basic Settings".

The utility allows you to configure **kdump** as well as to enable or disable starting the service at boot time. When you are done, click **Apply** to save the changes. Unless you are already authenticated, you will be prompted to enter the superuser password. The utility will also remind you that you must reboot the system in order to apply any changes you have made to the configuration.

Important

On IBM System z or PowerPC systems with **SELinux** running in Enforcing mode, the kdumpgui_run_bootloader Boolean must be enabled before launching the Kernel Dump Configuration utility. This Boolean allows system-config-kdump to run the boot loader in the bootloader_t SELinux domain. To permanently enable the Boolean, run the following command as root;

```
# setsebool -P kdumpgui_run_bootloader 1
```

Important

When dumping to DASD on s390x hardware, it is essential that the dump devices are correctly specified in **/etc/dasd.conf** before proceeding.

2.3.1. Configuring the Memory Usage

The **Basic Settings** tab enables you to configure the amount of memory that is reserved for the **kdump** kernel. To do so, select the **Manual settings** radio button, and click the up and down arrow buttons next to the **New kdump Memory** field to increase or decrease the amount of memory to be reserved. Notice that the **Usable Memory** field changes accordingly showing you the remaining memory that will be available to the system. See Section 1.2, “Memory Requirements” for more information on kdump's memory requirements.

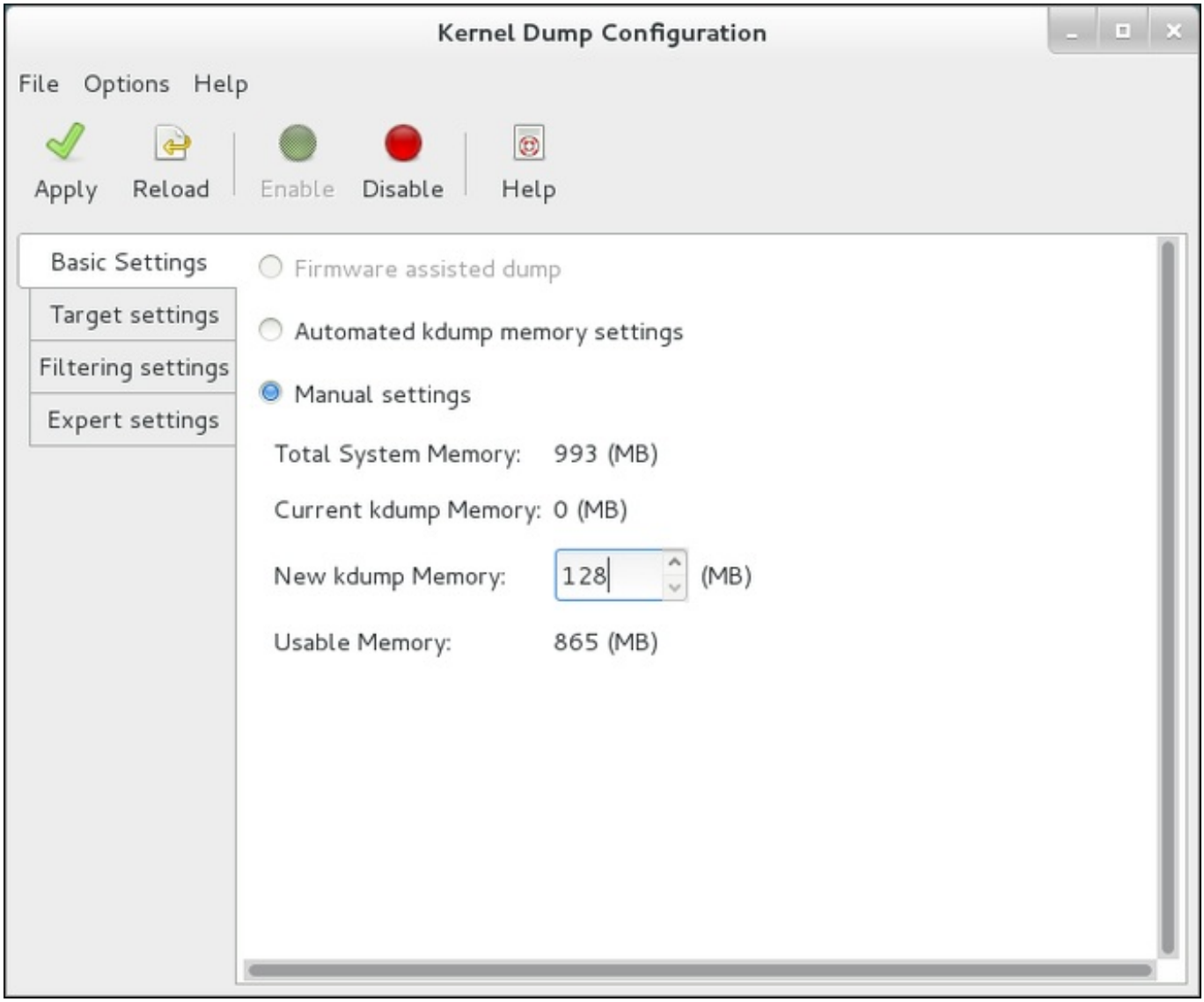


Figure 2.1. Basic Settings

2.3.2. Configuring the kdump Type

The **`Target Settings`** tab allows you to specify the target location for the **`vmcore`** dump. The dump can be either stored as a file in a local file system, written directly to a device, or sent over a network using the **NFS** (Network File System) or **SSH** (Secure Shell) protocol.

