Linux on System z

Using the Dump Tools

Development stream (Kernel 3.5)

SC33-8412-11

Linux on System z

Using the Dump Tools

Development stream (Kernel 3.5)

SC33-8412-11

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

This edition applies to the Linux on System z Development stream for kernel 3.5, s390-tools version 1.19, and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Summary of changes

This revision reflects changes to the Development stream for kernel 3.5.

Updates for kernel 3.5

This revision (SC33-8412-11) contains changes related to the kernel 3.5 release.

New Information

- You can now create a kernel dump for a live system without disruption. See Chapter 8, "Creating live-system dumps with zgetdump," on page 33.

Changed Information

- None.

This revision also includes maintenance and editorial changes.

Deleted Information

- None.

Updates for kernel 3.4

This revision (SC33-8412-10) contains changes related to the kernel 3.4 release.

New Information

- You can now detect guest relocations in a kernel dump or from a live system. See Appendix D, "How to detect guest relocation," on page 67.

Changed Information

- None.

This revision also includes maintenance and editorial changes.

Deleted Information

- None.

Updates for kernel 3.2

This revision (SC33-8412-09) contains changes related to the kernel 3.2 release.

New Information

- You can now use kdump for Linux on System z®. See Chapter 2, "Using kdump," on page 5.

Changed Information

- None.

This revision also includes maintenance and editorial changes.

Deleted Information

- None.

About this document

This document describes tools for obtaining dumps of Linux for IBM® System z instances. It describes how to use DASD, tape, and SCSI dump devices, as well as how to use kdump and VMDUMP.

In this document, System z is taken to include zSeries® in 64- and 31-bit mode.

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

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

Authority

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

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

Other Linux on System z publications

You can find Linux on System z publications on developerWorks and on the IBM Information Center for Linux.

These publications are available on developerWorks at

www.ibm.com/developerworks/linux/linux390/documentation_dev.html

- *Device Drivers, Features, and Commands*, SC33-8411
- *Using the Dump Tools*, SC33-8412
- *How to Improve Performance with PAV*, SC33-8414
- *How to use FC-attached SCSI devices with Linux on System z*, SC33-8413
- *How to use Execute-in-Place Technology with Linux on z/VM*, SC34-2594
- *How to Set up a Terminal Server Environment on z/VM*, SC34-2596
- *Kernel Messages*
- *libica Programmer's Reference*, SC34-2602

These publications are available in the System z section of the IBM Information Center for Linux at

publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaaf/lcon_System_z.htm

- *Linux on System z Troubleshooting*, SC34-2612
- *Linux Health Checker User's Guide*, SC34-2609
- *Kernel Messages*

Device nodes used in this publication

There can be multiple device nodes for the same device.

All examples in this publication use the standard device nodes for DASD, tape, and SCSI devices. If you are using a Linux distribution that provides udev, you can also use the device nodes that udev creates for you. See your distribution documentation to find out which nodes are available.

Chapter 1. Introduction

Different tools can be used for obtaining dumps of Linux on System z instances.

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	No	No	No	No	Yes	Yes
System size (See also "Dump size")	Large	Small	Large	Large	Large	Small	Large
Speed	Fast	Fast	Fast	Fast	Slow	Slow	Fast
Medium	Any available medium	ECKD™ or FBA(See 1) DASD	ECKD DASD	Linux file system	Tape cartridges	z/VM reader	Any available medium
Compression possible	While writing	No	No	While writing	Yes (See "Dump size")	No	No
Dump filtering possible	While writing	When copying	When copying	When copying	When copying	When copying	No
Disruptive(See 2)	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.

Dump size

The dump size depends on the size of the system for which the dump is to be created. Except for kdump, 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 27 provides 27 GB.

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 9, "Handling large dumps," on page 37 for information specific to large dumps.

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. If you install dump tools that are compiled for 64-bit, you can create both 64-bit and 31-bit Linux dumps. If you install dump tools that are compiled for 31-bit, you can create 31-bit Linux dumps only.

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 is destructive, that is, the running Linux operating system is killed. The IPL process writes the system memory to the IPL device (DASD and tape) or directly to a file on a SCSI disk.

You can configure a dump device that is automatically used when a kernel panic occurs. For more information, see "The dumpconf service" on page 61.

All examples for installing stand-alone tools by using a **zipl** configuration file assume that `/etc/zipl.conf` is used as the configuration file and that `/etc/zipl.conf` is the default configuration file.

For more information on **zipl**, refer to the **zipl** man page and to the **zipl** description in *Device Drivers, Features, and Commands*, SC33-8411. You can find the latest version of this document on developerWorks at:

`www.ibm.com/developerworks/linux/linux390/documentation_dev.html`

VMDUMP

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 on VMDUMP see *z/VM CP Commands and Utilities Reference*, SC24-6175.

kdump

The kdump feature 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.

IPLing the production system again clears the dump in memory, so the dump must be saved to persistent storage before a subsequent re-IPL.

Because kdump can write dumps though 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 41.

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 Load the kdump kernel and the initial RAM disk into the memory of the production system (**kexec**)	Write the stand-alone dump tool to the dump device (**zipl**) Define the panic shutdown action (**dumpconf**)	Define the panic shutdown action (**dumpconf**)	None
Dump trigger	**Automatic:** Kernel panic **Initiated by operator:** PSW restart	**Automatic:** Kernel panic **Initiated by operator:** IPL of the dump device	**Automatic:** Kernel panic **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	Using **zgetdump** from a new Linux instance	Using **vmur -c** from a new Linux instance	Through `/dev/mem`
Copying the initial dump to the final dump store (and releasing the initial dump space)	Copied from the kdump instance to any available storage	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	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 Linux on System z.

Before you begin

For using kdump, you require:

- A Linux kernel image that has been compiled with the common code kernel configuration options CONFIG_KEXEC, CONFIG_CRASH_DUMP, and CONFIG_PROC_VMCORE. This requirement applies to both, the production system for which kdump is set up and to the kdump image.
- An initial kdump RAM disk if required by the kdump kernel image.
- A kexec package with kdump support.
- A System z9 or later mainframe system.

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 kernel and its initial RAM disk (initrd) are loaded into the reserved memory area with the **kexec** tool.

