Linux on System z

# Using the Dump Tools on SUSE Linux Enterprise Server 12

SC34-2746-00

Linux on System z

# Using the Dump Tools on SUSE Linux Enterprise Server 12

SC34-2746-00

**Note**
Before using this information and the product it supports, read the information in "Notices" on page 71.

This edition applies to SUSE Linux Enterprise Server 12 on IBM System z, and to all subsequent releases and modifications until otherwise indicated in new editions.

# Contents

# About this document

This book describes tools for obtaining dumps of Linux for IBM® System z® instances running SUSE Linux Enterprise Server 12. This book describes how to use DASD, tape, and SCSI dump devices, as well as how to use VMDUMP.

Unless stated otherwise, all z/VM® related information in this document assumes a current z/VM version, see www.ibm.com/vm/techinfo.

In this publication, System z is taken to include all IBM mainframe systems supported by SUSE Linux Enterprise Server 12 for System z. In particular, this includes IBM zEnterprise® BC12 (zBC12), IBM zEnterprise EC12 (zEC12), IBM zEnterprise 196 (z196), and IBM zEnterprise 114 (z114) mainframes.

You can find the latest version of this document on developerWorks® at www.ibm.com/developerworks/linux/linux390/documentation_suse.html

## Authority

Most of the tasks described in this document require a user with root authority. In particular, writing to most of the described sysfs attributes requires root authority.

Throughout this document, it is assumed that you have root authority.

# Other relevant Linux on IBM System z publications

Several Linux on IBM System z publications for SUSE Linux Enterprise Server 12 are available on developerWorks.

You can find the latest versions of these publications on developerWorks at www.ibm.com/developerworks/linux/linux390/documentation_suse.html or on IBM Knowledge Center at ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_suse.html

- *Device Drivers, Features, and Commands on SUSE Linux Enterprise Server 12*, SC34-2745
- *Kernel Messages on SUSE Linux Enterprise Server 12*, SC34-2747

For each of the following publications, you can find the version that most closely reflects SUSE Linux Enterprise Server 12:

- *How to use FC-attached SCSI devices with Linux on System z*, SC33-8413
- *libica Programmer's Reference*, SC34-2602
- *Exploiting Enterprise PKCS #11 using openCryptoki*, SC34-2713
- *Secure Key Solution with the Common Cryptographic Architecture Application Programmer's Guide*, SC33-8294
- *Linux on System z Troubleshooting*, SC34-2612
- *Linux Health Checker User's Guide*, SC34-2609
- *How to Improve Performance with PAV*, SC33-8414
- *How to Set up a Terminal Server Environment on z/VM*, SC34-2596

## Device nodes used in this publication

There can be multiple device nodes for the same device.

The DASD and tape examples in this publication use the standard device nodes. You can also use the device nodes that udev creates for you. The SCSI examples use multipath device nodes.

# Chapter 1. Tools overview

Different tools can be used for obtaining dumps for instances of SUSE Linux Enterprise Server 12 running on IBM System z mainframes.

You can use the dump analysis tool **crash** to analyze a dump. Depending on your service contract, you might also want to send a dump to IBM support to be analyzed.

Table 1 summarizes the available dump tools:

*Table 1. Dump tools summary*

| Dump aspect | kdump | DASD | Multi-volume DASD | SCSI | Tape | VMDUMP | Live-system dump with zgetdump |
|---|---|---|---|---|---|---|---|
| **Environment** | z/VM and LPAR | z/VM and LPAR | z/VM and LPAR | z/VM and LPAR | z/VM and LPAR | z/VM only | z/VM and LPAR |
| **z/VM NSS** | No | Yes (see note 3) | Yes (see note 3) | Yes (see note 3) | Yes (see note 3) | Yes | Yes |
| **System size** (See also "Dump size" on page 2) | Large | Small | Large | Large | Large | Small | Large |
| **Speed** | Fast | Fast | Fast | Fast | Slow | Slow | Fast |
| **Medium** | Any available medium | ECKD™ or FBA (see note 1) DASD | ECKD DASD | SCSI partition | Tape cartridges | z/VM reader | Any available medium |
| **Compression possible** | While writing | No | No | No | Yes (See "Dump size" on page 2) | No | No |
| **Dump filtering possible** | While writing | When copying | When copying | When copying | When copying | When copying | No |
| **Disruptive** (see note2) | Yes | Yes | Yes | Yes | Yes | No | No |
| **Stand-alone** | No | Yes | Yes | Yes | Yes | No | No |

**Note:**

1. SCSI disks can be emulated as FBA disks. This dump method can, therefore, be used for SCSI-only z/VM installations.
2. In this context, disruptive means that the dump process kills a running operating system.
3. As of z/VM 6.3, you can use the `nssdata` IPL option to create a dump for a Linux instance that runs from a z/VM NSS.

### Dump size

The dump size depends on the size of the system for which the dump is to be created. All dump methods require persistent storage space to hold the kernel and user space of this system.

**kdump**
Initially uses the memory of the Linux instance for which a dump is to be created, and so supports any size. A persistent copy can be written to any medium of sufficient size. While writing, the dump size can be reduced through page filtering and compression.

**DASD**
Depends on the disk size. For example, ECKD model A provides several hundreds of GB, depending on the storage server model.

**Multivolume DASD**
Can be up to the combined size of 32 DASD partitions.

**SCSI** Depends on the capacity of the SCSI disk and which other data it contains.

**Tape** Depends on the tape drive. For example, IBM TotalStorage Enterprise Tape System 3592 supports large dumps and also offers hardware compression.

**VMDUMP**
Depends on the available spool space. The slow dump speed can lead to very long dump times for large dumps. Although technically possible, the slow dump speed makes VMDUMP unsuitable for large dumps.

**zgetdump live-system dump**
The dump can be written to any medium of sufficient size.

See Chapter 8, “Handling large dumps,” on page 27 for information specific to large dumps.

## The kdump tool

The kdump tool is made available through a Linux kernel and initial RAM disk that are preloaded in memory, along with a production system.

You do not have to install kdump on a dedicated dump device. The kdump system can access the memory that contains the dump of the production system through a procfs file.

Filtering out extraneous memory pages and compression can take place while the dump is written to persistent storage or transferred over a network. The smaller dump size can significantly reduce the write or transfer time, especially for large production systems.

Because kdump can write dumps through a network, existing file system facilities can be used to prevent multiple dumps from being written to the same storage space. Sharing space for dumps across an enterprise is possible without the more complex setups described in Chapter 10, “Sharing dump devices,” on page 35.

## Stand-alone tools

Stand-alone tools are installed on a device on which you perform an IPL. Different tools are available depending on the device type.

Four stand-alone dump tools are shipped in the s390-tools package as part of the **zipl** package:

- DASD dump tool for dumps on a single DASD device
- Multi-volume DASD dump tool for dumps on a set of ECKD DASD devices
- Tape dump tool for dumps on (channel-attached) tape devices
- SCSI disk dump tool for dumps on SCSI disks

You need to install these tools on the *dump device*. A dump device is used to initiate a stand-alone dump by IPL-ing the device. It must have a stand-alone dump tool installed and should provide enough space for the dump.

Typically, the system operator initiates a dump after a system crash, but you can initiate a dump at any time. To initiate a dump, you must IPL the dump device. This process is destructive, that is, the running Linux operating system is killed. The IPL process writes the system memory to the IPL device (DASD device, tape, or SCSI disk).

You can configure a dump device that is automatically used when a kernel panic occurs. For more information, see "dumpconf - Configure panic or PSW restart action" on page 58.

**GRUB 2 usage:** You cannot use GRUB 2 to install the standalone dump tools. You must use the **zipl** command as described in this document for the dump tools.

For more information about **zipl**, see "zipl - Prepare devices for stand-alone dump" on page 51.

## The VMDUMP tool

The **VMDUMP** tool is a part of z/VM and does not need to be installed separately.

Dumping with VMDUMP is not destructive. If you dump an operating Linux instance, the instance continues running after the dump is completed.

VMDUMP can also create dumps for z/VM guests that use z/VM named saved systems (NSS).

Do not use VMDUMP to dump large z/VM guests; the dump process is very slow. Dumping 1 GB of storage can take up to 15 minutes depending on the used storage server and z/VM version.

For more information about VMDUMP, see *z/VM CP Commands and Utilities Reference*, SC24-6175.

## Live-system dump

You can create a kernel dump from a live system without disruption.

Use the **zgetdump** tool that is shipped with the s390-tools package to create a kernel dump while the Linux system continues running. No dump device must be prepared, because the `/dev/crash` device node is used to create the dump.

## Dump methods compared

The process for preparing a dump device and obtaining a dump differs for the available dump methods.

*Table 2. Comparing the dump methods*

| Dump aspect | kdump | Stand-alone tools | VMDUMP | Live-system dump with zgetdump |
|---|---|---|---|---|
| Preparation | Reserve memory with the `crashkernel=` kernel parameter<br><br>Load the kdump kernel and the initial RAM disk into the memory of the production system. Use **kexec** or `systemctl start kdump` | Write the stand-alone dump tool to the dump device (**zipl**)<br><br>Define the panic shutdown action (**dumpconf**) | Define the panic shutdown action (**dumpconf**) | None |
| Dump trigger | **Automatic:** Kernel panic<br>**Initiated by operator:** PSW restart | **Automatic:** Kernel panic<br>**Initiated by operator:** IPL of the dump device | **Automatic:** Kernel panic<br>**Initiated by operator:** z/VM CP **VMDUMP** command | **Initiated by operator:** **zgetdump** invocation |
| Initial dump space | Memory | Dump device | Spool device | Memory |
| Accessing the initial dump | Through `/proc/vmcore` from the kdump instance (automatically done by kdump initrd) | Using **zgetdump** from a new Linux instance | Using **vmur -c** from a new Linux instance | Through `/dev/crash` |
| Copying the initial dump to the final dump store (and releasing the initial dump space) | Copied from the kdump instance to any available storage (automatically done by kdump initrd) | Copied from the new Linux instance to any available storage | Copied from the new Linux instance to any available storage | Copied from the current Linux instance to any available storage |
| Optional: Filtering the initial dump | Using `/proc/vmcore` and **makedumpfile** on the kdump instance (automatically done by kdump initrd) | Using **zgetdump** and **makedumpfile** on the new Linux instance | Using **zgetdump** and **makedumpfile** on the new Linux instance | Not recommended |

# Chapter 2. Using kdump

You can use kdump to create system dumps for instances of SUSE Linux Enterprise Server.

### Advantages of kdump

kdump offers these advantages over other dump methods:

- While writing the dump, you can filter out extraneous pages and compress the dump, and so handle large dumps in a short time.
- When writing dumps over a network, you can use existing file system facilities to share dump space without special preparations.

### Shortcomings of kdump

kdump has these drawbacks:

- kdump is not as reliable as the stand-alone dump tools. For critical systems, you can set up stand-alone dump tools as a backup, in addition to the kdump configuration (see "Failure recovery and backup tools" on page 7).
- kdump cannot dump a z/VM named saved system (NSS).
- For production systems that run in LPAR mode, kdump consumes memory (see "Memory consumption" on page 6).

## How kdump works on System z

You can set up kdump according to your needs.

With kdump, you do not need to install a dump tool on the storage device that is to hold a future dump. Instead, you use a kdump kernel, a Linux instance that controls the dump process.

The kdump kernel occupies a reserved memory area within the memory of the production system for which it is set up. The reserved memory area is defined with the `crashkernel=` kernel parameter. After the production system is started, the kdump init service loads the kdump kernel and its initial RAM disk (initrd) into the reserved memory area with the **kexec** tool.

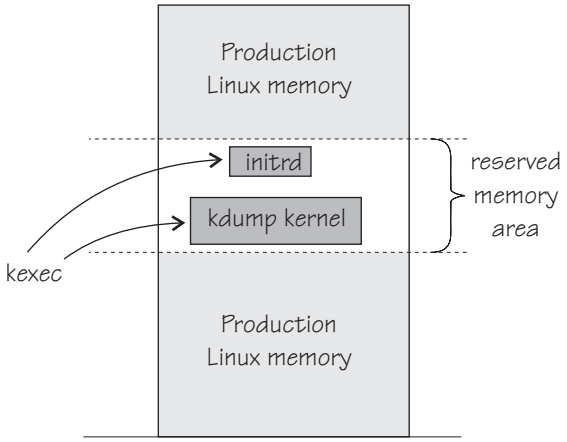



*Figure 1. Running production system with preloaded kdump kernel and initial RAM disk*

At the beginning of the dump process, the reserved memory area is exchanged with the lower memory regions of the crashed production system. The kdump system is then started and runs entirely in the memory that was exchanged with the reserved area. From the running kdump kernel, the memory of the crashed production system can be accessed as a virtual file, `/proc/vmcore`.