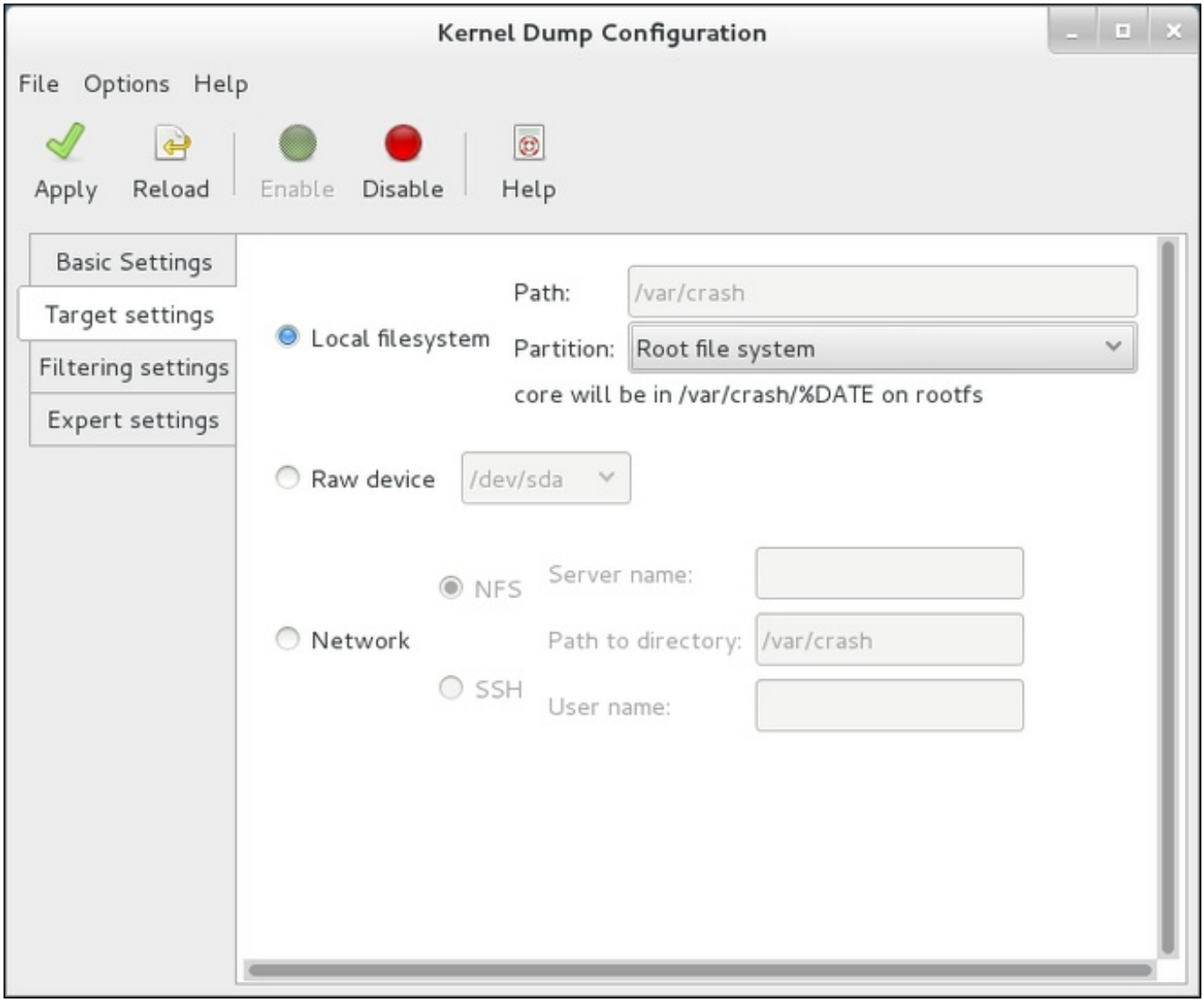


Figure 2.2. Target Settings

To save the dump to the local file system, select the **Local filesystem** radio button. Optionally, you can customize the settings by choosing a different partition from the **Partition** drop-down list and a target directory using the **Path** field.

Important

In Red Hat Enterprise Linux 7, the directory defined as the kdump target must exist when the **kdump** systemd service is started - otherwise the service will fail. This behavior is different from earlier releases of Red Hat Enterprise Linux, where the directory was being created automatically if it did not exist when starting the service.

To write the dump directly to a device, select the **Raw device** radio button, and choose the desired target device from the drop-down list next to it.

To send the dump to a remote machine over a network connection, select the **Network** radio button. To use the **NFS** protocol, select the **NFS** radio button, and fill the **Server name** and **Path to directory** fields. To use the **SSH** protocol, select the **SSH** radio button, and fill the **Server name**, **Path to directory**, and **User name** fields with the remote server address, target directory, and a valid user name respectively.

For information on how to configure an SSH server and set up a key-based authentication, see the *Red Hat Enterprise Linux 7 System Administrator's Guide*. For a complete list of currently supported targets, see Table B.3, “Supported kdump Targets”.

2.3.3. Configuring the Core Collector

The **`Filtering Settings`** tab enables you to select the filtering level for the **`vmcore`** dump.