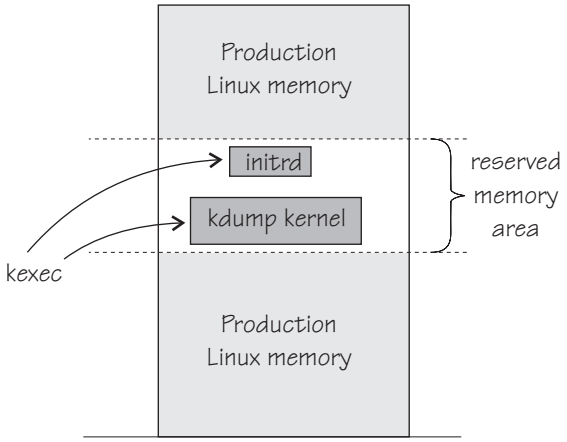


Figure 1. Running production system with preloaded kdump kernel and initial RAM disk

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 has been 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`.

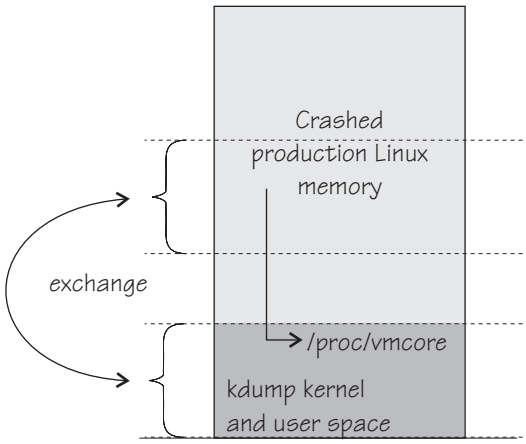


Figure 2. Running kdump kernel

This process is fast, because the kdump kernel is started from memory, and no dump data needs to be copied up to this stage. Typically, the **makedumpfile** tool is now used to write a filtered and compressed version of the dump to a file on persistent storage, locally or over a network. Again, this saves time, because the dump is reduced in size while it is written or transferred.

After the dump is written, the production system can be IPLed again.

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 amount of 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 "The dumpconf service" on page 61.

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 "The zgetdump tool" on page 57.

Setting up kdump

Before you can use kdump, you must load the kdump kernel into a reserved memory area.

Before you begin

- The steps that are described here are typically performed for you by your distribution or through distribution-specific configuration tools.
- You need a kdump kernel image and initial RAM disk.
- The kexec-tools package must be installed.

About this task

The reserved memory area must be sufficiently large to accommodate the kdump system, including user space, when it is booted. If too little memory is reserved, kdump itself will crash if booted (see "Failure recovery and backup tools").

Procedure

1. For your production system, code the `crashkernel=` kernel parameter according to this syntax:

   `crashkernel=<size>@<offset>`

where <*size*> specifies the amount of memory to be reserved and <*offset*> the beginning of the memory range. The values are integers, optionally followed by K for kilobyte, M for megabyte, or G for gigabyte. The values are adjusted to multiples of 1 MB.

The specified memory area, <*offset*> through <*offset*> + <*size*>, and a corresponding memory area `0` through <*size*>, must both be available and must both not contain any memory holes.

If you omit `@<offset>`, a suitable offset is chosen for you. If you specify an offset, it must be greater than the size of the reserved memory. For example, the following specification reserves 128 MB for the kdump kernel at an automatically selected suitable offset.

```
crashkernel=128M
```

The following specification reserves 128 MB for the kdump kernel at an offset of 256 MB.

```
crashkernel=128M@256M
```

Optionally, you can make the specification of reserved memory size and offset dependent on the size of the available memory. See `Documentation/kernelparameters.txt` in the Linux source tree for details about how to do this and about further details of the `crashkernel=` kernel parameter.

2. Boot your production system.
3. Optional: Issue the following command to confirm that a memory area has been reserved for the kdump kernel:

```
# cat /proc/iomem
```

The command output must include a memory range for `Crash kernel`.

4. Load the kdump kernel with the **kexec** command according to this syntax:

```
# kexec -p <image> --initrd <initrd> --command-line "<kparms>"
```

where:

<*image*>
specifies the kdump kernel image.

<*initrd*>
specifies the initial RAM disk of the kdump kernel. This specification can be omitted if the kdump kernel does not require an initial RAM disk.

<*kparms*>
specifies kernel parameters for the kdump kernel.

Example:

```
# kexec -p /boot/kdump.image --initrd /boot/kdump.initrd \
--command-line="dasd=eb90 root=/dev/ram0 maxcpus=1"
```

Results

A kernel panic or PSW restart now triggers an automatic dump process with kdump.

What to do next

As a backup, you can set up a stand-alone dump tool in addition to kdump. See "The dumpconf service" on page 61 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. When your Linux system does not respond and kdump is not triggered automatically, depending on your system environment, there are additional methods for triggering 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 "The dumpconf service" on page 61).

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 49 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*, SC33-8411 about how to configure the z/VM watchdog.

Results

The dump process loads the kdump kernel from which you can access the dump.

What to do next

Because the dump is initially held in memory, you must process the dump before IPLing your production system. A new IPL clears the dump.

Accessing the dump

After the kdump kernel has started, the dump can be accessed and copied from memory to a file on persistent storage. Typically, this process is automated through tools in an initial RAM disk that is provided with your distribution.

Before you begin

The kdump user space is typically set up to automatically perform the following actions for you:

1. Reading the dump of the production system

2. Filtering and compressing the dump
3. Saving the dump to a file on persistent storage
4. Rebooting the production system

The steps that follow describe how to perform these tasks from the command line.

About this task

On the running kdump kernel, the dump can be accessed through two virtual files:

/proc/vmcore
represents the dump in Executable and Linkable Format (ELF) core format and includes memory, CPU register, and vmcoreinfo information.

/dev/oldmem
represents the dump in form of an unstructured linear memory.

The following description uses the more convenient ELF format at `/proc/vmcore`.