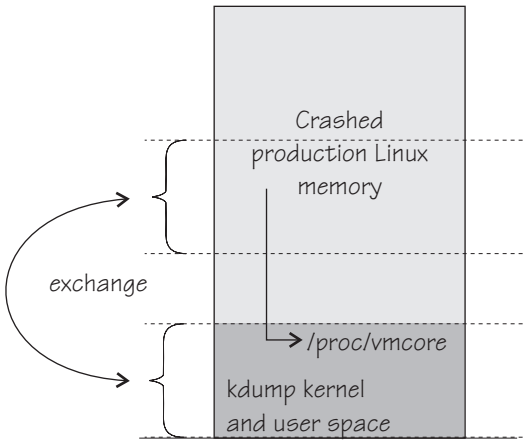



*Figure 2. Running kdump kernel*

This process is fast because the kdump kernel is started from memory, and no dump data needs to be copied up to this stage. For SUSE Linux Enterprise Server, the **`makedumpfile`** tool in the kdump initrd writes a filtered and compressed version of the dump to a file on persistent storage, locally or over a network. Again, this method saves time because the dump is reduced in size while it is written or transferred.

By default, kdump initrd automatically IPLs the production system after the dump is written.

## Memory consumption

Although each Linux instance must be defined with additional memory for kdump, the total memory consumption for your z/VM installation does not increase considerably.

On most architectures, the inactive kdump system consumes the entire memory that is reserved with the `crashkernel=` kernel parameter.

For Linux on z/VM, only the kdump image and its initial RAM disk consume actual memory. The remaining reserved memory is withheld by the z/VM hypervisor until it is required in exchange for the lower memory region of the crashed production system.

Because the kdump image and initial RAM disk are not used during regular operations, z/VM swaps them out of memory some time after IPL. Thereafter, no real memory is occupied for kdump until it is booted to handle a dump.

For Linux in LPAR mode, the reserved memory area consumes real memory.

### Failure recovery and backup tools

If kdump fails, stand-alone dump tools or VMDUMP can be used as backup tools. Backup tools are, typically, set up only for vital production systems.

Because of being preloaded into memory, there is a small chance that parts of kdump are overwritten by malfunctioning kernel functions. The kdump kernel is, therefore, booted only if a checksum assures the integrity of the kdump kernel and initial RAM disk. This failure can be recovered automatically by setting up a backup dump tool with the **dumpconf** service or through a backup dump that is initiated by a user. See "dumpconf - Configure panic or PSW restart action" on page 58.

A second possible failure is the kdump system itself crashing during the dump process. This failure occurs, for example, if the reserved memory area is too small for the kdump kernel and user space. For this failure, initiate a backup dump, which captures data for both the crashed production system and the crashed kdump kernel. You can separate this data with the **zgetdump --select** option. See "zgetdump - Copy and convert kernel dumps" on page 53.

## Setting up kdump

SUSE Linux Enterprise Server 12 provides two ways of setting up kdump.

### About this task

You can choose between the following methods of setting up kdump:

- The kdump configuration utility in YaST: Graphical tool with kdump configuration options.
- Manually, using the configuration file `/etc/sysconfig/kdump.conf`. For a configuration example, see the chapter about kexec and kdump in the *System Analysis and Tuning Guide*, available at

  `www.suse.com/documentation`

### What to do next

You can check whether kdump is set up with the lsshut command. If the command lists kdump as a shutdown action, kdump is configured. For example:

```
# lsshut
Trigger          Action
=======================
Halt             stop
Power off        stop
Reboot           reipl
Restart          kdump,stop
Panic            kdump,stop
```

As a backup, you can set up a stand-alone dump tool in addition to kdump. See “dumpconf - Configure panic or PSW restart action” on page 58 about how to run a backup tool automatically, if kdump fails.

## How kdump is triggered

A kernel panic automatically triggers the dump process with kdump. If this automation fails, there are other methods you can use to trigger the dump process.

### About this task

With kdump installed, a kernel panic or PSW restart trigger kdump rather than the shutdown actions defined in `/sys/firmware`. The definitions in `/sys/firmware` are used only if an integrity check for kdump fails (see also “Failure recovery and backup tools” on page 7 and “dumpconf - Configure panic or PSW restart action” on page 58).

### Procedure

Use one of the methods according to your environment:

- For Linux in LPAR mode: Run the **PSW restart** task on the HMC. See “HMC or SE” on page 43 for details.
- For Linux on z/VM: Run the z/VM CP **system restart** command. For example, issue this command from a 3270 terminal:

  ```
  #cp system restart
  ```

- For Linux on z/VM: Configure the z/VM watchdog to trigger kdump. Set **system restart** as the z/VM CP command to be issued if the watchdog detects that the Linux instance has failed. See *Device Drivers, Features, and Commands on SUSE Linux Enterprise Server 12*, SC34-2745 about how to configure the z/VM watchdog.

### Results

The dump process loads the kdump kernel. Depending on the kdump configuration `/proc/vmcore` is copied and filtered by the kdump initrd, and then the production system is rebooted.

### What to do next

Verify that your production system is up and running again. Send the created dump to your support organization.

# Chapter 3. Using a DASD dump device

To use a DASD dump device, you need to install the stand-alone DASD dump tool and perform the dump process. Then, copy the dump to a file in a Linux file system.

DASD dumps are written directly to a DASD partition that is not formatted with a file system. The following DASD types are supported:

- ECKD DASDs
  - 3380
  - 3390
- FBA DASDs

## Installing the DASD dump tool

Install the DASD dump tool on an unused DASD partition. Dumps are written to this partition.

### Before you begin

You need an unused DASD partition with enough space (memory size + 10 MB) to hold the system memory. If the system memory exceeds the capacity of a single DASD partition, use the multivolume dump tool, see Chapter 4, "Using DASD devices for multi-volume dump," on page 13.

**GRUB 2 usage:** You cannot use GRUB 2 to install the standalone dump tools. You must use the **zipl** command as described in this document for the dump tools.

### About this task

The examples in assume that `/dev/dasdc` is the dump device and that you want to dump to the first partition `/dev/dasdc1`.

The steps that you need to perform for installing the DASD dump tool depend on your type of DASD, ECKD or FBA:

- If you are using an ECKD-type DASD, perform all three of the following steps.
- If you are using an FBA-type DASD, skip steps 1 and 2 and perform step 3 only.

### Procedure

1. ECKD only: Format your DASD with **dasdfmt**. Use a block size of 4 KB. For example:

```
# dasdfmt -f /dev/dasdc -b 4096
```

2. ECKD only: Create a partition with **fdasd**. The partition must be sufficiently large (the memory size + 10 MB). For example:

```
# fdasd /dev/dasdc
```

3. Install the dump tool with the **zipl** command. Specify the dump device on the command line. For example:

```
# zipl -d /dev/dasdc1
```

**Note:** When you use a DASD of type ECKD that is formatted with the traditional Linux disk layout ldl, the dump tool must be reinstalled with **zipl** after each dump.

## Initiating a DASD dump

You can initiate a dump from a DASD device.

### Procedure

To obtain a dump with the DASD dump tool, perform the following main steps:

1. Stop all CPUs.
2. Store status on the IPL CPU.
3. IPL the dump tool on the IPL CPU.

   **Note:** Do not clear storage!

   The dump process can take several minutes depending on the device type you are using and the amount of system memory. After the dump completes, the IPL CPU should go into disabled wait.

   The following PSW indicates that the dump process completed successfully:

   ```
   (64-bit) PSW: 00020000 80000000 00000000 00000000
   ```

   Any other disabled wait PSW indicates an error.

   After the dump tool is IPLed, messages that indicate the progress of the dump are written to the console:

   ```
   Dumping 64 bit OS
   00000032 / 00000256 MB
   00000064 / 00000256 MB
   00000096 / 00000256 MB
   00000128 / 00000256 MB
   00000160 / 00000256 MB
   00000192 / 00000256 MB
   00000224 / 00000256 MB
   00000256 / 00000256 MB
   Dump successful
   ```

### Results

You can IPL Linux again.

See Appendix A, “Examples for initiating dumps,” on page 41 for more details.

## Copying the dump from DASD with zgetdump

You can copy a DASD dump to a file system by using the **zgetdump** tool.

### About this task

By default, the **zgetdump** tool takes the dump device as input and writes its contents to standard output. To write the dump to a file system, you must redirect the output to a file.

## Procedure

Assuming that the dump is on DASD device `/dev/dasdc1` and you want to copy it to a file named `dump.elf`:

```
# zgetdump /dev/dasdc1 > dump.elf
```

## What to do next

You can use **zgetdump** to display information about the dump. See "Checking whether a DASD dump is valid and printing the dump header" on page 57 for an example.

For general information about **zgetdump**, see "zgetdump - Copy and convert kernel dumps" on page 53 or the man page.

# Chapter 4. Using DASD devices for multi-volume dump

You can handle large dumps, up to the combined size of 32 DASD partitions, by creating dumps across multiple volumes.

## Before you begin

You need to prepare a set of ECKD DASD devices for a multivolume dump, and install the stand-alone dump tool on each DASD device that is involved. Then, perform the dump process, and copy the dump to a file in a Linux file system.

**GRUB 2 usage:** You cannot use GRUB 2 to install the standalone dump tools. You must use the **zipl** command as described in this document for the dump tools.

## About this task

You can specify up to 32 partitions on ECKD DASD volumes for a multivolume dump. The dump tool is installed on each volume involved. The volumes must be:

- In subchannel set 0.
- Formatted with the compatible disk layout (cdl, the default option when using the **dasdfmt** command.)

You must specify block size 4096 for **dasdfmt**.

For example, Figure 3 shows three DASD volumes, `dasdb`, `dasdc`, and `dasdd`, with four partitions selected to contain the dump. To earmark the partition for dump, a dump signature is written to each partition.

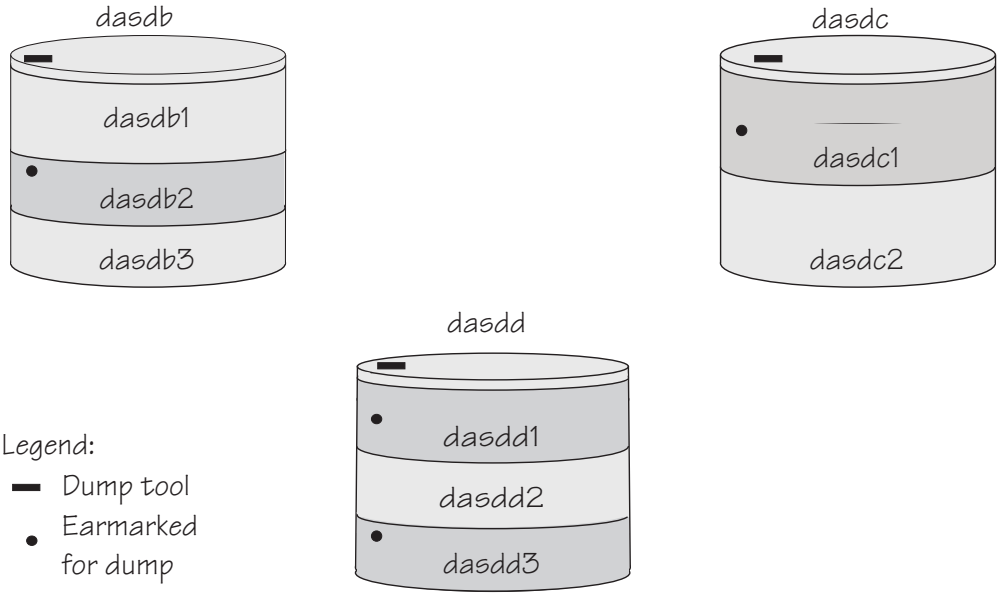



*Figure 3. Three DASD volumes with four partitions for a multivolume dump*

The partitions need to be listed in a configuration file, for example:

```
/dev/dasdb2
/dev/dasdc1
/dev/dasdd1
/dev/dasdd3
```

See “Installing the multi-volume DASD dump tool” for how to create the configuration file. All three DASD volumes are prepared for IPL; regardless of which you use the result is the same.

The following sections will take you through the entire process of creating a multi-volume dump.

## Installing the multi-volume DASD dump tool

This example shows how to perform the dump process on two partitions, `/dev/dasdc1` and `/dev/dasdd1`, which reside on ECKD volumes `/dev/dasdc` and `/dev/dasdd`.

### About this task

Assume that the corresponding bus IDs (as displayed by **lsdasd**) are 0.0.4711 and 0.0.4712, so the respective device numbers are 4711 and 4712.

### Procedure

1. Format both dump volumes with **dasdfmt**. The command shown uses the default cdl (compatible disk layout) and specifies a block size of 4KB.

   ```
   # dasdfmt -f /dev/dasdc -b 4096
   # dasdfmt -f /dev/dasdd -b 4096
   ```

2. Create the partitions with **fdasd**. The sum of the partition sizes must be sufficiently large (the memory size + 10 MB):

   ```
   # fdasd /dev/dasdc
   # fdasd /dev/dasdd
   ```

3. Create a file named `mvdump.conf` containing the device nodes of the two partitions, separated by one or more line feed characters (0x0a). The file's contents are as follows:

   ```
   /dev/dasdc1
   /dev/dasdd1
   ```

4. Prepare the volumes using the **zipl** command. Specify the dump list on the command line.

   **Command line example:**

   ```
   # zipl -M mvdump.conf
    Dump target: 2 partitions with a total size of 1234 MB.
    Warning: All information on the following partitions will be lost!
    /dev/dasdc1
    /dev/dasdd1
    Do you want to continue creating multi-volume dump partitions (y/n)?
   ```

### Results

Now the two volumes /dev/dasdc and /dev/dasdd with device numbers 4711 and 4712 are prepared for a multi-volume dump. Use the **--device** option of **zgetdump** to display information about these volumes:

```
# zgetdump -d /dev/dasdc
Dump device info:
  Dump tool.........: Multi-volume DASD dump tool
  Version...........: 2
  Architecture......: s390x (64 bit)
  Dump size limit...: none
  Force specified...: no

  Volume 0: 0.0.4711 (online/valid)
  Volume 1: 0.0.4712 (online/valid)
```

During **zipl** processing both partitions were earmarked for dump with a valid dump signature. The dump signature ceases to be valid when data other than dump data is written to the partition. For example, writing a file system to the partition overwrites the dump signature. Before writing memory to a partition, the dump tool checks the partition's signature and exits if the signature is invalid. Thus any data inadvertently written to the partition is protected.