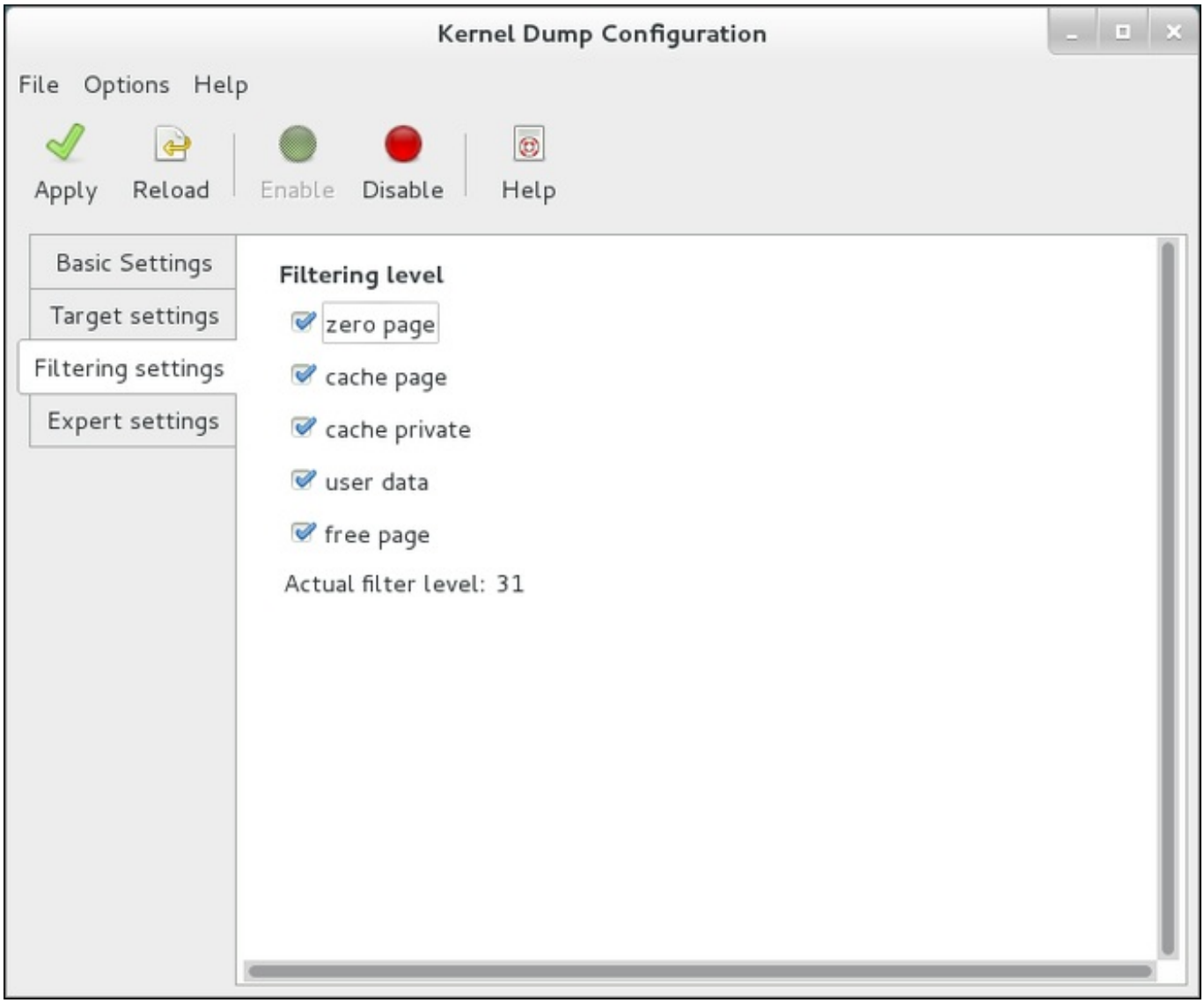


Figure 2.3. Filtering Settings

To exclude the **`zero page`**, **`cache page`**, **`cache private`**, **`user data`**, or **`free page`** from the dump, select the checkbox next to the appropriate label.

2.3.4. Configuring the Default Action

To choose what action to perform when **kdump** fails to create a core dump, select an appropriate option from the **`Default action`** drop-down list. Available options are **dump to rootfs and reboot** (the default action which attempts to save the core locally and then reboots the system), **reboot** (to reboot the system), **shell** (to present a user with an interactive shell prompt), **halt** (to halt the system), and **poweroff** (to power the system off).

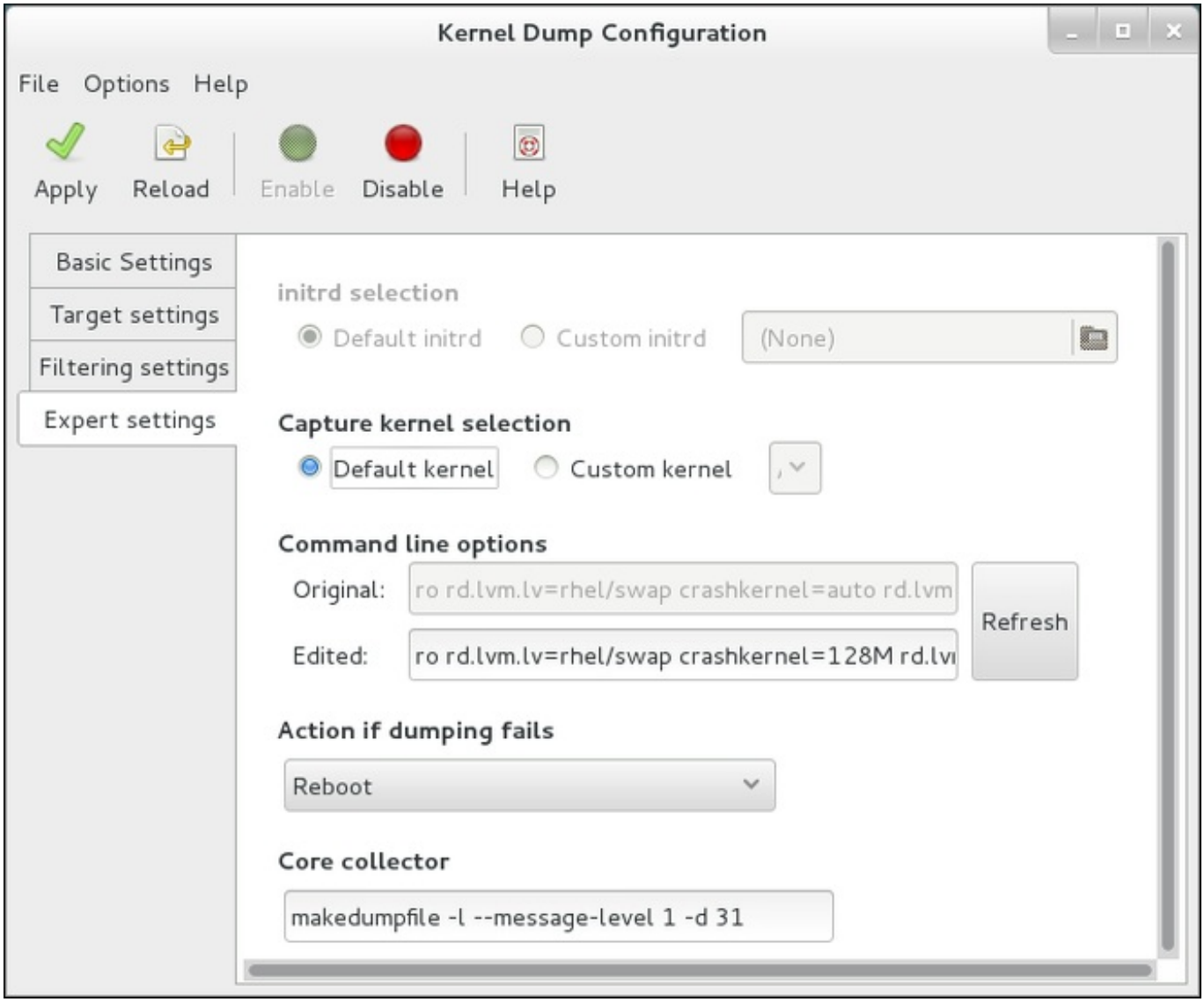


Figure 2.4. Filtering Settings

To customize the options that are passed to the **makedumpfile** core collector, edit the **Core collector** text field; see Section 2.2.3, "Configuring the Core Collector" for more information.

2.3.5. Enabling the Service

To start the **kdump** service at boot time, click the **Enable** button on the toolbar and then click the **Apply** button. This will enable and activate the service for **multi-user.target**. Clicking the **Disable** button and confirming by clicking the **Apply** button will disable the service immediately.