Procedure

1. Optional: Use **zgetdump** to display information about the dump.

```
# zgetdump -i /proc/vmcore
```

2. Optional: Analyze the dump at `/proc/vmcore` with the **crash** tool, as usual. This means that you cannot IPL your production system until you have finished the analysis.
3. Copy the dump to persistent storage. The following examples illustrate some of the options:

   This example copies the entire dump to a device that is available at `/dumps/dump.elf` in the Linux file system:

```
# cp /proc/vmcore /dumps/dump.elf
```

   This example uses scp to copy the entire dump to a file `/dumps/dump.elf` in the file system of another system, `dumpstore`:

```
# scp /proc/vmcore user@dumpstore:/dumps/dump.elf
```

   This example uses **makedumpfile** to copy a compressed and filtered version of the dump to a file `/dumps/dump.kdump` in the Linux file system:

```
# makedumpfile -c -d 31 /proc/vmcore /dumps/dump.kdump
```

   This example uses **makedumpfile** to copy a compressed and filtered version of the dump to a file `/dumps/dump.kdump` in the file system of another system, `dumpstore`:

```
# makedumpfile -F -c -d 31 /proc/vmcore | \
ssh user@dumpstore "cat > /dumps/dump.kdump"
```

   The **makedumpfile -c** option compresses the dump, the **-F** option converts it to the flat format required for transferring the file, and **-d** filters unwanted data from the dump. The number that follows **-d** must be in the range 0 through 31. The number represents a bit mask that specifies which page types to filter out.

   For more details, see the man page for **makedumpfile**.

4. Issue `reboot`, to start the production system again.

Results

You can now analyze the dump with **`crash`**.

Chapter 3. Using a DASD dump device

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

About this task

DASD dumps are written directly to a DASD partition that has not been 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 multi-volume dump tool, see Chapter 4, "Using DASD devices for multi-volume dump," on page 17.

About this task

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

The steps 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**. A block size of 4 KB is recommended. 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 using the **zipl** command. You can specify the dump device on the command line or use a configuration file. Command line example:

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

Configuration file example:

a. Edit `/etc/zipl.conf` to add the following lines:

```
[dump_dasd]
dumpto=/dev/dasdc1
```

b. Issue:

```
# zipl dump_dasd
```

Note: When using an ECKD-type DASD formatted with the traditional Linux disk layout ldl, the dump tool must be reinstalled using **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 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
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 47 for more details.

Copying the dump from DASD with zgetdump

You can copy a DASD 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 DASD device `/dev/dasdc1` and you want to copy it to a file named `dump_file`:

```
# zgetdump /dev/dasdc1 > dump_file
```

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 60 for an example.

For general information about **zgetdump**, see "The zgetdump tool" on page 57 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, install the stand-alone dump tool on each DASD device involved, perform the dump process, and copy the dump to a file in a Linux file system.

About this task

Multi-volume dumps are possible on 64-bit systems only.

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.
- Be formatted with the compatible disk layout (cdl, the default option when using the **dasdfmt** command.)

You can use any block size, even mixed block sizes. However, to speed up the dump process and to reduce wasted disk space, use block size 4096.

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.

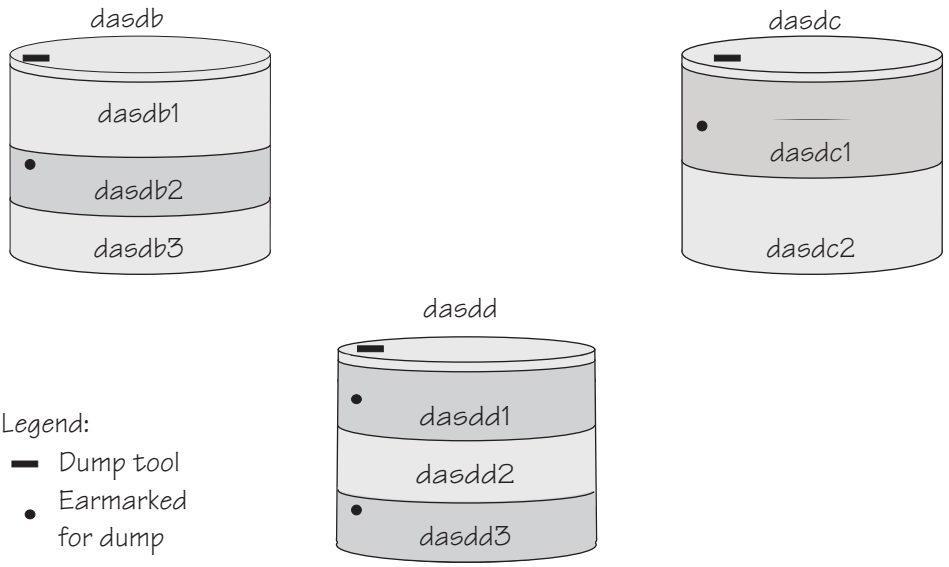


Figure 3. Three DASD volumes with four partitions for a multivolume dump

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

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

You can define a maximum of three partitions on one DASD. All three 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**. Specify cdl (compatible disk layout), which is the default. Preferably, use a block size of 4 KB:

   ```
   # 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. You can specify the dump list on the command line or use the zipl configuration file.

   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)?
   ```

   zipl configuration file example:

   a. Copy `mvdump.conf` to `/boot/mvdump.conf` and edit `/etc/zipl.conf` to add the following lines:

```
[multi_volume_dump]
mvdump=/boot/mvdump.conf
```

b. Issue:

```
# zipl multi_volume_dump
```

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
'/dev/dasdc' is part of Version 1 multi-volume dump,
which is spread along the following DASD volumes:
0.0.4711 (online, valid)
0.0.4712 (online, valid)
Dump size limit: none
Force option specified: no
```

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 55. For more details about the **zipl** command, see *Device Drivers, Features, and Commands*, SC33-8411.

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 > multi_volume_dump_file
Format Info:
  Source: s390mv
  Target: s390

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 60 for an example.

Chapter 5. Using a tape dump device

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

About this task

The following tape devices are supported:

- 3480
- 3490
- 3590
- 3592

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).

About this task

The examples assume that `/dev/ntibm0` is the tape device 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 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 should go into disabled wait.

The following PSW indicates that the dump was taken 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 47 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 using the **zgetdump** tool.

Before you begin

You must have installed the **mt** utility.

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 is the dump tool itself).

   For example:

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

Using the zgetdump tool to copy the dump

Use the **zgetdump** tools 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 22).

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 the dump is on tape device `/dev/ntibm0`

Procedure

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