You can circumvent this protection, for example, if you want to use a swap space partition for dumping, by using the **zipl** command with the **--force** option. This option inhibits the dump signature check, and any data on the device is overwritten. Exercise great caution when using the force option.

The **zipl** command also takes a size specification, see Appendix B, "Obtaining a dump with limited size," on page 49. For more details about the **zipl** command, see "zipl - Prepare devices for stand-alone dump" on page 51.

## Initiating a multi-volume DASD dump

After preparing the DASD volumes, you can initiate a multi-volume dump by performing an IPL from one of the prepared volumes.

### Procedure

To obtain a dump with the multivolume DASD dump tool, perform the following main steps:

1. Stop all CPUs.
2. Store status on the IPL CPU.
3. IPL the dump tool using one of the prepared volumes, either 4711 or 4712.

   **Note:** Do not clear storage!

   The dump process can take several minutes depending on each volume's block size and the amount of system memory. After the dump has completed, the IPL CPU should go into disabled wait.

   The following PSW indicates that the dump process has completed successfully:

   ```
   (64-bit) PSW: 00020000 80000000 00000000 00000000
   ```

   Any other disabled wait PSW indicates an error.

   After the dump tool is IPLed, messages that indicate the progress of the dump are written to the console:

```
Dumping 64 bit OS
Dumping to: 4711
00000128 / 00001024 MB
00000256 / 00001024 MB
00000384 / 00001024 MB
00000512 / 00001024 MB
Dumping to: 4712
00000640 / 00001024 MB
00000768 / 00001024 MB
00000896 / 00001024 MB
00001024 / 00001024 MB
Dump successful
```

4. You can IPL Linux again.

## Copying a multi-volume dump to a file

Use the **zgetdump** command to copy the multi-volume dump.

### About this task

This example assumes that the two volumes /dev/dasdc and /dev/dasdd (with device numbers 4711 and 4712) contain the dump. Dump data is spread along partitions /dev/dasdc1 and /dev/dasdd1.

### Procedure

Use **zgetdump** without any options to copy the dump parts to a file:

```
# zgetdump /dev/dasdc > dump.elf
Format Info:
Source: s390mv
Target: elf

Copying dump:
00000000 / 00001024 MB
00000171 / 00001024 MB
00000341 / 00001024 MB
00000512 / 00001024 MB
00000683 / 00001024 MB
00000853 / 00001024 MB
00001024 / 00001024 MB

Success: Dump has been copied
```

If you want to only check the validity of the multivolume dump rather than copying it to a file, use the **--info** option with **zgetdump**. See "Checking whether a DASD dump is valid and printing the dump header" on page 57 for an example.

# Chapter 5. Using a tape dump device

You can use a channel-attached tape as a dump device. To use a tape, you need to install the stand-alone tape dump tool and perform the dump process. Then, copy the dump to a file in a Linux file system.

The following tape devices are supported:
- 3480
- 3490
- 3590
- 3592

The following sections take you through the entire process of creating a dump on a tape device.

## Installing the tape dump tool

Install the tape dump tool on the tape that is to hold the dump.

### Before you begin

Have enough empty tapes ready to hold the system memory (memory size + 10 MB).

**GRUB 2 usage:** You cannot use GRUB 2 to install the standalone dump tools. You must use the **zipl** command as described in this document for the dump tools.

### About this task

The examples assume that `/dev/ntibm0` is the tape device that you want to dump to.

### Procedure

1. Insert an empty dump cartridge into your tape device.
2. Ensure that the tape is rewound.
3. Install the dump tool by using the **zipl** command. Specify the dump device on the command line. For example:

```
# zipl -d /dev/ntibm0
```

## Initiating a tape dump

Initiate a tape dump by performing an IPL on the IPL CPU.

### Procedure

To obtain a dump with the tape dump tool, perform the following main steps:

1. Ensure that the tape is rewound.
2. Stop all CPUs.
3. Store status on the IPL CPU.
4. IPL the dump tool on the IPL CPU.

**Note:** Do not clear storage!

The dump tool writes the number of dumped MB to the tape drive message display.

The dump process can take several minutes, depending on the device type you are using and the amount of system memory available. When the dump is complete, the message `dump*end` is displayed and the IPL CPU goes into disabled wait.

The following PSW indicates that the dump process completed successfully:

```
(64-bit) PSW: 00020000 80000000 00000000 00000000
```

Any other disabled wait PSW indicates an error.

After the dump tool is IPLed, messages that indicate the progress of the dump are written to the console:

```
Dumping 64 bit OS
00000032 / 00000256 MB
00000064 / 00000256 MB
00000096 / 00000256 MB
00000128 / 00000256 MB
00000160 / 00000256 MB
00000192 / 00000256 MB
00000224 / 00000256 MB
00000256 / 00000256 MB
Dump successful
```

5. You can IPL Linux again.

### What to do next

See Appendix A, "Examples for initiating dumps," on page 41 for more details.

## Tape display messages

Messages might be shown on the tape display.

### Messages

***number***
The number of MB dumped.

**dump*end**
The dump process ended successfully.

## Copying the dump from tape

You can copy a tape dump to a file system by using the **zgetdump** tool.

### Before you begin

The **mt** utility must be installed.

## Preparing the dump tape

You need to rewind the tape, and find the correct position on the tape to start copying from.

### About this task

Use the **mt** tool to manipulate the tape.

### Procedure

1. Rewind the tape.

   For example:

   ```
   # mt -f /dev/ntibm0 rewind
   ```

2. Skip the first file on the tape (this file is the dump tool itself).

   For example:

   ```
   # mt -f /dev/ntibm0 fsf
   ```

## Using the zgetdump tool to copy the dump

Use the **zgetdump** tool to copy the dump file from the tape to a file system.

### Before you begin

The tape must be in the correct position (see "Preparing the dump tape" on page 18).

### About this task

By default, the **zgetdump** tool takes the dump device as input and writes its contents to standard output. To write the dump to a file system, you must redirect the output to a file.

The example assumes that the dump is on tape device `/dev/ntibm0`.

### Procedure

Copy the dump from tape to a file named `dump.elf` in the file system:

```
# zgetdump /dev/ntibm0 > dump.elf
```

For general information on **zgetdump**, see "zgetdump - Copy and convert kernel dumps" on page 53 or the man page.

## Checking whether a dump is valid, and printing the dump header

To check whether a dump is valid, use the **zgetdump** command with the **-i** option.

### Procedure

1. Ensure that the volume is loaded.
2. Skip the first file on the tape (this is the dump tool itself):

   ```
   # mt -f /dev/ntibm0 fsf
   ```

3. Issue the **zgetdump** command with the **-i** option:

   ```
   # zgetdump -i  /dev/ntibm0
   ```

   The **zgetdump** command goes through the dump until it reaches the end. See also "Using zgetdump to copy a tape dump" on page 56.

# Chapter 6. Using a SCSI dump device

To use a SCSI dump device you need to install the stand-alone SCSI dump tool, perform the dump process, and copy the dump to a file in a Linux file system.

## Installing the SCSI disk dump tool

You install the SCSI dump tool with the **zipl** command.

### Before you begin

The dump partition needs enough free space (memory size + 10 MB) to hold the system memory.

**GRUB 2 usage:** You cannot use GRUB 2 to install the standalone dump tools. You must use the **zipl** command as described in this document for the dump tools.

### Example

A partition on a SCSI device is used as dump partition.

### About this task

This example assumes that `/dev/mapper/36005076303ffd40100000000000020c0` is the dump device, and that you want to dump to the first partition, `/dev/mapper/36005076303ffd40100000000000020c0-part1`. Always use multipath devices instead of single path SCSI disk device nodes, if possible.

### Procedure

1. Create a partition with **fdisk** or **parted**, using the PC-BIOS or GPT layout.
   For example:
   ```
   # fdisk /dev/mapper/36005076303ffd40100000000000020c0
   ```
2. Install the dump tool by using the **zipl** command. Specify the dump partition on the command line: For example:
   ```
   # zipl -d /dev/mapper/36005076303ffd40100000000000020c0-part1
   ```

### Results

When you perform an IPL from any of the paths of `/dev/mapper/36005076303ffd40100000000000020c0` by using boot program selector 1 or 0 (default), the memory dump is written directly to partition 1 of `/dev/mapper/36005076303ffd40100000000000020c0`. For LPAR, the boot program selector is located on the load panel, see Figure 6 on page 46 for an example. For z/VM, the boot program selector is configured with 'CP SET DUMPDEV BOOTPROG', see “Using SCSI” on page 42.

## Initiating a SCSI dump

To initiate the dump, IPL the SCSI dump tool by using the **SCSI dump** load type.

### About this task

The dump process can take several minutes depending on the device type you are using and the amount of system memory. The dump progress and any error messages are reported on the operating system messages console for LPAR, or on the 3270 console for a z/VM guest.

### Procedure

IPL the SCSI dump tool.
See Appendix A, "Examples for initiating dumps," on page 41 for more details.

### Results

The dump process copies the dump to the dump partition.

When the dump completes successfully, you can IPL Linux again.

You can now extract the dump from the dump partition into a file.

## Copying the dump from SCSI disks with zgetdump

You can copy a SCSI dump to a file system using the **zgetdump** tool.

### About this task

By default, the **zgetdump** tool takes the dump device as input and writes its contents to standard output. To write the dump to a file system, you must redirect the output to a file.

### Procedure

Assuming that the dump is on SCSI partition `/dev/mapper/36005076303ffd40100000000000020c0-part1` and you want to copy it to a file named `dump.elf`:

```
# zgetdump /dev/mapper/36005076303ffd40100000000000020c0-part1 > dump.elf
```

### What to do next

You can use **zgetdump** to display information about the dump. See "Checking whether a SCSI dump is valid and printing the dump header" on page 58 for an example. For general information about **zgetdump**, see "zgetdump - Copy and convert kernel dumps" on page 53 or the man page.

## Printing the SCSI dump header

To print the dump file header, use **zgetdump** with the **-i** option.

## Procedure

Specify the **zgetdump** command with the **-i** option:

```
# zgetdump -i /dev/mapper/36005076303ffd40100000000000020c0-part1
General dump info:
  Dump format........: elf
  Version............: 1
  UTS node name......: mylnxsys
  UTS kernel release.: 3.12.25-2-default
  UTS kernel version.: #1 SMP Mon Jul 28 12:18:48 UTC 2014 (1b84426)
  System arch........: s390x (64 bit)
  CPU count (online).: 3
  Dump memory range..: 768 MB

Memory map:
  0000000000000000 - 000000002fffffff (768 MB)
```

# Chapter 7. Creating dumps on z/VM with VMDUMP

Use VMDUMP to create dumps on z/VM systems, using the z/VM reader as the dump medium.

### About this task

Do not use **VMDUMP** to dump large z/VM guests; the dump process is very slow. Dumping 1 GB of storage can take up to 15 minutes depending on the used storage server and z/VM version.

This section describes how to create a dump with **VMDUMP**, how to transfer the dump to Linux, and how to convert the z/VM dump to a convenient format. **VMDUMP** does not need to be installed separately.

The following sections take you through the process of creating a dump with **VMDUMP**.

## Initiating a dump with VMDUMP

Start the VMDUMP process with the CP **VMDUMP** command.

### Procedure

Issue the following command from the 3270 console of the z/VM guest virtual machine:

```
#CP VMDUMP
```

### Results

z/VM CP temporarily stops the z/VM guest virtual machine and creates a dump file. The dump file is stored in the reader of the z/VM guest virtual machine. After the dump is complete, the Linux on z/VM instance continues operating.

You can use the **TO** option of the **VMDUMP** command to direct the dump to the reader of another guest virtual machine of the same z/VM system.

### Example

To write the dump to the reader of z/VM guest virtual machine `linux02` issue:

```
#CP VMDUMP TO LINUX02
```

For more information about **VMDUMP** refer to *z/VM CP Commands and Utilities Reference*, SC24-6175.

## Copying the dump to Linux

Copy the dump from the z/VM reader using the **vmur** command.

## Procedure

1. Find the spool ID of the **VMDUMP** spool file in the output of the **vmur li** command:

```
# vmur li
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME     NAME   TYPE DIST
T6360025 0463 V DMP 00020222 001 NONE 06/11 15:07:42 VMDUMP FILE T6360025
```

   In the example the required **VMDUMP** file spool ID is 463.