Important

In Red Hat Enterprise Linux 7, the directory defined as the kdump target must exist when the **kdump** systemd service is started - otherwise the service will fail. This behavior is different from earlier releases of Red Hat Enterprise Linux, where the directory was being created automatically if it did not exist when starting the service.

For more information on systemd targets and configuring services in general, see the *Red Hat Enterprise Linux 7 System Administrator's Guide*.

2.4. Testing the kdump Configuration

Warning

The commands below will cause the kernel to crash. Use caution when following these steps, and by no means use them on a production system.

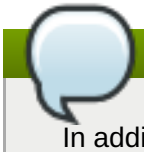
To test the configuration, reboot the system with **kdump** enabled, and make sure that the service is running:

```
~]# systemctl is-active kdump
active
```

Then type the following commands at a shell prompt:

```
echo 1 > /proc/sys/kernel/sysrq
echo c > /proc/sysrq-trigger
```

This will force the Linux kernel to crash, and the ***address-YYYY-MM-DD-HH:MM:SS*/vmcore** file will be copied to the location you have selected in the configuration (that is, to **/var/crash/** by default).



Note

In addition to confirming the validity of the configuration, this action can also be used to record how long a crash dump will take to complete if it is performed under a representative test load.

2.5. Additional Resources

2.5.1. Installed Documentation

- **kdump.conf**(5) — a manual page for the **/etc/kdump.conf** configuration file containing the full documentation of available options.
- **zipl.conf**(5) — a manual page for the **/etc/zipl.conf** configuration file.
- **zipl**(8) — a manual page for the **zipl** boot loader utility for IBM System z.
- **makedumpfile**(8) — a manual page for the **makedumpfile** core collector.
- **kexec**(8) — a manual page for **kexec**.
- **crash**(8) — a manual page for the **crash** utility.
- **/usr/share/doc/kexec-tools-*version*/kexec-kdump-howto.txt** — an overview of the **kdump** and **kexec** installation and usage.

2.5.2. Online Documentation

https://access.redhat.com/site/solutions/6038

The Red Hat Knowledgebase article about the **kexec** and **kdump** configuration.

https://access.redhat.com/site/solutions/223773

The Red Hat Knowledgebase article about supported **kdump** targets.

http://people.redhat.com/anderson/

The **crash** utility homepage.

https://www.gnu.org/software/grub/

The **GRUB2** boot loader homepage and documentation.

Chapter 3. Firmware Assisted Dump Mechanisms

3.1. The Case for Firmware Assisted Dump

The **kexec** and **kdump** mechanisms are a reliable and proven method of capturing a core dump on AMD64 and Intel 64 systems. However, some hardware with a longer history, particularly mini and mainframe systems, allows us to leverage the onboard firmware to isolate regions of memory and prevent any accidental overwriting of data that may be important to the crash analysis.

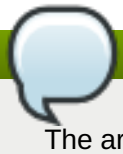
This chapter covers some of the available firmware assisted dump methods and how they integrate with Red Hat Enterprise Linux.

3.2. Using fadump on IBM PowerPC hardware

Firmware-assisted dump (**fadump**) is a reliable alternative to **kexec-kdump** available on IBM PowerPC LPARS. It captures vmcore from a fully-reset system with PCI and I/O devices reinitialized. While this mechanism uses the firmware to preserve the memory in case of a crash, it reuses the **kdump** userspace scripts to save the vmcore"

To achieve this, **fadump** registers the regions of memory that must be preserved in the event of a crash with the system firmware. These regions consist of all the system memory contents, except the boot memory, system registers and hardware Page Table Entries (PTEs).

For further details about the **fadump** mechanism, including PowerPC-specific methods of resetting hardware, review **/usr/share/doc/kexec-tools-*X.y.z*/fadump-howto.txt** where "X.y.z" correspond to the version number of *kexec-tools* installed on your system.



Note

The area of memory not preserved and known as **boot memory** is the amount of RAM required to successfully boot the kernel after a crash event. By default, the boot memory size is 256MB or 5% of total system RAM, whichever is larger.

Unlike a **kexec**-initiated event, the **fadump** process uses the production kernel to recover a crash dump. When booting after a crash, PowerPC hardware makes the device node **/proc/device-tree/rtas/ibm,kernel-dump** available to **procfs**, which the fadump-aware **kdump** scripts check for to save the vmcore. After this has completed, the system is rebooted cleanly.

Enabling fadump

1. Install and configure **kdump** as described in Chapter 2, *Installing and Configuring kdump*.
2. Add **fadump=on** to the **GRUB_CMDLINE_LINUX** line in **/etc/default/grub**:

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto
rd.lvm.lv=rhel/root rhgb quiet fadump=on"
```

3. (optional) If you want to specify reserved boot memory instead of accepting the defaults, add **fadump_reserve_mem=*xx*M** to **GRUB_CMDLINE_LINUX** in **/etc/default/grub**, where *xx* is the amount of the memory required in megabytes:

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto
rd.lvm.lv=rhel/root rhgb quiet fadump=on fadump_reserve_mem=xxM"
```

Important

As with all boot configuration options, it is strongly recommended that you test the configuration before it is needed. If you observe Out of Memory (OOM) errors when booting from the crash kernel, increase the value specified in **fadump_reserve_mem=** until the crash kernel can boot cleanly. Some trial and error may be required in this case.

3.3. Firmware Assisted Dump Methods on IBM z Systems

There are two firmware assisted dump mechanisms on IBM z Systems. They are **Stand-alone Dump** and **VMDUMP**.

The **kdump** infrastructure is supported and utilized on these systems and configuration from Red Hat Enterprise Linux is described in Chapter 2, *Installing and Configuring kdump*. However, there are potentially some advantages to using either of the firmware assisted methods IBM z System hardware provides.

The Stand-alone Dump (SADMP) mechanism is initiated and controlled from the system console, and must be stored on an IPL bootable device.

Similar to SADMP is VMDUMP. This tool is also initiated from the system console, but has a mechanism to retrieve the resulting dump from hardware and copy it to a system for analysis.

One advantage of these methods (and similarly to other hardware based dump mechanisms), is the ability to capture the state of a machine in the Early Boot phase (before the kdump service is started)

Although VMDUMP contains a mechanism to receive the dump file into a Red Hat Enterprise Linux system, the configuration and control of both SADMP and VMDUMP are managed from the IBM z System Hardware console.