```
# zgetdump /dev/ntibm0 > dump_file
```

For general information on **zgetdump**, see "The zgetdump tool" on page 57 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 59.

Chapter 6. Using a SCSI dump device

You can use SCSI disks that are accessed through the zfcp device driver as dump devices. SCSI disk dumps are written as files in an existing file system on the dump partition. No copying is necessary.

Installing the SCSI disk dump tool

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

Before you begin

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

About this task

The SCSI dump tool (also referred to as the SCSI Linux System Dumper, or SD) is written to one partition, referred to here as the *target partition*. The dump can be written to a second partition, the *dump partition*, provided it is on the same physical disk. Only the target partition need be mounted when **zipl** is run. In a single-partition configuration, the target partition is also the dump partition.

SCSI dump tool parameters

When installing the SCSI disk dump tool, the following parameters can be specified using the `-P` option in the **zipl** command line.

Parameters

dump_dir=/<*directory*>
: Path to the directory (relative to the root of the dump partition) to which the dump file is to be written. This directory is specified with a leading slash. The directory must exist when the dump is initiated.

  For example, if the dump partition is mounted as `/dumps`, and the parameter **dump_dir=/mydumps** is defined, the dump directory would be accessed as `/dumps/mydumps`.

  The default is `/` (the root directory of the partition).

dump_compress=gzip|none
: Dump compression option. Compression can be time-consuming on slower systems with a large amount of memory.

  The default is **none**.

dump_mode=interactive|auto
: Action taken if there is no room on the file system for the new dump file. **interactive** prompts the user to confirm that the dump with the lowest number is to be deleted. **auto** automatically deletes this file.

  The default is **interactive**.

In rare cases, you might want to complement or overwrite the SCSI dump tool parameters that have been configured with **zipl**. For example, you might want to change the dump mode setting when you initiate the dump. How you specify such

parameters depends on whether your Linux instance runs in LPAR mode or as a z/VM guest. For more information, see the SCSI examples in Appendix A, "Examples for initiating dumps," on page 47.

Example 1: Combined dump and target partition

A single partition on a SCSI device can be used as both the dump partition and target partition.

About this task

This example assumes that `/dev/sda` is a SCSI device that contains no data and is to be used exclusively as a dump device. Because no other data is to be stored on the device, a single partition is created that serves as both dump and target partition. The example also shows how to use the `dump_compress` parameter to generate the dump in **`gzip`** format.

Procedure

1. Create a single partition with **`fdisk`**, using the PC-BIOS layout:

   For example:

   ```
   # fdisk /dev/sda
   ```

   The created partition is `/dev/sda1`.
2. Format this partition with a file system that is supported by the SCSI dump tool (for example, ext2 or ext3).

   For example:

   ```
   # mke2fs /dev/sda1
   ```

3. Mount the partition at a mount point of your choice and create a subdirectory to hold the dump files.

   For example:

   ```
   # mount /dev/sda1 /dumps
   # mkdir /dumps/mydumps
   ```

4. Install the dump tool using the **`zipl`** command. You can specify the dump device on the command line or use a configuration file.

   Command line example:

   ```
   # zipl -D /dev/sda1 -t /dumps -P "dump_dir=/mydumps dump_compress=gzip"
   ```

   Configuration file example:

   a. Edit `/etc/zipl.conf` to add the following lines:

      ```
      [scsidump]
      target=/dumps
      dumptofs=/dev/sda1
      parameters="dump_dir=/mydumps dump_compress=gzip"
      ```

   b. Issue **`zipl`**:

      ```
      # zipl scsidump
      ```

5. Unmount the file system:

```
# umount /dumps
```

Results

When you IPL /dev/sda1 using boot program selector 1 or 0 (default), the dump is written to directory mydumps on partition 1 of /dev/sda. The boot program selector is located on the load panel, see Figure 6 on page 52 for an example.

Example 2: Menu configuration with separate dump and target partitions

You can create a menu configuration for a SCSI dump device. To keep the dump data separated from other data, use separate dump and target partitions.

About this task

This example assumes that a SCSI device /dev/sda is to be used as a dump device. In the example, the dump configuration is part of a menu configuration. Menu configurations are specified in a zipl configuration file and have a common target directory that is specified in the menu section of the configuration file. To keep the dumps separated from other data, separate dump and target partitions are used. The example assumes that there are already three partitions:

- /dev/sda1 is the production partition and mounted as the root file system.
- /dev/sda2 is the target partition, has been formatted with the PC-BIOS disk layout, and is mounted under /boot. /boot contains files for two Linux boot configurations (parmfile, image-1, and image-2).
- /dev/sda3 is the dump partition and has been formatted with the PC-BIOS disk layout. The dump files are to be written to the root directory of the dump partition.

Procedure

1. On the dump partition, create a file system that is supported by the SCSI dump tool (for example, ext2 or ext3):

```
# mke2fs /dev/sda3
```

2. Edit /etc/zipl.conf to add the following lines:

```
[ipl1]
image=/boot/image-1
parmfile=/boot/parmfile
target=/boot

[ipl2]
image=/boot/image-2
parmfile=/boot/parmfile
target=/boot

[scsidump1]
dumptofs=/dev/sda3
parameters="dump_compress=gzip"
target=/boot

# Menu containing all 3 configurations
:menu1
1=ipl1
```

```
2=ipl2
3=scsidump1
default=1
target=/boot
```

3. Install the menu configuration, including the dump tool, by issuing:

```
# zipl --menu menu1
```

Results

When you specify the "scsidump1" configuration at IPL-time using boot program selector 3, the dump configuration is used and a system dump is initiated. The boot program selector is located on the load panel, see Figure 6 on page 52 for an example.

For more information on using a configuration file, see the `zipl.conf` man page or refer to *Device Drivers, Features, and Commands*, SC33-8411.

Initiating a SCSI dump

To initiate the dump, IPL the SCSI dump tool 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.

Procedure

IPL the SCSI dump tool.
See Appendix A, "Examples for initiating dumps," on page 47 for more details.

Results

The dump process creates a new dump file in the dump directory. All dumps are named `dump.`<*n*> where <*n*> is the dump number. A new dump receives the next highest dump number out of all dumps in the dump directory (see the `dump_dir` parameter under "SCSI dump tool parameters" on page 25).

For example, if there are already two dump files named `dump.0` and `dump.1` in the dump directory, the new dump will be named `dump.2`.

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

You do not need to convert the dump or copy it to a different medium. To access the dumps, mount the dump partition.

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 dump.0
General dump info:
  Dump format........: lkcd
  Version............: 8
  System arch........: s390x (64 bit)
  CPU count (online).: 2
  CPU count (real)...: 2
  Dump memory range..: 1024 MB

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

Chapter 7. Creating dumps on z/VM with VMPDUMP

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.

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

You can use the **vmur** command under Linux or the **DUMPLOAD** command under CMS to copy the dump file.

Using the vmur command

You can 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**.

Using the DUMPLOAD command

You can use the **DUMPLOAD** command under CMS to access the dump.

About this task

The **DUMPLOAD** command copies the dump from the z/VM reader to the CMS file system.

What to do next

From the CMS file system, you can now transfer the dump to a Linux file system, for example with **ftp**.

Chapter 8. Creating live-system dumps with zgetdump

If you require a kernel dump of a Linux, 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

- A Linux kernel image that was compiled with the common code kernel configuration option CONFIG_STRICT_DEVMEM=n.
- 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`.

Procedure

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