2. Copy the dump into your Linux file system using the **vmur receive** command. To convert the dump into a format that can be processed with the Linux dump analysis tool **crash**, convert the dump using the **--convert** option:

```
# vmur rec 463 -c myvmdump
vmdump information:
  architecture: 64 bit (big)
  storage.....: 256 MB
  date........: Thu Feb  5 08:39:48 2009
  cpus........: 1
   256 of    256 |################################################| 100%
```

## Results

The created file, named myvmdump, can now be used as input to **crash**.

# Chapter 8. Handling large dumps

Methods exist for handling memory dumps that are especially large (greater than 10 GB in size).

## Before you begin

The preferred method for handling dumps of large production systems is using kdump. With kdump, you do not need to set up a dedicated dump device with a dump tool for each individual system. Instead, you set aside storage space to receive any dumps from across your installation. When using kdump, the information in this section applies if you want to set up a backup dump method for a critical system with a large memory.

## About this task

Large dumps present a challenge as they:

- Take up a large amount of disk space
- Take a long time dumping
- Use considerable network bandwidth when being sent to a support organization.

**Note:** Sometimes you can re-create the problem on a test system with less memory, which makes the dump handling much easier. Consider this option before creating a large dump.

## Procedure

Complete these steps to prepare and process a large dump.

1. Choose a dump device. If you want to dump a system with a large memory footprint, you have to prepare a dump device that is large enough. You can use the following dump devices for large dumps:

   **Single-volume DASD**
   - 3390 model 9 (up to 45 GB)
   - 3390 model A (up to 180 GB)

   **Multivolume DASD**
   Up to 32 DASDs are possible.
   - 32 x 3390 model 9 (up to 1.4 TB)
   - 32 x 3390 model A (up to 5.7 TB)

   **z/VM emulated FBA device that represents a real SCSI device**
   FBA disks can be defined with the CP command **SET EDEVICE**. These disks can be used as single-volume DASD dump disks. The SCSI disk size depends on your storage server setup.

   **SCSI dump**
   The SCSI disk size depends on your storage server setup. For SCSI dump partitions greater than 2 TB, you must use the GPT disk layout.

   **Dump on 3592 channel-attached tape drive**
   Cartridges with up to 300 GB capacity.

Do not use **VMDUMP** for large systems, because this dump method is very slow.

2. Estimate the dump time. The dump speed depends on your environment, for example your SAN setup and your storage server. With a dump speed of approximately 100 MB per second on DASDs or SCSI disks, and a system with 50 GB memory, the dump takes approximately 8 minutes. Do a test dump on your system to determine the dump speed for it. Then you will have an indication of how long a dump will take in case of emergency.
3. Reduce the dump size. For transferring dumps in a short amount of time to a support organization, it is often useful to reduce the dump size or split the dump into several parts for easier and faster transmission. To reduce the dump, choose one of these methods:
   - "Compressing a dump using makedumpfile"
   - "Compressing a dump using gzip and split" on page 29
4. Send the dump.

## Compressing a dump using makedumpfile

Use the **makedumpfile** tool to compress s390 dumps and exclude memory pages that are not needed for analysis. Alternatively, you can use the **gzip** and **split** commands.

### About this task

Compressing the dump substantially reduces the size of dump files and the amount of time needed to transmit them from one location to another. In SUSE Linux Enterprise Server 12, the **makedumpfile** tool is part of the makedumpfile RPM that you can install, for example, with the command:

```
# zypper in makedumpfile
```

Because **makedumpfile** expects as input dump files in ELF format, you first have to transform your s390 format dump to ELF format. This is best done by mounting the dump using the **zgetdump** command. If you have read access to the dump file, you do not need root authority to mount the dump with **zgetdump**.

### Procedure

1. Mount the dump in ELF format by performing one of these steps:
   - To mount a DASD dump from the partition `/dev/dasdb1` to `/mnt`, issue:

     ```
     # zgetdump -m /dev/dasdb1 /mnt
     ```

   - To mount a SCSI dump from the partition `/dev/mapper/36005076303ffd40100000000000020c0-part1` to `/mnt`, issue:

     ```
     # zgetdump -m /dev/mapper/36005076303ffd40100000000000020c0-part1 /mnt
     ```

   After mounting the dump in ELF format with **zgetdump**, the dump is available in the file named `/mnt/dump.elf`.
2. Use the **-d** (dump level) option of **makedumpfile** to specify which pages to exclude from the dump. See the man page for **makedumpfile** for a description of the dump level and other options of **makedumpfile**.

This example compresses the dump file named `/mnt/dump.elf` (**-c** option) and excludes pages that are typically not needed to analyze a kernel problem. Excluded pages are: pages containing only zeroes, pages used to cache file contents (cache, cache private), pages belonging to user spaces processes, and free pages (maximum dump level 31):

```
# makedumpfile -c -d 31 /mnt/dump.elf dump.kdump
```

The newly created file, named `dump.kdump` should be much smaller than the original file, named `dump.elf`. Keep the original dump file until your kernel problem is resolved. This will enable you to reduce the dump level if it turns out that the pages that had been excluded are still needed for problem determination.

3. For initial problem analysis, you can also extract the kernel log with **makedumpfile**, and send it to your support organization:

```
# makedumpfile --dump-dmesg /mnt/dump.elf kernel.log
```

### What to do next

After you have used **makedumpfile**, you can unmount the dump:

```
# zgetdump -u /mnt
```

## Compressing a dump using gzip and split

Use the **gzip** and **split** commands to compress the dump and split it into parts. Alternatively, you can use the **makedumpfile** command.

### Procedure

1. Compress the dump and split it into parts of 1 GB by using the **gzip** and **split** commands.
   - For a DASD dump:

     ```
     # zgetdump /dev/dasdd1 | gzip | split -b 1G
     ```

   - For a tape dump:

     ```
     # mt -f /dev/ntibm0 rewind
     # mt -f /dev/ntibm0 fsf
     # zgetdump /dev/ntibm0 | gzip | split -b 1G
     ```

   - For a SCSI dump:

     ```
     # cat /mnt/dump.0 | gzip | split -b 1G
     ```

   The split creates several compressed files in your current directory:

   ```
   # ls
   # xaa xab xac xad xae
   ```

2. Create md5 sums of parts:

   ```
   # md5sum * > dump.md5
   ```

3. Upload the parts together with the MD5 information to the support organization.
4. The receiver (the service organization) must do the following:
   a. Verify md5 sums:

```
# cd dumpdir
# md5sum -c dump.md5
xaa: OK
xab: OK
...
```

   b. Merge parts and extract the dump:

```
# cat x* | gunzip -c > dump
```

# Chapter 9. Creating live-system dumps with zgetdump

If you require a kernel dump of a Linux instance, but no downtime is acceptable, you can create a kernel dump from a live system without disruption.

Because the Linux system continues running while the dump is written, and kernel data structures are changing during the dump process, the resulting dump contains inconsistencies. The faster the dump process completes, the fewer inconsistencies the resulting live-system dump will contain. Therefore, run the dump process with the highest acceptable priority.

You can change the scheduling priority with the nice command. For example, use **nice -n -20** to set the highest possible priority.

## Creating a kernel dump on a live system

You can create non-disruptive kernel dumps on a running Linux system with the **zgetdump** tool.

### Before you begin

- The dump directory needs enough free space (memory size + 10 MB) to hold the system memory.
- Ensure that during the dump process no memory hotplug or CPU hotplug is performed.
- If applicable, stop the cpuplugd service by issuing the command:

```
# service cpuplugd stop
```

- Load the crash kernel module by issuing the command:

```
# modprobe crash
```

### Procedure

1. Optional: Use the **-i** option to print information for the currently running Linux image:

```
# zgetdump -i /dev/crash
General dump info:
  Dump format........: devmem
  Dump method........: live
  UTS node name......: mylnxsys
  UTS kernel release.: 3.12.25-2-default
  UTS kernel version.: #1 SMP Mon Jul 28 12:18:48 UTC 2014 (1b84426)
  System arch........: s390x (64 bit)
  Dump memory range..: 896 MB

Memory map:
 0000000000000000 - 0000000037ffffff (896 MB)
```

2. Create a dump from a live system by specifying `/dev/crash` as input dump and redirecting the output to a dump file. Run the dump process with a high priority.

```
# nice -n -20 zgetdump /dev/crash > dump.elf
```

Optionally, you can also specify a target dump format with the **-f** option:

```
# zgetdump /dev/crash -f elf > dump.elf
```

3. Optional: Print information for the live-system dump. Use the **-i** option to print information for live-system dumps that are generated by **zgetdump**:

```
# zgetdump -i dump.elf
General dump info:
  Dump format........: elf
  Version............: 1
  Dump method........: live
  UTS node name......: mylnxsys
  UTS kernel release.: 3.12.25-2-default
  UTS kernel version.: #1 SMP Mon Jul 28 12:18:48 UTC 2014 (1b84426)
  System arch........: s390x (64 bit)
  Dump memory range..: 896 MB

Memory map:
 0000000000000000 - 0000000037ffffff (896 MB)
```

The value "live" in the **Dump method** field indicates that this is a dump from a live system.

### Example

```
# nice -n -20 zgetdump /dev/crash -f elf > dump.elf
Format Info:
  Source: devmem
  Target: elf
Copying dump:
  00000000 / 00000896 MB
  00000149 / 00000896 MB
  ...
  00000747 / 00000896 MB
  00000896 / 00000896 MB
Success: Dump has been copied
```

### What to do next

After you create a dump from a live system, you can work with crash, see "Opening a live-system dump with the crash tool."

After the live dump has been copied to a file system, you can compress it with **makedumpfile**. Note that the dump level must not be greater than 1 because of the dump inconsistencies.

For example:

```
# makedumpfile dump.elf -c -d 1 dump.kdump
```

## Opening a live-system dump with the crash tool

Inconsistencies in a kernel dump from a live system can cause some crash commands to fail.

## Before you begin

You might need to install debuginfo to see information about a dump when you use the **crash** command. Use the **zypper** command to install debuginfo, for example:

```
# zypper in kernel-default-debuginfo
```

## Procedure

- Use the **crash** command to find information about whether a dump is from a live system. This information is displayed in the startup messages, or when you use the sys command:

```
# crash dump.elf /boot/vmlinux-3.12.<xx>-<y>-default.gz
...
    KERNEL: /boot/vmlinux-3.12.<xx>-<y>-default.gz
 DEBUGINFO: /usr/lib/debug/boot/vmlinux-3.12.<xx>-<y>-default.debug
  DUMPFILE: dump.elf [LIVE DUMP]
      CPUS: 6
...
crash> sys | grep DUMPFILE
...
 DUMPFILE: dump.elf  [LIVE DUMP]
...
```

  The tag [LIVE DUMP] informs you that the dump contains inconsistencies.

- Detect whether a dump is from a live system by using the **help -p** command:

```
# crash> help -p | grep flags2
flags2: 40 (LIVE_DUMP)
```

- Use the `--minimal` option if the crash tool fails to start because of inconsistent data structures in the kernel dump. With this option, crash tolerates a degree of inconsistency. However, only a subset of crash commands is then available:

```
# crash --minimal dump.elf /boot/vmlinux-3.12.<xx>-<y>-default.gz
...
NOTE: minimal mode commands: log, dis, rd, sym, eval, set, extend and exit
```

# Chapter 10. Sharing dump devices

For reasons of economy, you might want to share dump devices rather than setting up a dedicated dump device for each Linux instance.

This section applies to sharing dump devices that are set up with stand-alone dump tools.

With kdump, you can transmit the dump through a network and use existing mechanisms to prevent conflicts when concurrently writing multiple dumps to a shared persistent storage space. VMDUMP uses z/VM resources to hold the initial dump and the integrity of each dump is handled by the z/VM system.

## Serialization and device locking

To share devices, some type of serialization is needed to prevent two systems from dumping at the same time, and thus corrupting the dumps.

Either the involved operators must prevent concurrent dumps manually, or, in some cases, available system mechanisms can be used to prevent concurrent dumping. It is possible in many cases to use a pool of devices for sharing. For the sake of simplicity, most of the following examples use only one dump device.

Possible serialization mechanisms:

**External**
: Operators must find an external way to ensure serialization manually.

**Link** Exclusive write for minidisk is used as a locking mechanism (see "Sharing DASD devices under z/VM" on page 36).

**Attach** Attach and detach is used as locking mechanism (see "Using attach and detach as locking mechanism under z/VM" on page 37).

**vmcmd**
: Use the **vmcmd** panic action (see "DASD (vmcmd panic action)" on page 37).

Alternatively, use no serialization and take the risk that dumps are overwritten, see "DASD (dump or dump_reipl panic action)" on page 37).

Table 3 shows the serialization methods available for different system configurations.

*Table 3. Serialization of dump devices overview*

| | DASD | | SCSI | |
|---|---|---|---|---|
| | z/VM | LPAR | z/VM | LPAR |
| **Manual dump** | link, attach, external | external | attach, external | external |
| **Automatic dump** | overwrite, vmcmd | overwrite | N/A | N/A |

# Sharing devices when dumping manually

In the following sections, it is assumed that you start the dump process manually, without using automatic dump on panic.

## Sharing DASD devices on LPARs

Configure your IOCDS so that all LPARs that want to share the dump device can access the DASD device. There is no system mechanism available for serialization. Exclusive access must be ensured manually by the involved system operators.

## Sharing DASD devices under z/VM

Under z/VM, DASD devices can be shared if they are defined as shareable minidisks for a **NOLOG** user.