IBM discuss SADMP in detail, at http://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.ieav100/standa.htm and VMDUMP at http://www.ibm.com/support/knowledgecenter/en/linuxonibm/com.ibm.linux.z.lgdt/lgdt_t_vmdump.html

IBM also have a documentation set for using the dump tools on Red Hat Enterprise Linux 7 at http://www.ibm.com/support/knowledgecenter/linuxonibm/com.ibm.linux.z.lgdt/lgdt_t_usingdumptools.html

3.4. Using sadump on Fujitsu PRIMEQUEST systems

The Fujitsu **sadump** mechanism is designed to provide a fallback dump capture in the event **kdump** is unable to complete successfully.

The **sadump** process is invoked manually from the system ManageMent Board (MMB) interface.

With this system, you should configure kdump as normal for an X86_64 server and then perform the following additional steps to enable **sadump**.

Add or edit the following lines in **/etc/sysctl.conf** to ensure that **kdump** starts as expected for **sadump**.

```
kernel.panic=0
kernel.unknown_nmi_panic=1
```

In addition to the above, you must also add some options to **/etc/kdump.conf** to ensure that **kdump** behaves correctly for sadump.

In particular, you should ensure that after **kdump**, the system does not reboot. If the system reboots after **kdump** has failed to save core, then you will have no opportunity to invoke **sadump**.

You may set the **default** action in **/etc/kdump.conf** to be either *halt* or *shell* to achieve this.

```
default shell
blacklist kvm-intel
```

Important

For details on configuring your hardware for **sadump**, see the FUJITSU Server PRIMEQUEST 2000 Series Installation Manual.

Chapter 4. Analyzing a Core Dump

To determine the cause of the system crash, you can use the **crash** utility, which provides an interactive prompt very similar to the GNU Debugger (GDB). This utility allows you to interactively analyze a running Linux system as well as a core dump created by **netdump**, **diskdump**, **xendump**, or **kdump**.

4.1. Installing the crash Utility

To install the **crash** analyzing tool, execute the following command from a shell prompt as **root**:

```
yum install crash
```

In addition to **crash**, it is also necessary to install the *kernel-debuginfo* package that corresponds to your running kernel, which provides the data necessary for dump analysis. To install *kernel-debuginfo* we use the **debuginfo-install** command as **root**:

```
debuginfo-install kernel
```

For more information on how to install new packages in Red Hat Enterprise Linux using the **Yum** package manager, see the *Red Hat Enterprise Linux 7 System Administrator's Guide*.

4.2. Running the crash Utility

To start the utility, type the command in the following form at a shell prompt:

```
crash /var/crash/<timestamp>/vmcore
/usr/lib/debug/lib/modules/<kernel>/vmlinux
```

Note that the <*kernel*> version should be the same that was captured by **kdump**. To find out which kernel you are currently running, use the **uname -r** command.

Example 4.1. Running the crash utility

```
~]# crash /usr/lib/debug/lib/modules/2.6.32-69.el6.i686/vmlinux \
/var/crash/127.0.0.1-2010-08-25-08:45:02/vmcore

crash 5.0.0-23.el6
Copyright (C) 2002-2010  Red Hat, Inc.
Copyright (C) 2004, 2005, 2006  IBM Corporation
Copyright (C) 1999-2006  Hewlett-Packard Co
Copyright (C) 2005, 2006  Fujitsu Limited
Copyright (C) 2006, 2007  VA Linux Systems Japan K.K.
Copyright (C) 2005  NEC Corporation
Copyright (C) 1999, 2002, 2007  Silicon Graphics, Inc.
Copyright (C) 1999, 2000, 2001, 2002  Mission Critical Linux, Inc.
This program is free software, covered by the GNU General Public License,
and you are welcome to change it and/or distribute copies of it under
certain conditions.  Enter "help copying" to see the conditions.
This program has absolutely no warranty.  Enter "help warranty" for
details.
```

```
GNU gdb (GDB) 7.0
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-pc-linux-gnu"...

      KERNEL: /usr/lib/debug/lib/modules/2.6.32-69.el6.i686/vmlinux
    DUMPFILE: /var/crash/127.0.0.1-2010-08-25-08:45:02/vmcore  [PARTIAL
DUMP]
        CPUS: 4
        DATE: Wed Aug 25 08:44:47 2010
      UPTIME: 00:09:02
LOAD AVERAGE: 0.00, 0.01, 0.00
       TASKS: 140
    NODENAME: hp-dl320g5-02.lab.bos.redhat.com
     RELEASE: 2.6.32-69.el6.i686
     VERSION: #1 SMP Tue Aug 24 10:31:45 EDT 2010
     MACHINE: i686  (2394 Mhz)
      MEMORY: 8 GB
       PANIC: "Oops: 0002 [#1] SMP " (check log for details)
         PID: 5591
     COMMAND: "bash"
        TASK: f196d560  [THREAD_INFO: ef4da000]
         CPU: 2
       STATE: TASK_RUNNING (PANIC)

crash>
```

4.3. Displaying the Message Buffer

To display the kernel message buffer, type the **log** command at the interactive prompt.

Example 4.2. Displaying the kernel message buffer

```
crash> log
... several lines omitted ...
EIP: 0060:[<c068124f>] EFLAGS: 00010096 CPU: 2
EIP is at sysrq_handle_crash+0xf/0x20
EAX: 00000063 EBX: 00000063 ECX: c09e1c8c EDX: 00000000
ESI: c0a09ca0 EDI: 00000286 EBP: 00000000 ESP: ef4dbf24
 DS: 007b ES: 007b FS: 00d8 GS: 00e0 SS: 0068
Process bash (pid: 5591, ti=ef4da000 task=f196d560 task.ti=ef4da000)
Stack:
 c068146b c0960891 c0968653 00000003 00000000 00000002 efade5c0 c06814d0
<0> fffffffb c068150f b7776000 f2600c40 c0569ec4 ef4dbf9c 00000002 b7776000
<0> efade5c0 00000002 b7776000 c0569e60 c051de50 ef4dbf9c f196d560 ef4dbfb4
Call Trace:
 [<c068146b>] ? __handle_sysrq+0xfb/0x160
 [<c06814d0>] ? write_sysrq_trigger+0x0/0x50
 [<c068150f>] ? write_sysrq_trigger+0x3f/0x50
```

```
 [<c0569ec4>] ? proc_reg_write+0x64/0xa0
 [<c0569e60>] ? proc_reg_write+0x0/0xa0
 [<c051de50>] ? vfs_write+0xa0/0x190
 [<c051e8d1>] ? sys_write+0x41/0x70
 [<c0409adc>] ? syscall_call+0x7/0xb
Code: a0 c0 01 0f b6 41 03 19 d2 f7 d2 83 e2 03 83 e0 cf c1 e2 04 09 d0 88
41 03 f3 c3 90 c7 05 c8 1b 9e c0 01 00 00 00 0f ae f8 89 f6 <c6> 05 00 00
00 00 01 c3 89 f6 8d bc 27 00 00 00 00 8d 50 d0 83
EIP: [<c068124f>] sysrq_handle_crash+0xf/0x20 SS:ESP 0068:ef4dbf24
CR2: 0000000000000000
```

Type **`help log`** for more information on the command usage.