```
# zgetdump -i /dev/mem
General dump info:
 Dump format........: devmem
 Dump method........: live
 UTS node name......: mylnxsys
 UTS kernel release.: 3.3.0
 UTS kernel version.: #68 SMP PREEMPT Thu Mar 22 10:21:30 CET 2012
 System arch........: s390x (64 bit)
 Dump memory range..: 512 MB

Memory map:
 0000000000000000 - 000000001fffffff (512 MB)
```

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

```
# nice -n -20 zgetdump /dev/mem > dump.s390
```

You can specify a target dump format with the `-f` option:

```
# zgetdump /dev/mem -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.: 3.3.0
 UTS kernel version.: #68 SMP PREEMPT Thu Mar 22 10:21:30 CET 2012
 System arch........: s390x (64 bit)
 Dump memory range..: 512 MB

Memory map:
 0000000000000000 - 000000001fffffff (512 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/mem -f elf > dump.elf
Format Info:
Source: devmem
Target: elf
Copying dump:
00000000 / 00000512 MB
00000110 / 00000512 MB
...
00000512 / 00000512 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."

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.

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 vmlinux vmlinux.debug
...
   KERNEL: /boot/vmlinux
 DUMPFILE: /mnt/dump.s390 [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 vmlinux vmlinux.debug
...
NOTE: minimal mode commands: log, dis, rd, sym, eval, set and exit
```

Chapter 9. Handling large dumps

This topic describes how to handle 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 need to 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 the service organization.

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

Procedure

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)

   Multi-volume 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 FBA emulated SCSI dump disk

   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. The ext2 and ext3 file system dump size limit using block size 4 KB is 2 TB.

   Note: SCSI dump compression (the dump_compress option) will create smaller dumps, but due to CPU consumption it slows down the dump speed significantly. Therefore you should use this option on large systems only if dump speed is not important for your scenario.

   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. Assuming about 100 MB per second dump speed on DASDs or SCSI disks and you have a system with 50 GB memory, the dump will take about eight 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 service 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 39
4. Send the dump.

Compressing a dump using makedumpfile

Use the **makedumpfile** tool (version 1.3.7 or higher) can be used 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. 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.

Procedure

1. Mount the dump in ELF format by performing one of these steps:
   - For a DASD dump:

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

   - For a SCSI dump:

     ```
     # zgetdump -m -f elf dump.0 /mnt
     ```

2. Create a file with a filtered and compressed version of the dump.

   Use the **makedumpfile -d** (dump level) option to excludes pages that are typically not needed to analyze a kernel problem. For dump level 31, pages containing only zeroes, pages used to cache file contents (cache, cache private), pages belonging to user-space processes, and free pages are all excluded.

   See the man page for **makedumpfile** for a description of the dump level and other options of **makedumpfile**.

   The following command accesses a dump at `/mnt/dump.elf` filters it with dump level 31, compresses it, and writes it to a file `/dumps/dump.kdump`:

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

You might want to retain a copy of the original dump file until the problem is resolved. This reserves the option to create further copies at different dump levels should any of the excluded pages be required for problem determination.

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

```
# makedumpfile --dump-dmesg  /mnt/dump.elf /dumps/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 one GB 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
     ```

   This will create 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 service 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 uncompress the dump:

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

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 have been 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 kind 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 this. While it is possible in many cases to use a pool of devices for sharing, for simplicity most of the following examples use only one dump device.

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 42).

Attach Attach and detach is used as locking mechanism (see "Sharing DASD devices under z/VM" on page 42).

vmcmd
: Use the **vmcmd** panic action (see "DASD (vmcmd panic action)" on page 43).

Alternatively, use no serialization and take the risk that dumps are overwritten, see "DASD (dump or dump_reipl panic action)" on page 43).

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

Table 3. Serialization of dump devices overview

Disk type	**DASD**		**SCSI**	
Environment	**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, 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 both the dump file and the file system on the dump device.

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.

When the Linux guest virtual machines that use the shared dump device have the permission to attach devices (that is, class B guest virtual machines) device attachment can also be used as a locking mechanism. Only one guest can attach a device at the same time. If you use one single FCP adapter for dumps on all systems, attach and detach can 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 "The dumpconf service" on page 61).

DASD (dump or dump_reipl panic action)

It is possible to share DASD devices for automatic dump on panic, but there is no serialization mechanism available.

About this task

As there is no serialization mechanism available, two systems dumping at the same time might corrupt the dumps. Normally, system crashes are rare and therefore the chance of corrupted dumps is low, but you must consider carefully if this risk is acceptable. Such a dump setup is a trade-off between reliability and resource expenses. You must consider the likelihood of two concurrent system crashes and the business impact of loosing a dump.

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 five 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 five 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 43, 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.

For automatic dump on an FCP-attached SCSI device, do not use device sharing. If multiple Linux systems write to the shared dump device at the same time, you might not only corrupt the dump file but also the file system on the dump device.

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 10 and SUSE Linux Enterprise Server 11, or between Red Hat Enterprise Linux 5 and Red Hat Enterprise Linux 6.

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 11 SP1 and SP2, prepare your dump device with the **zipl** tool from the SP1 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.