### About this task

Exclusive access can be guaranteed by the **link** CP command using the exclusive write mode option. With this mode only one DASD can be linked to one z/VM guest virtual machine at the same time. Therefore, the dump device is excluded from other systems until it is detached.

### Procedure

To create a dump after a system crash, perform these steps:

1. To link the dump device, issue a command of the form:

   ```
   #cp link <disk owner> <vdev1> <vdev2> EW
   ```

   where
   - <*disk owner*> is the user ID in the system directory whose entry is to be searched for device <*vdev1*>.
   - <*vdev1*> is the specified user's virtual device number.
   - <*vdev2*> is the virtual device number that is to be assigned to the device for your virtual machine configuration.
2. Create the dump with device <*vdev2*>
3. Reboot your Linux system.
4. On your Linux system, set dump device <*vdev2*> online.
5. On your Linux system, copy the dump using the **zgetdump**.
6. On your Linux system, set dump device <*vdev2*> offline.
7. Detach the dump device:

   ```
   #cp detach <vdev2>
   ```

### Results

The dump DASD is free again and can be used by other systems.

## Sharing SCSI devices

You can share SCSI devices for dumping from multiple Linux systems.

You can share FCP-attached SCSI disks for dump. The disks must be accessible through your SAN on all Linux systems that want to use the dump device. The involved operators must ensure manually that two dumps are not taking place at the same time. If multiple Linux systems write to the shared dump device at the same time, you might corrupt the dump.

### Using attach and detach as locking mechanism under z/VM

For your shared dump devices, you can use attach and detach as a locking mechanism.

At any time, only one z/VM guest virtual machine can attach a device. A Linux instance that runs as a guest of a class B z/VM guest virtual machine can lock a shared dump device by attaching it. If you use one single FCP adapter for dumps on all systems, attach and detach can be also be used as locking mechanism for SCSI dump.

## Sharing devices when dumping automatically

You can configure a memory dump to be created automatically if a kernel panic occurs.

#### About this task

The automatic dump on panic can be configured in `/etc/sysconfig/dumpconf` (see "dumpconf - Configure panic or PSW restart action" on page 58).

### DASD (dump or dump_reipl panic action)

You can share DASD for automatic dump on panic, but no serialization mechanism is available.

#### About this task

Because no serialization mechanism is available, two systems dumping at the same time might corrupt the dumps. Such a dump setup is a trade-off between reliability and resource expenses. Carefully weigh the likelihood of two concurrent system crashes against the business impact of loosing a dump before using this setup.

#### Procedure

To share DASDs under z/VM, you must use minidisks that are linked in access mode multiple-write (MW) to all systems where you want to configure dump on panic.

### DASD (vmcmd panic action)

You can specify up to eight CP commands in a configuration file. These commands run if a kernel panic occurs.

#### Before you begin

Define minidisks 4e1 and 4e2 with disk owner user **SHARDISK** and prepare them as dump DASDs.

### About this task

With z/VM, you can use the panic action **vmcmd** in `/etc/sysconfig/dumpconf` to specify up to eight commands that are run in case of a kernel panic. You can use this mechanism to implement locking through the exclusive link or attach method.

In this example, assume that we want to link either 4e1 or 4e2 as device number 5000 and then create the dump using device 5000. The first free DASD is linked. If both devices are already linked to other z/VM guest virtual machines, the system stops without creating a dump.

### Procedure

The corresponding configuration for `/etc/sysconfig/dumpconf` looks like this:

```
ON_PANIC=vmcmd
VMCMD_1="LINK SHARDISK 4E1 5000 EW"
VMCMD_2="LINK SHARDISK 4E2 5000 EW"
VMCMD_3="STORE STATUS"
VMCMD_4="IPL 5000"
```

### Results

After the dump process has finished, you must perform an IPL on the Linux system manually, copy the dump, and detach disk 5000.

Compared to "DASD (dump or dump_reipl panic action)" on page 37, this option has the advantage that you cannot get corrupted dumps, and you can use more than one dump device. It has the disadvantage that automatic re-IPL is not possible.

## FCP-attached SCSI devices

Device sharing for automatic dumps is risky when using FCP-attached devices.

If multiple Linux systems write to the shared dump device at the same time, you might corrupt the dump partition.

# Sharing dump devices between different versions of Linux

Do not share dump devices between Linux installations with different major releases.

For example, do not share dump devices between SUSE Linux Enterprise Server 11 and SUSE Linux Enterprise Server 12.

You can share dump devices between Linux installations with different service levels. Prepare the dump device with the **zipl** tool from the lowest service level. For example, if you have systems with SUSE Linux Enterprise Server 12 and SP1, prepare your dump device with the **zipl** tool from the SUSE Linux Enterprise Server 12 system. Newer tools such as **zgetdump** or dump analysis tools such as **crash** can always process dumps that were created with older **zipl** versions. The other way around might work, but it is not guaranteed to work.

# Sharing dump resources with VMDUMP

With z/VM, **VMDUMP** can be run concurrently on different guest virtual machines.

The dump speed is slow, and therefore is best for very small systems. The shared resource here is the z/VM spool space. You must ensure that it has enough space to hold multiple dumps created by **VMDUMP**.

# Appendix A. Examples for initiating dumps

You can initiate dumps from different control points, such as the z/VM 3270 console or the HMC.

## z/VM

You can initiate dumps from z/VM using kdump, a DASD device, tape, a SCSI disk, or **VMDUMP**.

### About this task

The following examples assume the 64-bit mode. Corresponding 31-bit examples would have a different PSW but be the same otherwise.

## Using kdump

With kdump you do not need a dump device to initiate the dump.

### Before you begin

Your Linux instance must have been set up for kdump as described in "Setting up kdump" on page 7.

### Procedure

Issue the **system restart** z/VM CP command, for example from a 3270 terminal emulation for the Linux instance to be dumped:

```
#cp system restart
```

Boot messages for the kdump kernel indicate that the dump process has started.

## Using DASD

You can initiate a dump from a DASD device.

### Example

If 193 is the dump device:

```
#cp cpu all stop
#cp store status
#cp i 193
```

On z/VM, a three-processor machine in this example, you will see messages about the disabled wait:

```
01: The virtual machine is placed in CP mode due to a SIGP stop from CPU 00.
02: The virtual machine is placed in CP mode due to a SIGP stop from CPU 00.
"CP entered; disabled wait PSW 00020000 80000000 00000000 00000000"
```

You can now IPL your Linux instance and resume operations.

## Using tape

You can initiate a dump under z/VM using tape.

### Example

If 193 is the tape device:

```
#cp rewind 193
#cp cpu all stop
#cp store status
#cp i 193
```

On z/VM, a three-processor machine in this example, you will see messages about the disabled wait:

```
01: The virtual machine is placed in CP mode due to a SIGP stop from CPU 00.
02: The virtual machine is placed in CP mode due to a SIGP stop from CPU 00.
"CP entered; disabled wait PSW 00020000 80000000 00000000 00000000"
```

You can now IPL your Linux instance and resume operations.

## Using SCSI

You can initiate a dump using a SCSI disk.

### About this task

Assume your SCSI dump disk has the following parameters:

- WWPN: 4712076300ce93a7
- LUN: 4712000000000000
- FCP adapter device number: 4711

### Results

Messages on the operating system console will show when the dump process is finished.

### Example

```
#cp set dumpdev portname 47120763 00ce93a7 lun 47120000 00000000
#cp ipl 4711 dump
```

### What to do next

You can now IPL your Linux instance and resume operations.

## Dumping NSSs

You can create dumps from NSSs with VMDUMP. As of z/VM 6.3 you can also use stand-alone dump with the `nssdata` IPL option to dump NSSs.

### Procedure

- To initiate a DASD or tape dump for NSS, issue this command from the console of your z/VM guest virtual machine:

```
#cp i <devno> nssdata
```

where <*devno*> is the device number for the dump device.

- To initiate a SCSI dump for NSS, issue this command from the console of your z/VM guest virtual machine:

```
#cp i <devno> dump nssdata
```

where <*devno*> is the FCP adapter device number.

## Using VMDUMP

You can initiate a dump under z/VM by using VMDUMP.

### Procedure

To initialize a dump with **VMDUMP**, issue this command from the console of your z/VM guest virtual machine:

```
#cp vmdump
```

### Results

Dumping does not force you to perform an IPL. If the Linux instance ran as required before dumping, it continues running after the dump is completed.

# HMC or SE

You can initiate a dump process on an LPAR from an HMC (Hardware Management Console) or SE (Support Element).

### About this task

The following description refers to an HMC, but the steps also apply to an SE. The steps are similar for DASD, tape, and SCSI. Differences are noted where applicable. You cannot initiate a dump with **VMDUMP** from the HMC or SE.

### Procedure

1. In the left navigation pane of the HMC, expand Systems Management and Servers and select the mainframe system you want to work with. A table of LPARs is displayed in the upper content area on the right.
2. Select the LPAR for which you want to initiate the dump.
3. In the **Tasks** area, expand **Recovery**. Proceed according to your dump device:
   - If you are using kdump, click **PSW restart**. This initiates the dump process. Skip the remaining procedure, no further steps are required.
   - If you are dumping to DASD or tape, click **Stop all** in the **Recovery** list to stop all CPUs. Confirm when you are prompted to do so.
   - If you are dumping to a SCSI disk, skip this step and proceed with step 4 on page 44

   Figure 4 on page 44 shows an example of an HMC with a selected mainframe system and LPAR. The **Load**, **PSW restart**, and **Stop all** tasks can be seen in the

expanded **Recovery** list.

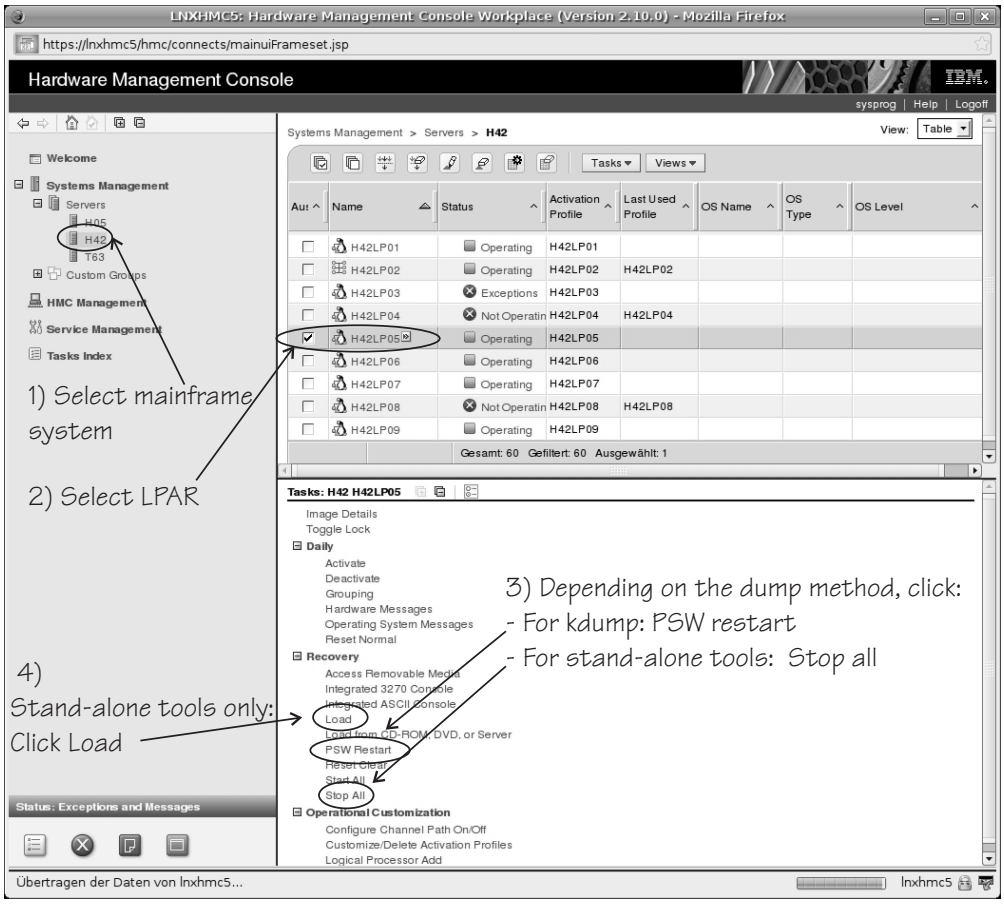



*Figure 4. HMC with the* **Load**, **PSW restart**, *and* **Stop all** *tasks*

4. Click **Load** in the **Recovery** list to display the Load panel.

   **For a dump to DASD or tape:**

   a. Select **Load type** "Normal".
   b. Select the **Store status** check box.
   c. Type the device number of the dump device into the **Load address** field.

   Figure 5 on page 45 shows a Load panel with all entries and selections required to start the dump process for a DASD or tape dump device.