Note

The kernel message buffer includes the most essential information about the system crash and, as such, it is always dumped first in to the **`vmcore-dmesg.txt`** file. This is useful when an attempt to get the full **`vmcore`** file failed, for example because of lack of space on the target location. By default, **`vmcore-dmesg.txt`** is located in the **`/var/crash/`** directory.

4.4. Displaying a Backtrace

To display the kernel stack trace, type the **`bt`** command at the interactive prompt. You can use **`bt <pid>`** to display the backtrace of a single process.

Example 4.3. Displaying the kernel stack trace

```
crash> bt
PID: 5591   TASK: f196d560  CPU: 2   COMMAND: "bash"
 #0 [ef4dbdcc] crash_kexec at c0494922
 #1 [ef4dbe20] oops_end at c080e402
 #2 [ef4dbe34] no_context at c043089d
 #3 [ef4dbe58] bad_area at c0430b26
 #4 [ef4dbe6c] do_page_fault at c080fb9b
 #5 [ef4dbee4] error_code (via page_fault) at c080d809
    EAX: 00000063  EBX: 00000063  ECX: c09e1c8c  EDX: 00000000  EBP:
00000000
    DS:  007b      ESI: c0a09ca0  ES:  007b      EDI: 00000286  GS:  00e0
    CS:  0060      EIP: c068124f  ERR: ffffffff  EFLAGS: 00010096
 #6 [ef4dbf18] sysrq_handle_crash at c068124f
 #7 [ef4dbf24] __handle_sysrq at c0681469
 #8 [ef4dbf48] write_sysrq_trigger at c068150a
 #9 [ef4dbf54] proc_reg_write at c0569ec2
#10 [ef4dbf74] vfs_write at c051de4e
#11 [ef4dbf94] sys_write at c051e8cc
#12 [ef4dbfb0] system_call at c0409ad5
    EAX: ffffffda  EBX: 00000001  ECX: b7776000  EDX: 00000002
    DS:  007b      ESI: 00000002  ES:  007b      EDI: b7776000
    SS:  007b      ESP: bfcb2088  EBP: bfcb20b4  GS:  0033
    CS:  0073      EIP: 00edc416  ERR: 00000004  EFLAGS: 00000246
```

Type **help bt** for more information on the command usage.

4.5. Displaying a Process Status

To display status of processes in the system, type the **ps** command at the interactive prompt. You can use **ps *<pid>*** to display the status of a single process.

Example 4.4. Displaying the status of processes in the system

```
crash> ps
   PID    PPID  CPU   TASK    ST  %MEM     VSZ    RSS  COMM
>     0      0   0  c09dc560  RU   0.0       0      0  [swapper]
>     0      0   1  f7072030  RU   0.0       0      0  [swapper]
      0      0   2  f70a3a90  RU   0.0       0      0  [swapper]
>     0      0   3  f70ac560  RU   0.0       0      0  [swapper]
      1      0   1  f705ba90  IN   0.0    2828   1424  init
... several lines omitted ...
   5566      1   1  f2592560  IN   0.0   12876    784  auditd
   5567      1   2  ef427560  IN   0.0   12876    784  auditd
   5587   5132   0  f196d030  IN   0.0   11064   3184  sshd
>  5591   5587   2  f196d560  RU   0.0    5084   1648  bash
```

Type **help ps** for more information on the command usage.

4.6. Displaying Virtual Memory Information

To display basic virtual memory information, type the **vm** command at the interactive prompt. You can use **vm *<pid>*** to display information on a single process.

Example 4.5. Displaying virtual memory information of the current context

```
crash> vm
PID: 5591   TASK: f196d560  CPU: 2   COMMAND: "bash"
   MM       PGD      RSS    TOTAL_VM
f19b5900  ef9c6000  1648k    5084k
  VMA       START      END    FLAGS  FILE
f1bb0310    242000    260000 8000875  /lib/ld-2.12.so
f26af0b8    260000    261000 8100871  /lib/ld-2.12.so
efbc275c    261000    262000 8100873  /lib/ld-2.12.so
efbc2a18    268000    3ed000 8000075  /lib/libc-2.12.so
efbc23d8    3ed000    3ee000 8000070  /lib/libc-2.12.so
efbc2888    3ee000    3f0000 8100071  /lib/libc-2.12.so
efbc2cd4    3f0000    3f1000 8100073  /lib/libc-2.12.so
efbc243c    3f1000    3f4000 100073
efbc28ec    3f6000    3f9000 8000075  /lib/libdl-2.12.so
efbc2568    3f9000    3fa000 8100071  /lib/libdl-2.12.so
efbc2f2c    3fa000    3fb000 8100073  /lib/libdl-2.12.so
f26af888    7e6000    7fc000 8000075  /lib/libtinfo.so.5.7
f26aff2c    7fc000    7ff000 8100073  /lib/libtinfo.so.5.7
```

```
efbc211c    d83000    d8f000 8000075  /lib/libnss_files-2.12.so
efbc2504    d8f000    d90000 8100071  /lib/libnss_files-2.12.so
efbc2950    d90000    d91000 8100073  /lib/libnss_files-2.12.so
f26afe00    edc000    edd000 4040075
f1bb0a18   8047000   8118000 8001875  /bin/bash
f1bb01e4   8118000   811d000 8101873  /bin/bash
f1bb0c70   811d000   8122000 100073
f26afae0   9fd9000   9ffa000 100073
... several lines omitted ...
```

Type **help vm** for more information on the command usage.

4.7. Displaying Open Files

To display information about open files, type the **files** command at the interactive prompt. You can use **files *<pid>*** to display files opened by only one selected process.

Example 4.6. Displaying information about open files of the current context

```
crash> files
PID: 5591   TASK: f196d560  CPU: 2   COMMAND: "bash"
ROOT: /    CWD: /root
 FD    FILE     DENTRY    INODE    TYPE  PATH
  0  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
  1  efade5c0  eee14090  f00431d4  REG   /proc/sysrq-trigger
  2  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
 10  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
255  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
```

Type **help files** for more information on the command usage.