Sharing dump resources with VMDUMP

With z/VM, you can use **VMDUMP** 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 file. 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 device, 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
- Boot program selector: 3

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 bootprog 3
#cp ipl 4711 dump
```

What to do next

You can now IPL your Linux instance and resume operations.

In rare cases, you might want to overwrite or complement the existing SCSI dump tools parameters that have been configured with **zipl**. For example, you might want to change the dump mode setting. You can use a command of this form to specify SCSI dump tools parameters to be concatenated to the existing parameters:

```
#cp set dumpdev scpdata '<parameters>'
```

Enter this command before entering the **IPL** command.

In contrast to SCSI IPL configurations, where you can use a leading equal sign to replace all kernel parameters, you cannot use a leading equal sign to replace all SCSI dump tool parameters. Specifying the parameters with a leading equal sign causes the dump to fail.

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 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 content area.
2. Select the LPAR for which you want to initiate the dump.
3. In the **Tasks** area, expand **Recovery**. Proceed according to your dump tool or 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 50

   Figure 4 on page 50 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.

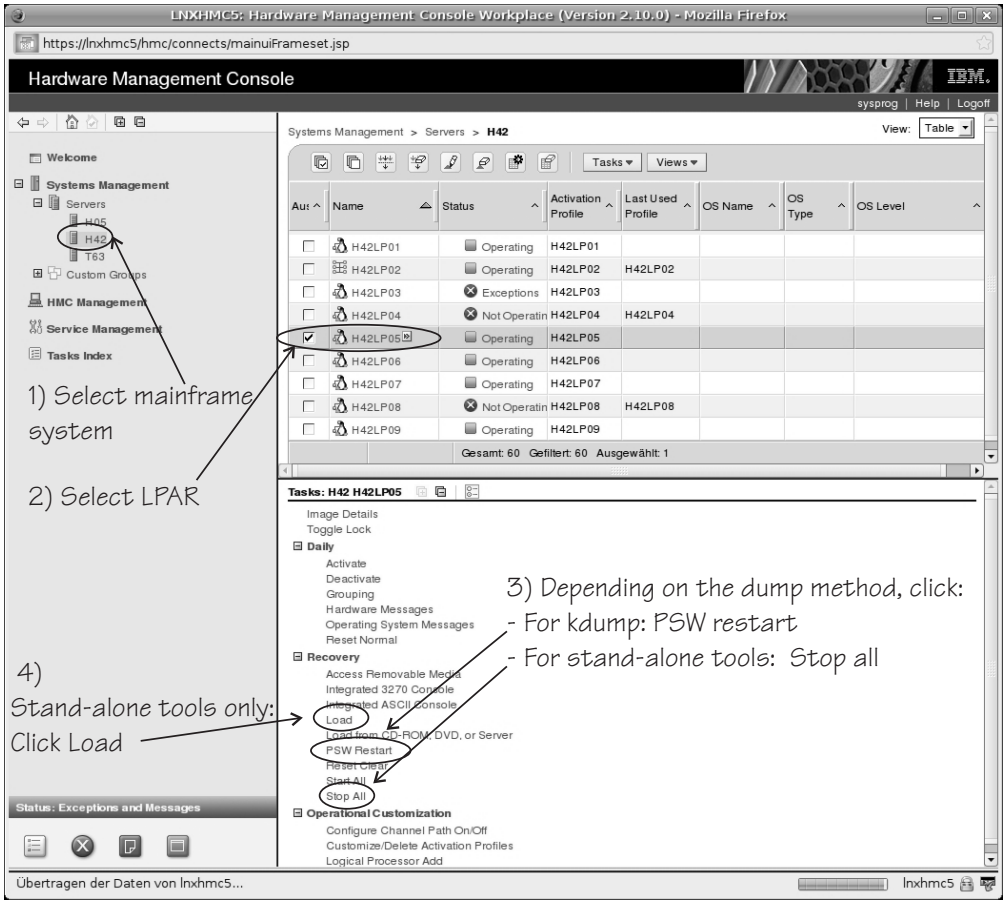


Figure 4. HMC with the **Load**,**PSW restart**, *and* **Stop all** *tasks*

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 51 shows a Load panel with all entries and selections required to start the dump process for a DASD or tape dump device.

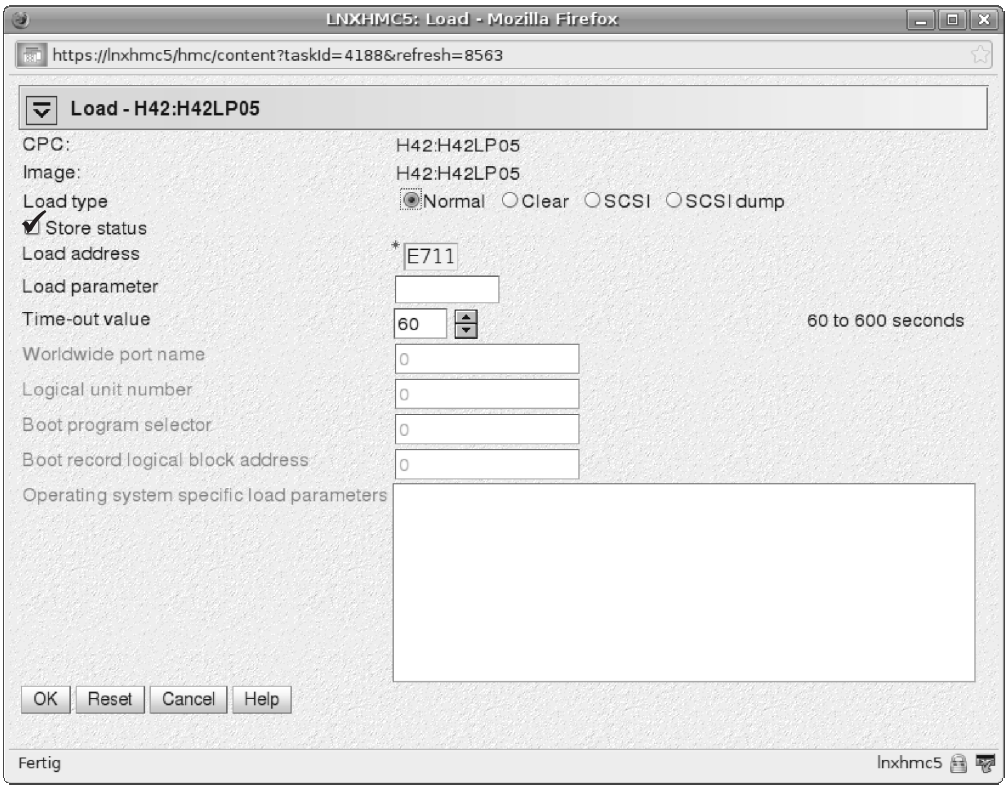


Figure 5. Load panel for dumping to DASD or tape

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 the configuration number of the dump IPL configuration in the **Boot program selector** field.

   The 'configuration number' defines the IPL or dump configuration which is to be IPLed. The numbering starts with 1 and is related to the menu of IPL/dump entries in the **zipl** configuration file for the SCSI disk. Configuration number 0 specifies the default configuration. In “Example 2: Menu configuration with separate dump and target partitions” on page 27, the dump configuration has the number 3.

f. Accept the defaults for the remaining fields.

   In rare cases, you might want to overwrite or complement the existing SCSI dump tools parameters that have been configured with **zipl**. For example, you might want to change the dump mode setting. In the **Operating system specific load parameters** field, you can specify SCSI dump tools parameters to be concatenated to the existing parameters.

   In contrast to SCSI IPL configurations, where you can use a leading equal sign to replace all kernel parameters, you cannot use a leading equal sign to replace all SCSI dump tool parameters. Specifying the parameters with a leading equal sign causes the dump to fail.

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

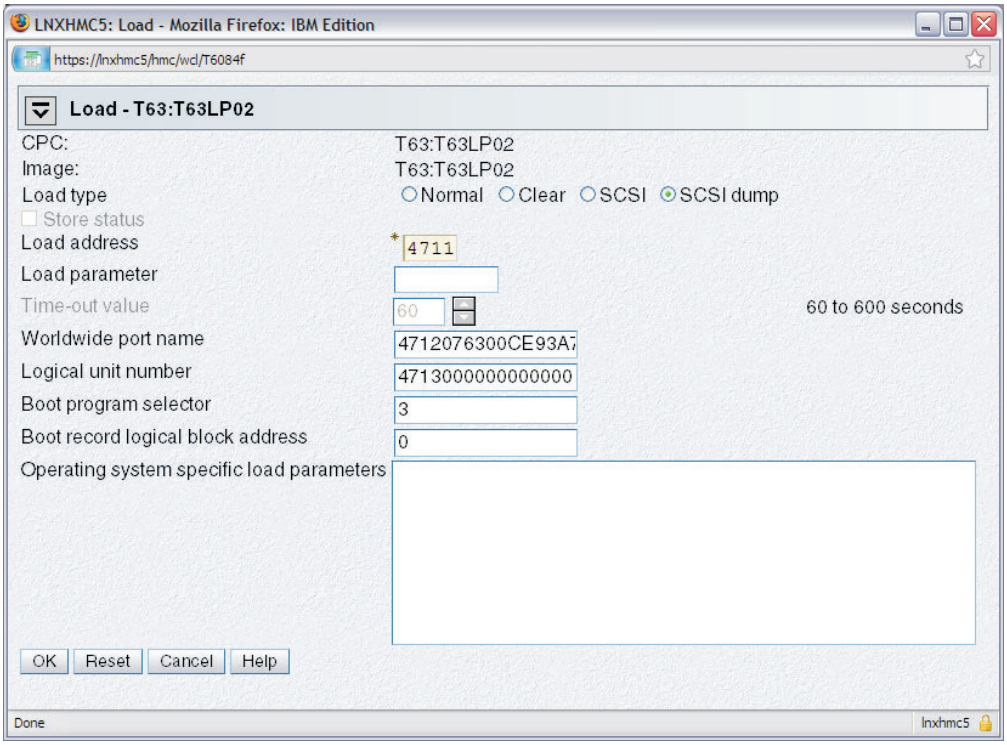


Figure 6. Load panel with enabled SCSI feature for dumping to SCSI disk

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, process the dump from the now running kdump kernel before the next IPL.

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*, SC33-8411. You can dump to a DASD device or a SCSI device.

About this task

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