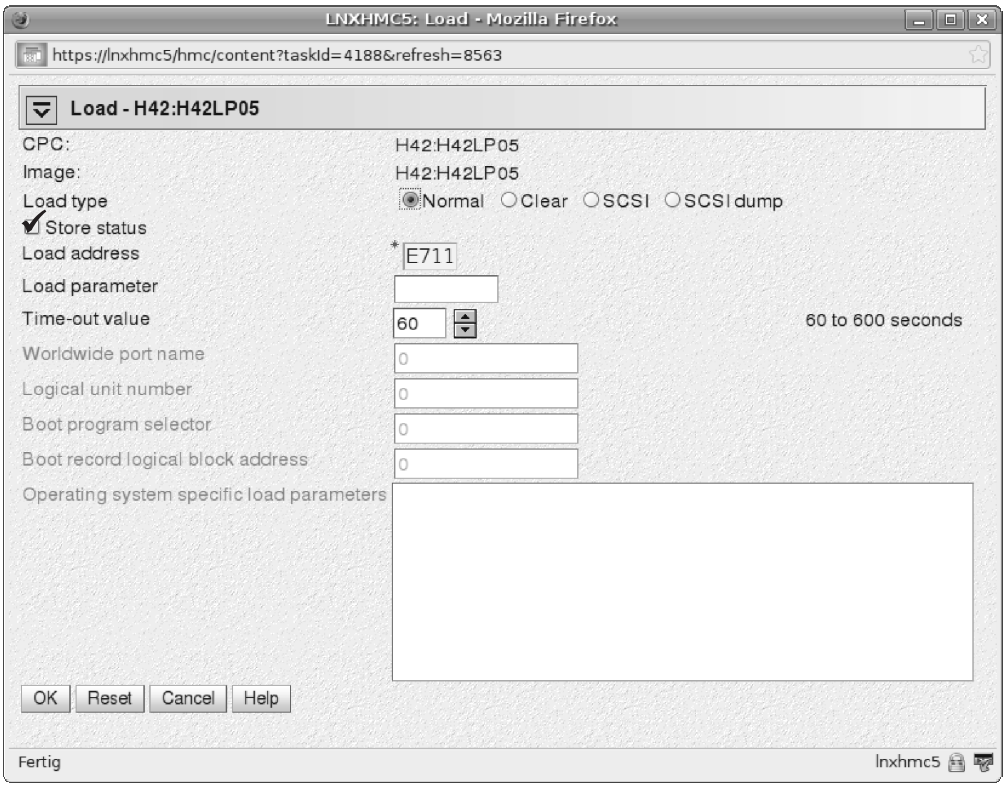



*Figure 5. Load panel for dumping to DASD or tape*

**For a dump to SCSI disk:**

a. Select **Load type** "SCSI dump".

b. Type the device number of the FCP adapter for the SCSI disk into the **Load address** field.

c. Type the World Wide Port name of the SCSI disk into the **World wide port name** field.

d. Type the Logical Unit Number of the SCSI disk into the **Logical unit number** field.

e. Type `0` in the **Boot program selector** field.

f. Accept the defaults for the remaining fields.

Figure 6 on page 46 shows a Load panel with all entries and selections required to start the SCSI dump process.

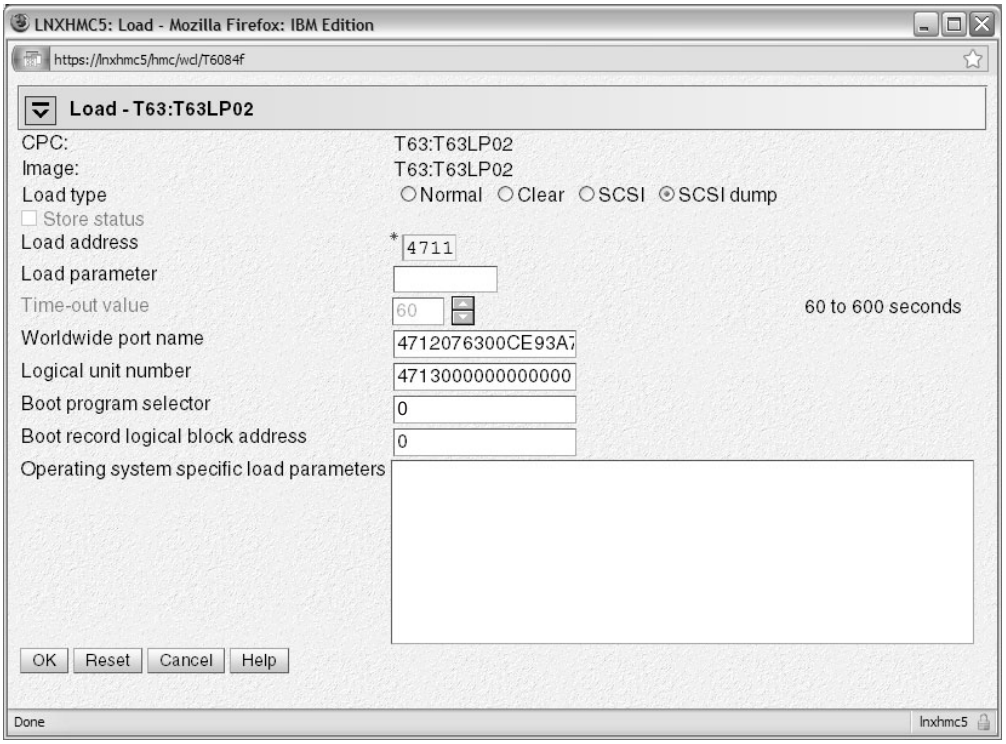



*Figure 6. Load panel with enabled SCSI feature for dumping to SCSI disk*

5. Click **OK** to start the dump process.
6. Wait until the dump process completes. Click the **Operating System Messages** icon for progress and error information.

### Results

When the dump has completed successfully for a stand-alone dump tool, you can IPL Linux again. When using kdump, the reboot is normally done automatically.

## Triggering a dump remotely

You can trigger HMC or SE activities remotely by using the **snipl** command.

### Before you begin

As of **snipl** version 2.1.9, the **snipl** command can be used for dump handling. The required setup for **snipl** usage and further details are described in *Device Drivers, Features, and Commands on SUSE Linux Enterprise Server 12*, SC34-2745. You can dump to a DASD device or a SCSI disk.

### About this task

For example, assume that you have a **snipl** configuration file `/etc/snipl.conf` containing the following specifications:

```
server=myse9.example.com
image=LPARLNX1
image=LPARLNX2

server=myse5.example.com
image=LPRLNX05
```

Further assume that you have prepared a dump DASD (in this example with device number 5199) with the **zipl** tool.

### Procedure

Use the following **snipl** commands to write a memory dump of LPARLNX1 to the prepared DASD:

1. Stop the CPUs:

```
# snipl LPARLNX1 --stop
Server myse9.example.com from config file /etc/snipl.conf is used
processing......
LPARLNX1: acknowledged.
```

2. IPL the dump tool on DASD 5199, prepared with the dump tool:

```
# snipl LPARLNX1 --load -A 5199 --storestatus
Server myse9.example.com from config file /etc/snipl.conf is used
processing......
LPARLNX1: acknowledged.
```

3. Monitor the dump progress:

```
# snipl LPARLNX1 --dialog
LPARLNX1: acknowledged.
Starting operating system messages interaction for
partition  LPARLNX1 (Ctrl-D to abort):
00000128 / 00001024 MB
...
00000896 / 00001024 MB
00001024 / 00001024 MB
Dump successful
```

### Example

The corresponding **snipl** command to write a memory dump to a SCSI disk with WWPN 500507630303c562, LUN 4010404900000000, and FCP adapter 5000 is:

```
# snipl LPRLNX05 --scsidump -A 5000 --wwpn_scsiload 500507630303c562 --lun_scsiload 4010404900000000
Server myse5.example.com from config file /etc/snipl.conf is used
processing...
LPRLNX05: acknowledged.
```

## Testing automatic dump-on-panic

Cause a kernel panic to confirm that your dump configuration is set up to automatically create a dump if a kernel panic occurs.

### Before you begin

You need a Linux instance with active magic sysrequest functions.

## Procedure

Crash the kernel with a forced kernel panic.

| If your method for triggering the magic sysrequest function is: | Enter: |
|---|---|
| **A command on the 3270 terminal or line-mode terminal on the HMC** | `^-c` |
| **A command on the hvc0 terminal device** | Ctrl+o `c` |
| **Writing to procfs** | `echo c > /proc/sysrq-trigger` |

**Note:** Ctrl+o means pressing `o` while holding down the control key.
For more details about the magic sysrequest functions, see the documentation in the Linux source tree at `/usr/src/linux/Documentation/sysrq.txt` (requires installation of the kernel-source package).

## Results

The production system crashes. Provided that the setup for automatic dump on panic is correct, a dump is now written.

# Appendix B. Obtaining a dump with limited size

The `mem` kernel parameter can make Linux use less memory than is available to it. A dump of such a Linux system does not need to include the unused memory. You can use the **zipl** command with the **size** option to limit the amount of memory that is dumped.

## About this task

The **size** option is available for all **zipl** based dumps except SCSI: DASD and tape in command-line mode or in configuration-file mode. The **size** option is appended to the dump device specification with a comma as separator.

The value is a decimal number that can optionally be suffixed with K for kilobytes, M for megabytes, or G for gigabytes. Values that are specified in byte or kilobyte are rounded to the next megabyte boundary.

Be sure not to make the dump size smaller than the amount of memory that is actually used by the system to be dumped. Limiting the dump size to less than the amount of used memory results in an incomplete dump.

## Example

The following command, entered in a Linux shell, prepares a DASD dump device for a dump that is limited to 100 MB:

```
# zipl -d /dev/dasdc1,100M
Setting dump size limit to 100MB
Dump target: partition '/dev/dasdc1' with a size of 7043 MB.
Warning: All information on partition '/dev/dasdc1' will be lost!
Do you want to continue creating a dump partition (y/n)?y
Done.
```

After IPL of the dump device, you can see output from the dump tool on the z/VM console as the dump is created:

```
00:
00: CP I 63AD
01: HCPGSP2630I The virtual machine is placed in CP mode due to a SIGP stop
    and store status from CPU 00.
02: HCPGSP2630I The virtual machine is placed in CP mode due to a SIGP stop
    and store status from CPU 00.
00: zIPL v1.15.0-0.132.4 dump tool (64 bit)
00: Dumping 64 bit OS
00:
00: 00000012 / 00000100 MB
00: 00000025 / 00000100 MB
00: 00000037 / 00000100 MB
00: 00000050 / 00000100 MB
00: 00000062 / 00000100 MB
00: 00000075 / 00000100 MB
00: 00000087 / 00000100 MB
00: 00000100 / 00000100 MB
00: Dump successful
00: HCPGIR450W CP entered; disabled wait PSW 00020000 80000000 00000000 00000000
```

# Appendix C. Command summary

The descriptions of the commands contain only the options and parameters that are relevant to dumping on SUSE Linux Enterprise Server. For a full description see the man pages.



## zipl - Prepare devices for stand-alone dump

Use **zipl** to prepare a dump device with a stand-alone dump tool.

Command options that do not apply to SUSE Linux Enterprise Server 12 have been omitted. For details, see the man page.

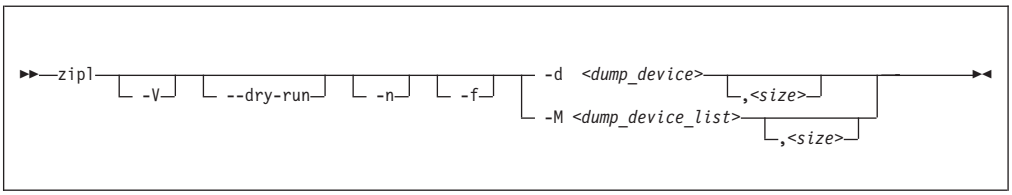
**zipl** supports the following dump devices:

- Enhanced Count Key Data (ECKD) DASDs with fixed block Linux disk layout (ldl)
- ECKD DASDs with z/OS-compliant compatible disk layout (cdl)
- Fixed Block Access (FBA) DASDs
- Magnetic tape subsystems compatible with IBM3480, IBM3490, IBM3590, or IBM3592
- SCSI with PC-BIOS or GPT disk layout

For multivolume dumps, only ECKD DASDs with compatible disk layout are supported.

### zipl syntax

**Note:** You can specify zipl parameters in a configuration file, but the preferred way of using zipl is the command line. For details about the configuration file, see the man page.




### Parameters

**-d** *<dump_device>* **or --dumpto=***<dump_device>*
is the device node of the DASD or SCSI partition, or tape device to be prepared as a dump device. **zipl** deletes all data on the partition or tape and installs the boot loader code there.

**Note:**

- If the dump device is an ECKD disk with fixed-block layout (ldl), a dump overwrites the dump utility. You must reinstall the dump utility before you can use the device for another dump.
- If the dump device is a tape, SCSI disk, FBA disk, or ECKD disk with the compatible disk layout (cdl), you do not need to reinstall the dump utility after every dump.

**-M** *<dump_device_list>*
contains the device nodes of the dump partitions, separated by one or more line feed characters. **zipl** writes a dump signature to each involved partition and installs the stand-alone multi-volume dump tool on each involved volume. Duplicate partitions are not allowed. A maximum of 32 partitions can be listed. The volumes must be formatted with cdl and use block size 4096.

*<size>*
(Optional) The amount of memory to be dumped. The value is a decimal number that can optionally be suffixed with K for kilobytes, M for megabytes, or G for gigabytes. The value is rounded to the next megabyte boundary.