4.8. Exiting the Utility

To exit the interactive prompt and terminate **crash**, type **exit** or **q**.

Example 4.7. Exiting the crash utility

```
crash> exit
~]#
```

Appendix A. Frequently Asked Questions

Q: **What considerations need to be made for using Kdump in a clustered environment?**

A: How do I configure kdump for use with the RHEL 6, 7 High Availability Add-On? shows the options available to system administrators using the High Availability Add-On.

Q: **Kdump fails during early boot, How do I capture the boot log?**

A: If there is a problem booting the second kernel, it may be necessary to review the early boot logs, these can be obtained by enabling a serial console to the affected machine.

How do I setup serial console in RHEL7? shows the configuration needed to enable access to the early boot messages.

Q: **How do I increase the messaging from makedumpfile for debugging?**

A: In the event that **makedumpfile** fails, then it may be necessary to increase the log level to understand what is going wrong. This is different from setting the dump level and is achieved by editing **/etc/kdump.conf** and increasing the **message_level** option to **makedumpfile** on the **core_collector** line entry.

By default **makedumpfile** is set to level 7, which includes the progress indicator, common message, and error message output. Set this level to 31 to get further debugging information.

Your core_collector config line should look similar to this when set:

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

Q: **How do I debug Dracut?**

A: Sometimes **dracut** can fail to build an initramfs. If this happens, then you will need to increase the log level in **dracut** to isolate the issue.

Edit **/etc/kdump.conf** and change the **dracut_args** line to include the option **-L 5** in addition to any other dracut arguments you require.

If you have no other options configured in **dracut_args**, the result will look similar to this:

```
dracut_args -L 5
```

Q: **What methods of dumping are available for virtual machines?**

A: In most cases, the **kdump** mechanism will be sufficient for obtaining a memory dump from a machine after a crash or panic. This can be set up in the same manner as installations to bare metal.

However, in some cases, it may be necessary to work directly with the hypervisor to obtain a crash dump. There are two mechanisms available with **libvirt** to achieve this; **pvpanic** and **virsh dump**. Both of these methods are described in the Virtualization Deployment and Administration Guide.

The **pvpanic** mechanism can be found at Virtualization Deployment and Administration Guide - Setting a Panic Device.

The **virsh dump** command is discussed in Virtualization Deployment and Administration Guide -

Creating a Dump File of a Domain's Core.

Q: How do I upload a large dump file to Red Hat Support Services?

A: In some cases, it might be necessary to send a kernel crash dump file to Red Hat Global Support Services for analysis. However, the dump file can be very large, even after being filtered. Since files larger than 250 MB cannot be uploaded directly through the Red Hat Customer Portal when opening a new support case, an FTP server is provided by Red Hat for uploading large files.

The FTP server's address is **`dropbox.redhat.com`** and the files are to be uploaded in the **`/incoming/`** directory. Your FTP client needs to be set into passive mode; if your firewall does not allow this mode, you may use the **`origin-dropbox.redhat.com`** server using active mode.

Make sure that the uploaded files are compressed using a program such as **gzip** and properly and descriptively named. Using your support case number in the file name is recommended. After successfully uploading all necessary files, provide the engineer in charge of your support case with the exact file name and its SHA1 or MD5 checksum.

For more specific instructions and additional information, see How to provide files to Red Hat Support.

Q: How much time is needed for a crash dump to complete?

A: It is often necessary, for the purposes of disaster recovery planning, to know how long a dump will take to complete. However, the length of time it takes is highly dependent on the amount of memory being copied to disk and the speed of the interfaces between RAM and storage.

For any test of timings, the system must be operating under a representative load, otherwise the page exclusion choices can present a false view of kdump behavior with a fully loaded production system. This discrepancy will be observed more particularly when working with very large quantities of RAM.

Storage interfaces should also be considered in your planning when assessing time to dump. Because of network constraints, a connection dumping over **`ssh`** for example, can take longer to complete than a locally attached SATA disk.

Q: How is Kdump configured during installation?

A: You can configure **kdump** during installation with a limited set of options in kickstart or the interactive GUI.

The **kdump** configuration using the **anaconda** installation GUI is documented in the KDUMP section of the Installation Guide.

The **kickstart** syntax is:

```
%addon com_redhat_kdump [--disable,enable] [--reserve-mb=[auto,value]]
%end
```

With this add-on to Kickstart, you can disable or enable kdump functionality, optionally defining the reserved memory size, either by explicitly invoking the default option of auto (which is also the case if the entire switch is omitted) or specifying a numeric value in megabytes.

To learn how Kickstart can be used to automate system deployments, please read Kickstart Installations in the Installation Guide.

For further details about Kickstart add-on syntax, please review the Kickstart Syntax Reference in the Installation Guide.

Appendix B. Supported kdump Configurations and Targets

B.1. Memory Requirements for kdump

In order for kdump to be able to capture a kernel crash dump and save it for further analysis, a part of the system memory has to be permanently reserved for the capture kernel. The table below contains a list of minimum memory requirements for kdump based on the system's architecture and total available physical memory.

For information on how to change memory settings on the command line, see Section 2.2.1, “Configuring the Memory Usage”. For instructions on how to set up the amount of reserved memory in the graphical user interface, see Section 2.3.1, “Configuring the Memory Usage”.

Table B.1. Minimum Amount of Reserved Memory Required for kdump

Architecture	Available Memory	Minimum Reserved Memory
AMD64 and Intel 64 (**x86_64**)	2 GB and more	160 MB + 2 bits for every 4 KB of RAM. For a system with 1 TB of memory, 224 MB is the minimum (160 + 64 MB).
IBM POWER (**ppc64**)	2 GB to 4 GB	256 MB of RAM.
	4 GB to 32 GB	512 MB of RAM.
	32 GB to 64 GB	1 GB of RAM.
	64 GB to 128 GB	2 GB or RAM.
	128 GB and more	4 GB of RAM.
IBM System z (**s390x**)	2 GB and more	160 MB + 2 bits for every 4 KB of RAM. For a system with 1 TB of memory, 224 MB is the minimum (160 + 64 MB).

B.2. Minimum Threshold for Automatic Memory Reservation

On some systems, it is possible to allocate memory for kdump automatically, either by using the **crashkernel=auto** parameter in the bootloader's configuration file, or by enabling this option in the graphical configuration utility. For this automatic reservation to work, however, a certain amount of total memory needs to be available in the system. This amount differs based on the system's architecture.