```
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
```

Results

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.
See *Device Drivers, Features, and Commands*, SC33-8411 for more details about the magic sysrequest functions.

Results

The production system crashes. If kdump is set up correctly, the kdump kernel is booted and the dump can be accessed through `/proc/vmcore`.

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 a Linux system like this does not need to include the unused memory. You can use the **`zipl`** `size` option to limit the amount of memory that is dumped.

About this task

This section does not apply to kdump.

The **size** option is available for all **`zipl`** based dumps: DASD, tape, and SCSI, 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 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 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 prepares a DASD dump device for a dump that is limited to 100 megabyte:

```
# zipl -d /dev/dasdc1,100M
```

An equivalent section in a configuration file could look like this:

```
[dump1]
dumpto=/dev/dasdc1,100M
```

Appendix C. Command summary

The descriptions of the commands contain only the relevant options and parameters, for a full description refer to the man pages.

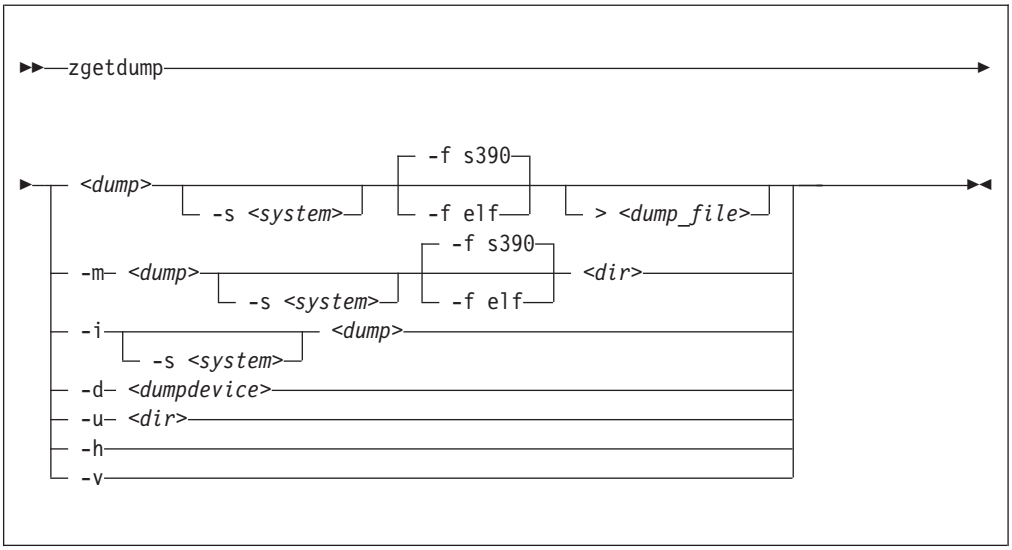
- The zgetdump tool
- The dumpconf service
- The crash tool
- The vmconvert tool
- "The vmur tool" on page 65

The zgetdump tool

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 to be loaded before you can use the `-m` option.

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 accessing 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 (64 bit only)

s390 S/390® dump (default)

-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_file:

```
# zgetdump /dev/dasdb1 > dump_file
```

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_file_on_target_host"
```

Using zgetdump to transfer a dump with FTP

Follow these steps to transfer a dump with FTP:

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 host issue a command of the following form:

   ```
   ftp> put |"zgetdump /dev/dasdb1" <dump_file_on_target_host>
   ```

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 multi_volume_dump_file:

```
# zgetdump /dev/dasdc > multi_volume_dump_file
```

For an example of the output from this command, see Chapter 4, “Using DASD devices for multi-volume dump,” on page 17.