If you limit the dump size below the amount of memory that is used by the system to be dumped, the resulting dump is incomplete. If no limit is provided, all of the available physical memory is dumped.

**Note:** For SCSI dump devices, the "size" option is not available.

**-f or --force**
ensures that no signature checking takes place when dumping. Any data on all involved partitions is overwritten without warning.

**-n or --noninteractive**
suppresses confirmation prompts that require operator responses to allow unattended processing (for example, for processing DASD or tape dump configuration sections). This option is available on the command line only.

**-V or --verbose**
provides more detailed command output.

**--dry-run**
simulates a **zipl** command. Use this option to test a configuration without overwriting data on your device.

During simulation, **zipl** performs all command processing and issues error messages where appropriate. Data is temporarily written to the target directory and is cleared up when the command simulation is completed.

**-v or --version**
displays version information.

**-h or --help**
displays help information.

## Preparing a DASD dump device

The following command prepares a DASD partition `/dev/dasdc1` as a dump device and suppresses confirmation prompts that require an operator response:

```
# zipl -d /dev/dasdc1 -n
```

### Preparing a SCSI dump device

The following command prepares a SCSI partition `/dev/mapper/36005076303ffd40100000000000020c0-part1` as a dump device:

```
# zipl -d /dev/mapper/36005076303ffd40100000000000020c0-part1
```

### Preparing a multi-volume dump on ECKD DASD

The following command prepares two DASD partitions `/dev/dasdc1`, `/dev/dasdd1` for a multi-volume dump and suppresses confirmation prompts that require an operator response:

```
# zipl -M mvdump.conf -n
```

where the `mvdump.conf` file contains the two partitions, separated by line breaks:

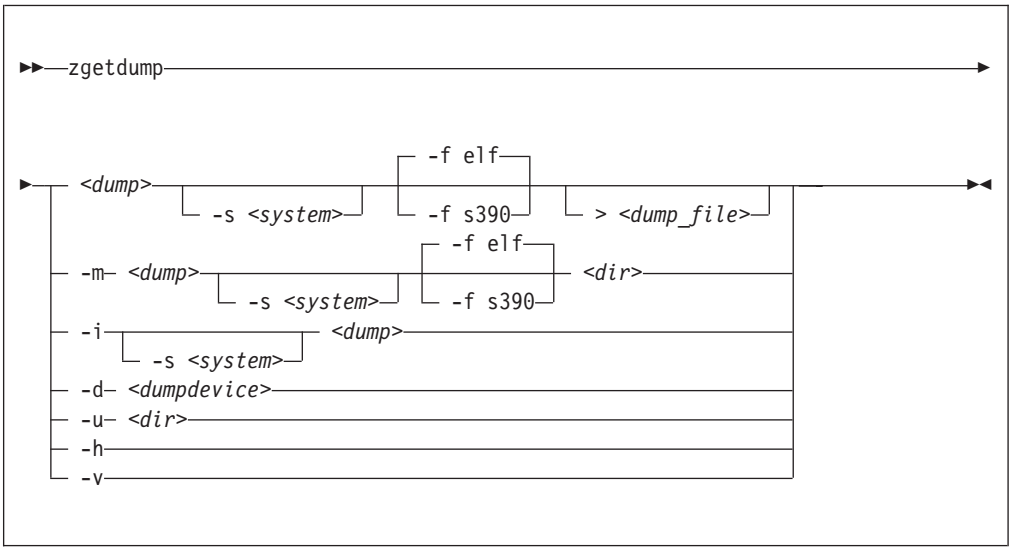
```
/dev/dasdc1
/dev/dasdd1
```

## zgetdump - Copy and convert kernel dumps

The **zgetdump** tool copies a source dump into a target dump with a configurable dump format. The source dump can be located either on a dump device or on a file system. The source dump content is written to standard output, unless you redirect it to a specific file. You can also mount the dump content, print dump information, check whether a DASD device contains a valid dump tool, or create a non-disruptive kernel dump on a live system.

**Before you begin:** Mounting is implemented with fuse (file system in user space). Therefore, the fuse kernel module must be loaded before you can use the **-m** option. You do not need root authority to work with a dump using **zgetdump**.

### zgetdump syntax



## Parameters

*<dump>*
is the file, DASD device or partition, tape device, or live system device node where the source dump is located:

- Regular dump file (for example `/testdir/dump.0`)
- DASD partition device node (for example `/dev/dasdc1`)
- DASD device node for multivolume dump (for example `/dev/dasdc`)
- Tape device node (for example `/dev/ntibm0`)
- Device node for live system (`/dev/mem`)

**Note:** For a DASD multivolume dump, it is sufficient to specify only one of the multivolume DASDs as *<dump>*.

*<dump_file>*
is the file to which the output is redirected. The default is standard output.

*<dumpdevice>*
specifies the dump device for the `-d` option. The device node of the DASD device, for example `/dev/dasdb`.

**-s** *<system>* **or --select** *<system>*
for dumps that capture two systems, selects the system of interest. This option is mandatory when you access the dump of a crashed kdump instance, but returns an error if applied to a regular dump.

A dump can contain data for a crashed production system and for a crashed kdump system. A dump like this is created if a stand-alone dump tool is used to create a dump for a kdump instance that crashed while creating a dump for a previously crashed production system. *<system>* can be:

**prod**
to select the data for the crashed production system.

**kdump**
to select the data for the kdump instance that crashed while creating a dump for the previously crashed production system.

**-m** *<dump>* *<dir>* **or --mount** *<dump>* *<dir>*
mounts the source dump *<dump>* to mount point *<dir>* and generates a virtual target dump file instead of writing the content to standard output. The virtual dump file is named *<dump>*`.`*<FMT>*, where *<FMT>* is the name of the specified dump format (see the **--fmt** option).

**-u** *<dir>* **or --umount** *<dir>*
unmounts the dump that is mounted at mount point `<dir>`. You can specify the dump itself instead of the directory, for example `/dev/dasdd1`. This option is a wrapper for **fusermount -u**.

**-i** *<dump>* **or --info** *<dump>*
displays the dump header information from the dump and performs a validity check.

**-d** *<dumpdevice>* **or --device** *<dumpdevice>*
checks whether the specified ECKD or FBA device contains a valid dump tool and prints information about it.

**-f** *<format>* **or --fmt** *<format>*
uses the specified target dump format *<format>* when writing or mounting the dump. The following target dump formats are supported:

**elf** Executable and Linking Format core dump (default)

**s390** S/390® dump

**-h or --help**
displays the help information for the command.

**-v or --version**
displays the version information for the command.

## Using zgetdump to copy a dump

Assuming that the dump is on DASD partition /dev/dasdb1 and that you want to copy it to a file named dump.elf:

```
# zgetdump /dev/dasdb1 > dump.elf
```

## Using zgetdump to transfer a dump with ssh

Assuming that the dump is on DASD partition /dev/dasdd1 and that you want to transfer it to a file on another system with ssh:

```
# zgetdump /dev/dasdd1  | ssh user@host "cat > dump.elf"
```

## Using zgetdump to transfer a dump with FTP

Assuming that you want to use FTP to transfer a dump to a file, dump.elf, on another system:

1. Establish an FTP session with the target host and log in.
2. To transfer a file in binary mode, enter the FTP **binary** command:

   ```
   ftp> binary
   ```

3. To send the dump file to the FTP host issue:

   ```
   ftp> put |"zgetdump /dev/dasdb1" dump.elf
   ```

## Using zgetdump to copy a multi-volume dump

Assuming that the dump is on DASD devices /dev/dasdc and /dev/dasdd spread along partitions /dev/dasdc1 and /dev/dasdd1, and that you want to copy it to a file named dump.elf:

```
# zgetdump /dev/dasdc > dump.elf
```

For an example of the output from this command, see Chapter 4, “Using DASD devices for multi-volume dump,” on page 13.

## Using zgetdump to copy a tape dump

Assuming that the tape device is /dev/ntibm0:

```
# zgetdump /dev/ntimb0 > dump.elf
Format Info:
  Source: s390tape
  Target: elf

Copying dump:
  00000000 / 00001024 MB
  00000171 / 00001024 MB
  00000341 / 00001024 MB
  00000512 / 00001024 MB
  00000683 / 00001024 MB
  00000853 / 00001024 MB
  00001024 / 00001024 MB

Success: Dump has been copied
```

## Using zgetdump to create a dump from a live system

To store an ELF-format dump from a live system in a file called dump.elf issue:

```
# nice -n -20 zgetdump /dev/crash > dump.elf
```

For an example of the output from this command, see "Creating a kernel dump on a live system" on page 31.

## Checking whether a tape dump is valid, and printing the dump header

Assuming that the tape device is /dev/ntibm0:

```
# zgetdump -i /dev/ntibm0
Checking tape, this can take a while...
General dump info:
  Dump format........: s390tape
  Version............: 5
  Dump created.......: Mon, 28 Jul 2014 17:26:46 +0200
  Dump ended.........: Mon, 28 Jul 2014 17:27:58 +0200
  Dump CPU ID........: ff00012320948000
  Build arch.........: s390x (64 bit)
  UTS node name......: mylnxsys
  UTS kernel release.: 3.12.25-2-default
  UTS kernel version.: #1 SMP Mon Jul 28 12:18:48 UTC 2014
  System arch........: s390x (64 bit)
  CPU count (online).: 2
  CPU count (real)...: 2
  Dump memory range..: 1024 MB
  Real memory range..: 1024 MB

Memory map:
  0000000000000000 - 000000003fffffff (1024 MB)
```

## Checking whether a DASD dump is valid and printing the dump header

Assuming that the dump is on a partition, `part1`, of a DASD device `/dev/dasdb1`:

```
# zgetdump -i /dev/dasdb1
General dump info:
  Dump format........: s390
  Version............: 5
  Dump created.......: Wed, 13 Aug 2014 11:14:33 +0100
  Dump ended.........: Wed, 13 Aug 2014 11:14:46 +0100
  Dump CPU ID........: ff00012320978000
  UTS node name......: mylnxsys
  UTS kernel release.: 3.12.25-2-default
  UTS kernel version.: #1 SMP Mon Jul 28 12:18:48 UTC 2014
  Build arch.........: s390x (64 bit)
  System arch........: s390x (64 bit)
  CPU count (online).: 3
  CPU count (real)...: 3
  Dump memory range..: 1024 MB
  Real memory range..: 1024 MB

Memory map:
  0000000000000000 - 000000003fffffff (1024 MB)
```

## Checking whether a device contains a valid dump record

Checking DASD device `/dev/dasda`, which is a valid dump device:

```
# zgetdump -d /dev/dasdb
Dump device info:
  Dump tool.........: Single-volume DASD dump tool
  Version...........: 2
  Architecture......: s390x (64 bit)
  DASD type.........: ECKD
  Dump size limit...: none
```

Checking DASD device `/dev/dasdc`, which is not a valid dump device:

```
# zgetdump -d /dev/dasdc
zgetdump: No dump tool found on "/dev/dasdc"
```

## Using the mount option

Mounting is useful for multivolume DASD dumps. After a multivolume dump has been mounted, it is shown as a single dump file that can be accessed directly with dump processing tools such as **crash**.

The following example mounts a multivolume source DASD dump as an ELF dump, processes it with **crash**, and unmounts it with **zgetdump**:

```
# zgetdump -m /dev/dasdx /dumps
# crash  /dumps/dump.elf /boot/vmlinux-3.12.<xx>-<y>-default.gz
# zgetdump -u /dumps
```

Mounting can also be useful when you want to process the dump with a tool that cannot read the original dump format. Use the **--fmt** option to mount the dump with a format other then the default format.

### Selecting data from a dump that includes a crashed kdump

The following example mounts dump data for a crashed production system from a DASD backup dump for a failed kdump (see “Failure recovery and backup tools” on page 7 for details).

```
# zgetdump -s prod -m /dev/dasdb1 /mnt
```

### Checking whether a SCSI dump is valid and printing the dump header

Assuming that the dump is on the first partition of a SCSI disk, for example `/dev/mapper/36005076303ffd40100000000000020c0-part1`:

```
# zgetdump -i /dev/mapper/36005076303ffd40100000000000020c0-part1
General dump info:
  Dump format........: elf
  Version............: 1
  UTS node name......: mylnxsys
  UTS kernel release.: 3.12.25-2-default
  UTS kernel version.: #1 SMP Mon Jul 28 12:18:48 UTC 2014
  System arch........: s390x (64 bit)
  CPU count (online).: 3
  Dump memory range..: 1024 MB

Memory map:
  0000000000000000 - 000000003fffffff (1024 MB)
```

## dumpconf - Configure panic or PSW restart action

The **dumpconf** service configures the action to be taken if a kernel panic or PSW restart occurs.

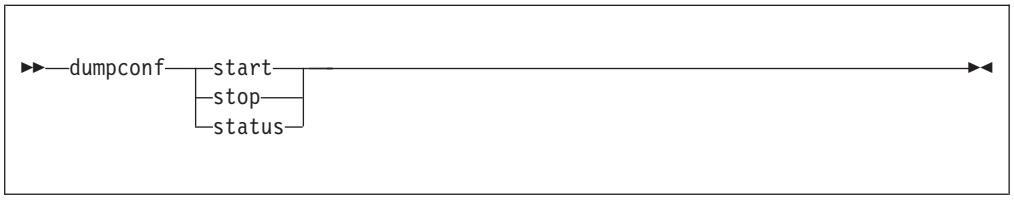
The service is installed as a script under `/etc/init.d/dumpconf` and reads the configuration file `/etc/sysconfig/dumpconf`.