The table below lists the thresholds for automatic memory allocation. If the system has less memory than specified in the table, memory will have to be reserved manually.

For information on how to change these settings on the command line, see Section 2.2.1, “Configuring the Memory Usage”. For instructions on how to change the amount of reserved memory in the graphical user interface, see Section 2.3.1, “Configuring the Memory Usage”.

Table B.2. Minimum Amount of Memory Required for Automatic Memory Reservation

Architecture	Required Memory
AMD64 and Intel 64 (**x86_64**)	2 GB
IBM POWER (**ppc64**)	2 GB
IBM System z (**s390x**)	4 GB

B.3. Supported kdump Targets

When a kernel crash is captured, the core dump can be either written directly to a device, stored as a file on a local file system, or sent over a network. The table below contains a complete list of dump targets that are currently supported or explicitly unsupported by kdump.

For information on how to configure the target type on the command line, see Section 2.2.2, “Configuring the kdump Type”. For information on how to do so in the graphical user interface, see Section 2.3.2, “Configuring the kdump Type”.

Table B.3. Supported kdump Targets

Type	Supported Targets	Unsupported Targets
Raw device	All locally attached raw disks and partitions.	—
Local file system	**ext2**, **ext3**, **ext4**, **btrfs** and **xfs** file systems on directly attached disk drives, hardware RAID logical drives, LVM devices, and **mdraid** arrays.	Any local file system not explicitly listed as supported in this table, including the **auto** type (automatic file system detection).
Remote directory	Remote directories accessed using the **NFS** or **SSH** protocol over **IPv4**.	Remote directories on the **rootfs** file system accessed using the **NFS** protocol.
	Remote directories accessed using the **iSCSI** protocol over both hardware and software initiators.	Remote directories accessed using the **iSCSI** protocol on **be2iscsi** hardware.
	Multipath-based storages.	—
	—	Remote directories accessed over **IPv6**.
		Remote directories accessed using the **SMB**/**CIFS** protocol.
		Remote directories accessed using the **FCoE** (*Fibre Channel over Ethernet*) protocol.
		Remote directories accessed using wireless network interfaces.

B.4. Supported kdump Filtering Levels

To reduce the size of the dump file, kdump uses the **makedumpfile** core collector to compress the data and optionally leave out irrelevant information. The table below contains a complete list of filtering levels that are currently supported by the **makedumpfile** utility.

For instructions on how to configure the core collector on the command line, see Section 2.2.3, “Configuring the Core Collector”. For information on how to do so in the graphical user interface, see Section 2.3.3, “Configuring the Core Collector”.

Table B.4. Supported Filtering Levels

Option	Description
1	Zero pages
2	Cache pages
4	Cache private
8	User pages
16	Free pages

Note

The **makedumpfile** command supports removal of transparent huge pages and hugetlbfs pages on Red Hat Enterprise Linux 7.3 and later. Both these types of hugepages are considered User Pages and will be removed using the **-8** level.

B.5. Supported Default Actions

By default, when kdump fails to create a core dump, it mounts the root file system and attempts to save the core locally. You can, however, configure kdump to perform a different operation in case it fails to save the core dump to the primary target. The table below lists all default actions that are currently supported by kdump.

For detailed information on how to set up the default action on the command line, see Section 2.2.4, “Configuring the Default Action”. For information on how to do so in the graphical user interface, see Section 2.3.4, “Configuring the Default Action”.

Table B.5. Supported Default Actions

Option	Description
dump_to_rootfs	Attempt to save the core dump to the root file system. This option is especially useful in combination with a network target: if the network target is unreachable, this option configures kdump to save the core dump locally. The system is rebooted afterwards.
reboot	Reboot the system, losing the core dump in the process.
halt	Halt the system, losing the core dump in the process.
poweroff	Power off the system, losing the core dump in the process.
shell	Run a shell session from within the initramfs, allowing the user to record the core dump manually.

B.6. Estimating Kdump Size

When planning and building your **kdump** environment it is necessary to know how much space is required for the dump file before one is produced. The **makedumpfile** command can help with this.

The **--mem-usage** option provides a useful report about excludable pages, that can be used to determine which dump level you want to assign. This command should be run when the system is under representative load, otherwise **makedumpfile** will return a smaller value than is expected in your production environment.

```
[root@hostname ~]# 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`** command reports in **pages**. This means that you must calculate the size of memory in use against the kernel page size, which in the Red Hat Enterprise Linux kernel, is 4 kilobytes (4096 bytes).

Appendix C. Portal Labs relevant to Kdump

The Portal Labs are small web applications that can help system administrators perform several system tasks. There are currently two labs focused on Kdump. The Kdump Helper and the Kernel Oops Analyzer.

C.1. Kdump Helper

The Kdump Helper is a series of questions and actions that assist in preparing the configuration files for **kdump**.

The Lab's workflow includes steps for both clustered and standalone environments.

C.2. Kernel Oops Analyzer

The Kernel Oops Analyzer is a tool to process Oops messages and search for known solutions without having to unwind the crash dump stack.

The Kernel Oops Analyzer uses information from **makedumpfile** to compare the oops message from a crashed machine with known issues in the knowledge base. This can enable System Administrators to rule out known issues quickly after an unexpected outage, and before opening a support ticket for a further analysis.

Appendix D. Revision History

Revision 1.3-3 **Fri Jul 28 2017** **Mark Flitter**

Document version for 7.4 GA publication.

Revision 1.3-2 **Fri Nov 4 2016** **Mark Flitter**

Version for 7.3 GA publication.

Revision 1.2-9 **Thu 18 Aug 2016** **Mark Flitter**

Updates for 7.3 Beta, addition of Z Series specific notes and estimating the size of vmcores.

Revision 1.2-0 **Fri 06 Mar 2015** **Petr Bokoč**

Update fixing several issues such as wrong information for memory configuration and outdated screenshots

Revision 1.1-3 **Wed 18 Feb 2015** **Petr Bokoč**

Red Hat Enterprise Linux 7.1 GA release of the Kernel Crash Dump Guide.

Revision 1.1-0 **Fri 05 Dec 2014** **Petr Bokoč**

Red Hat Enterprise Linux 7.1 Beta release of the Kernel Crash Dump Guide.

Revision 1.0-0 **Mon 02 Jun 2014** **Jaromír Hradílek**

Red Hat Enterprise Linux 7.0 GA release of the Kernel Crash Dump Guide.

Revision 0.0-8 **Thu Jan 17 2013** **Jaromír Hradílek**

Initial creation of the book.