Using zgetdump to copy a tape dump

Assuming that the tape device is /dev/ntibm0:

```
# zgetdump /dev/ntimb0 > dump_file
Format Info:
  Source: s390tape
  Target: s390

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/mem -f elf > dump.elf
```

For an example of the output from this command, see "Creating a kernel dump on a live system" on page 33.

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, 10 May 2010 17:26:46 +0200
  Dump ended.........: Mon, 10 May 2010 17:27:58 +0200
  Dump CPU ID........: ff00012320948000
  Build arch.........: s390x (64 bit)
  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.......: Thu, 15 Dec 2011 11:14:33 +0100
  Dump ended.........: Thu, 15 Dec 2011 11:14:46 +0100
  Dump CPU ID........: ff00012320978000
  UTS node name......: h4245049
  UTS kernel release.: 3.1.0
  UTS kernel version.: #216 SMP Wed Dec 14 10:38:19 CET 2011
  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 -f elf /dev/dasdx /dumps
# crash vmlinux /dumps/dump.elf
# 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. To do this, mount the dump and specify the required target dump format with the --fmt option.

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
```

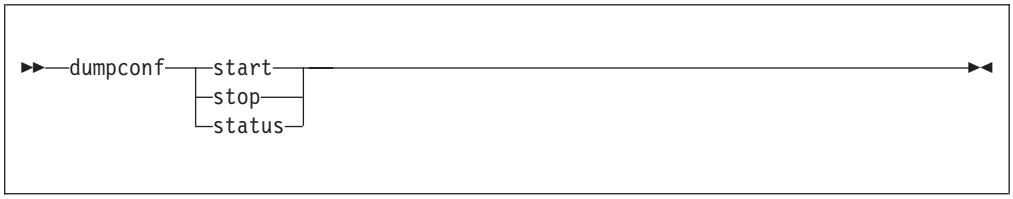
The dumpconf service

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

The command can be installed as service script under /etc/init.d/dumpconf or can be called manually. It reads the configuration file /etc/sysconfig/dumpconf.

Note: kdump does not depend on **dumpconf** and can neither be enabled nor disabled with **dumpconf**. If kdump has been 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.

dumpconf syntax



Parameters

start
Enable configuration defined in `/etc/sysconfig/dumpconf`.

stop
Disable **dumpconf**.

status
Show current configuration status of **dumpconf**.

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

-v or --version
Display version number on console and exit.

Keywords for the configuration file

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

dump Dump Linux and stop system.

reipl Reboot Linux.

dump_reipl
Dump Linux and reboot system. Note that dump_reipl is only available on LPAR with z9® machines and later, and on z/VM with version 5.3 and later.

vmcmd
Execute specified CP commands and stop system.

stop Stop Linux (default).

DELAY_MINUTES
The number of minutes that the activation of **dumpconf** 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. When the specified time has elapsed, **dumpconf** activates the specified panic action. This action is taken should the system subsequently crash. If the system crashes before the time has elapsed, the previously defined action is taken. If no previous action has been 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
Type of dump device. Possible values are `ccw` and `fcp`.

DEVICE
Device number of dump device.

WWPN
WWPN for SCSI dump device.

LUN LUN for SCSI dump device.

BOOTPROG
Boot program selector

BR_LBA
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 **dumpconf** to enable and disable the configuration.

- To enable the configuration:

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

- To display the status:

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

- To disable dump on panic or PSW restart:

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

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

```
# dumpconf status
type....: no dump device configured
on_panic: stop
```

If the dumpconf script is installed under `/etc/init.d`, **dumpconf** can be called with the service utility. For example, `service dumpconf start`.

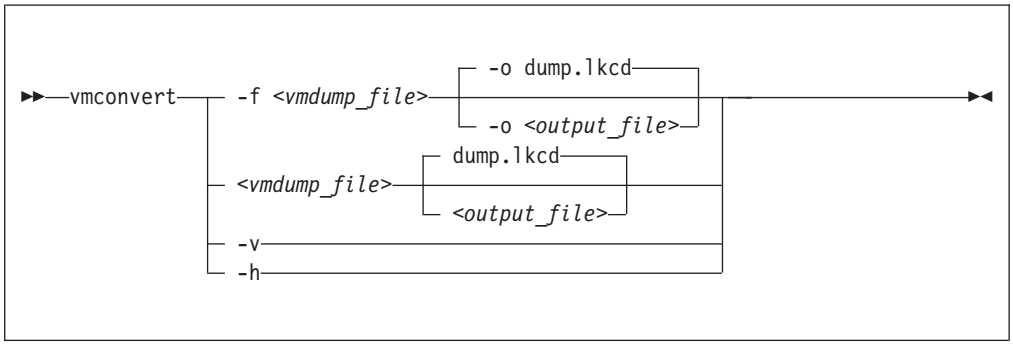
The crash tool

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

The vmconvert tool

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.
```

The vmur tool

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 data of the reader spool file.

For more details, see the **vmur** man page and *Device Drivers, Features, and Commands*, SC33-8411.

Example

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

```
# vmur rec -c 463 dump_file
```

Appendix D. 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 Reader Comment 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 3248
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, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks 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.

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

Linux on System z
Using the Dump Tools
Development stream (Kernel 3.5)

Publication No. SC33-8412-11

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
SC33-8412-11

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 3248
Schoenaicher Strasse 220
71032 Boeblingen
Germany

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

SC33-8412-11

Cut or Fold
Along Line

SC33-8412-11