**Note:** kdump does not depend on **dumpconf** and cannot be enabled or disabled with **dumpconf**. If kdump was set up for your production system, dump tools as configured with **dumpconf** are used only if the integrity check for kdump fails. With kdump set up, you can use **dumpconf** to enable or disable backup dump tools. See also “Failure recovery and backup tools” on page 7.

To enable the **dumpconf** service, issue:

```
# chkconfig --add dumpconf
```

### dumpconf service syntax



## Parameters

**start**
enables the configuration that is defined in `/etc/sysconfig/dumpconf`.

**stop**
disables the **dumpconf** service.

**status**
shows current configuration status of the **dumpconf** service.

**-h or --help**
displays a short usage text on console. To view the man page, enter **man dumpconf**.

**-v or --version**
displays the version number on console, and exits.

## Keywords for the configuration file

**ON_PANIC**
Shutdown action to be taken if a kernel panic or PSW restart occurs. Possible values are:

**dump** dumps Linux and stops the system.

**reipl** reboots Linux.

**dump_reipl**
dumps Linux and reboots the system.

**vmcmd**
executes the specified CP commands and stops the system.

**stop** stops Linux (default).

**DELAY_MINUTES**
specifies the number of minutes that the activation of the **dumpconf** service is to be delayed. The default is zero.

Using **reipl** or **dump_reipl** actions with **ON_PANIC** can lead to the system looping with alternating IPLs and crashes. Use **DELAY_MINUTES** to prevent such a loop. **DELAY_MINUTES** delays activating the specified panic action for a newly started system. After the specified time elapses, the **dumpconf** service activates the specified panic action. This action is taken should the system subsequently crash. If the system crashes before the time elapses, the previously defined action is taken. If no previous action was defined, the default action (STOP) is performed.

**VMCMD_<*X*>**
specifies a CP command, <*X*> is a number from one to eight. You can specify up to eight CP commands that are executed in case of a kernel panic or PSW restart. z/VM commands, device addresses, and names of z/VM guest virtual machines must be uppercase.

**DUMP_TYPE**
specifies the type of dump device. Possible values are ccw and fcp.

**DEVICE**
specifies the device number of dump device.

**WWPN**
specifies the WWPN for SCSI dump device.

**LUN** specifies the LUN for SCSI dump device.

**BOOTPROG**
specifies the boot program selector.

**BR_LBA**
specifies the boot record logical block address.

## Example configuration files for the dumpconf service

- Example configuration for a CCW dump device (DASD) using **reipl** after dump and **DELAY_MINUTES**:

```
ON_PANIC=dump_reipl
DUMP_TYPE=ccw
DEVICE=0.0.4714
DELAY_MINUTES=5
```

- Example configuration for FCP dump device (SCSI disk):

```
ON_PANIC=dump
DUMP_TYPE=fcp
DEVICE=0.0.4711
WWPN=0x5005076303004712
LUN=0x4713000000000000
BOOTPROG=0
BR_LBA=0
```

- Example configuration for re-IPL if a kernel panic or PSW restart occurs:

```
ON_PANIC=reipl
```

- Example of sending a message to the z/VM guest virtual machine "MASTER", executing a **CP VMDUMP** command, and rebooting from device 4711 if a kernel panic or PSW restart occurs:

```
ON_PANIC=vmcmd
VMCMD_1="MSG MASTER Starting VMDUMP"
VMCMD_2="VMDUMP"
VMCMD_3="IPL 4711"
```

z/VM commands, device addresses, and names of z/VM guest virtual machines must be uppercase.

## Examples for using the dumpconf service

Use the **dumpconf** service to enable and disable the configuration.

- To enable the configuration:

```
# service dumpconf start
ccw dump device configured. "dump" on panic configured.
```

- To display the status:

```
# service dumpconf status
type....: ccw
device..: 0.0.4714
on_panic: dump
```

- To disable dump on panic:

```
# service dumpconf stop
Dump on panic is disabled now
```

- To display the status again and check that the status is now stopped.

```
# service dumpconf status
on_panic: stop
```

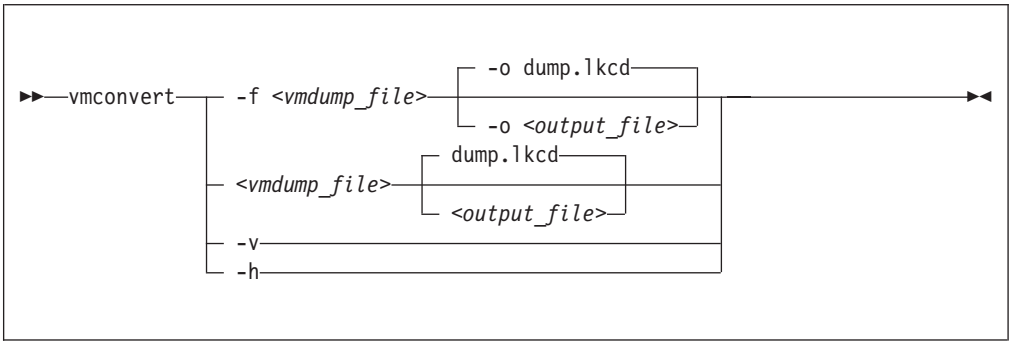
## crash - Analyze kernel dumps

The **crash** tool is a GPL-licensed tool that is maintained by Red Hat. For more information, see the tool online help.

## vmconvert - Convert z/VM VMDUMPS for Linux

The **vmconvert** tool converts a dump that was created with VMDUMP into a file that can be analyzed with **crash**.

### vmconvert syntax




### Parameters

*<vmdump_file>* **or -f** *<vmdump_file>* **or --file** *<vmdump_file>*
specifies the VMDUMP created dump file to be converted.

*<output_file>* **or -o** *<output_file>* **or --output** *<output_file>*
specifies the name of the dump file to be created. The default is `dump.lkcd`.

**-v or --version**
displays the tool version.

**-h or --help**
displays the help information for the command.

### Example

To convert a VMDUMP-created dump file `vmdump1` into a dump file `dump1.lkcd` that can be processed with **crash** issue:

```
# vmconvert -f vmdump1 -o dump1.lkcd
```

You can also use positional parameters:

```
# vmconvert vm.dump lkcd.dump
vmdump information:
  architecture: 32 bit
  date........: Fri Feb 18 11:06:45 2005
  storage.....: 16 MB
  cpus........: 6
   16 of 16 |################################################| 100%
'lkcd.dump' has been written successfully.
```

## vmur - Receive dumps from the z/VM reader

The **vmur** command can receive a VMDUMP file from the z/VM reader and convert it into a file that can be analyzed with **crash**.

Issue a command of the following form:

```
# vmur receive -c <spool ID> <dump file name>
```

### Parameters

*<spool ID>*
specifies the VMDUMP file spool ID.

*<dump file name>*
specifies the name of the output file to receive the reader spool file's data.

For more details, see the **vmur** man page and *Device Drivers, Features, and Commands on SUSE Linux Enterprise Server 12*, SC34-2745

### Example

To receive and convert a VMDUMP spool file with spool ID 463 to a file named `dump.lkcd` on the Linux file system in the current working directory:

```
# vmur rec -c 463 dump.lkcd
```

# Appendix D. Preparing for analyzing a dump

To analyze your dump with **crash**, additional files are required.

If you need to send your dump for analysis, it might be good to include these additional files with the dump file. Your distribution typically provides the additional files in RPMs.

If the dump is to be analyzed with **crash**, include:

- vmlinux (text): Contains addresses of kernel symbols
- vmlinux (debug): Contains datatype debug information

## SLES debug files

The SLES debug files are:

*Table 4. SUSE Linux Enterprise Server debug file names.*

| Debug file | Path |
|---|---|
| System.map | /boot/System.map-3.12.<*xx*>-<*y*>-default |
| vmlinux (text) | /boot/vmlinux-3.12.<*xx*>-<*y*>-default.gz |
| vmlinux (debug) | /usr/lib/debug/boot/vmlinux-3.12.<*xx*>-<*y*>-default.debug |

The RPMs that contain the debuginfo files are:

*Table 5. SUSE Linux Enterprise Server debuginfo RPM names.*

| SLES version | RPM |
|---|---|
| SLES 12 | • kernel-default-3.12.<*xx*>-<*y*>.s390x.rpm<br>• kernel-default-debuginfo-3.12.<*xx*>-<*y*>.s390x.rpm |

For debugging problems that are related to kernel modules, additional RPMs might be required.

# Appendix E. Installing the DASD or SCSI dump tool with YaST

You can prepare ECKD DASD and SCSI dump devices with the YaST tool.

## Procedure

1. Start YaST:

```
# yast
```

2. Ensure that the dump DASDs or SCSI devices (LUNs) are activated.
   a. Enter menu: **Hardware** > **DASD** or **Hardware** > **Zfcp**
   b. Activate the correct DASDs or SCSI disks, if not already activated.
3. Prepare the DASDs or SCSI device for dump use:
   a. Enter menu: **Hardware** > **Dump Devices**.
   b. Select the dump DASDs or SCSI device.
   c. Click **Create** to prepare the DASDs or SCSI disk.
4. A message is displayed when your dump device has been prepared.

## What to do next

After a dump device has been prepared it is possible to configure the **dumpconf** service for automatic dump on panic:

1. Enter menu: **Hardware** > **On Panic**.
2. Configure the **dumpconf** settings, see "dumpconf - Configure panic or PSW restart action" on page 58.

**Note:** The dump-related YaST dialogues can also be entered directly with these commands:

- **yast dasd**
- **yast zfcp**
- **yast kdump**
- **yast dump**
- **yast onpanic**

# Appendix F. How to detect guest relocation

Information about guest relocations are stored in the s390 debug feature (s390dbf). You can access this information in a kernel dump or from a running Linux instance.

## About this task

You can detect if a Linux instance has been moved to another guest virtual machine or LPAR. One available mechanism for guest relocation is z/VM Single System Image (SSI).

You can access the s390 debug feature `lgr` from a live system or with the **crash** tool from a kernel dump. When the debug feature contains only one entry, no relocation has been detected and the entry identifies the boot virtual guest machine. For each detected relocation one additional entry is written.

## Procedure

Choose the method that suits your purpose:

- Use the **crash** tool to read from a kernel dump. Issue a command of this form:

```
# crash <vmlinux files> <dump file>
crash> s390dbf lgr hex_ascii
```

- Use the **cat** command on a live system to read the debugfs entry for `lgr`. Issue a command of this form:

```
# cat /sys/kernel/debug/s390dbf/lgr/hex_ascii
```

## Example

Assume that one relocation of the guest virtual machine ZVMGUEST has been detected from a z/VM in LPAR VM000A to a z/VM in LPAR VM000B. You can see this in the kernel dump:

```
# crash vmlinux dump
crash> s390dbf lgr hex_ascii
00 01317816806:277332 3 - 00 .. | ... IBM281700000000000EAA1402 ... VM000A ... ZVMGUEST
00 01317866806:277332 3 - 00 .. | ... IBM281700000000000EAA1402 ... VM000B ... ZVMGUEST
```

Alternatively, you can see such a relocation from a running system:

```
# cat /sys/kernel/debug/s390dbf/lgr/hex_ascii
00 01317816806:277332 3 - 00 .. | ... IBM281700000000000EAA1402 ... VM000A ... ZVMGUEST
00 01317866806:277332 3 - 00 .. | ... IBM281700000000000EAA1402 ... VM000B ... ZVMGUEST
```

For more information about the complete s390dbf record, see the `struct os_info` definition in the Linux kernel source code.

# Accessibility

Accessibility features help users who have a disability, such as restricted mobility or limited vision, to use information technology products successfully.

## Documentation accessibility

The Linux on System z publications are in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when you use the PDF file and want to request a Web-based format for this publication, use the Readers' Comments form in the back of this publication, send an email to eservdoc@de.ibm.com, or write to:

IBM Deutschland Research & Development GmbH
Information Development
Department 3282
Schoenaicher Strasse 220
71032 Boeblingen
Germany

In the request, be sure to include the publication number and title.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

## IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility at

`www.ibm.com/able`

# Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

## Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at
www.ibm.com/legal/copytrade.shtml

Adobe is either a registered trademark or trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

# Index

## Special characters



## A



## C



## D



## E



## F



## G



## H



## I

# Readers' Comments — We'd Like to Hear from You

**Linux on System z**
**Using the Dump Tools**
**on SUSE Linux Enterprise Server 12**

**Publication No. SC34-2746-00**

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send your comments via email to: eservdoc@de.ibm.com

If you would like a response from IBM, please fill in the following information:

| | |
|---|---|
| Name | Address |
| Company or Organization | |
| Phone No. | Email address |

**Readers' Comments — We'd Like to Hear from You**
SC34-2746-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Deutschland Research & Development GmbH
Information Development
Department 3282
Schoenaicher Strasse 220
71032 Boeblingen
Germany

Fold and Tape | **Please do not staple** | Fold and Tape

SC34-2746-00

Cut or Fold
Along Line

SC34-2746-00