z/VM

Running Guest Operating Systems

version 6 release 1

SC24-6228-00

z/VM

Running Guest Operating Systems

version 6 release 1

SC24-6228-00

Note!
Before using this information and the product it supports, read the information under “Notices” on page 127.

This edition applies to version 6, release 1, modification 0 of IBM z/VM (product number 5741-A07) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC24-6115-03.

Contents

About This Document

This book is intended to help you to plan for and to operate guest operating systems under z/VM®. It also includes sample execs to help you automate certain tasks.

Note: Unless otherwise specified, assume that any exec mentioned in this book is not available on the z/VM product tape.

This book provides no programming information for customers.

Intended Audience

This book is intended for anyone who has installed a stand-alone operating system and who now needs help planning for and running it as a guest operating system under z/VM.

This book is not a substitute for training or for having a basic understanding of z/VM. So, before using this book, you should:

- Be able to operate the guest system on a real machine
- Be familiar with the z/VM IPL procedure
- Understand the basic concepts and facilities of z/VM
- Be able to operate a z/VM terminal.

The material presented may not be directly applicable to all installations. You must determine the usefulness of the procedures and examples presented.

A Word About Commands

z/VM commands can be used for a variety of purposes from a variety of environments. You can process commands from execs, from program services such as CMSCALL and DIAGNOSE X'08', or from a terminal. Some parts of command output are more useful when handled by an exec or other program service. Other elements are more useful when appearing on the terminal for you to read.

When using commands, consider the following:

- Textual output from commands may change from release to release. In particular, the format of command output may change, based on display terminal considerations. In addition, the output may be available in several languages. It is IBM®'s intent to change textual output only when necessary.
- IBM intends to keep return codes compatible from release to release.

Where to Find More Information

For more information about z/VM functions, see the books listed in the “Bibliography” on page 133.

Links to Other Online Documents

If you are viewing the Adobe® Portable Document Format (PDF) version of this document, it might contain links to other documents. A link to another document is based on the name of the requested PDF file. The name of the PDF file for an IBM document is unique and identifies the edition. The links provided in this document are for the editions (PDF names) that were current when the PDF file for this document was generated. However, newer editions of some documents (with different PDF names) might exist. A link from this document to another document works only when both documents reside in the same directory.

How to Send Your Comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

Use one of the following methods to send us your comments:

1. Send an e-mail to mhvrcfs@us.ibm.com
2. Visit the z/VM reader's comments Web page at www.ibm.com/systems/z/os/zvm/zvmforms/webqs.html
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRCFS Reader Comments
   Department H6MA, Mail Station P181
   2455 South Road
   Poughkeepsie, NY 12601-5400
   U.S.A.
4. Fax the comments to us as follows:
   From the United States and Canada: 1+845+432-9405
   From all other countries: Your international access code +1+845+432-9405

Include the following information:

- Your name and address
- Your e-mail address
- Your telephone or fax number
- The publication title and order number:
  z/VM V6R1 Running Guest Operating Systems
  SC24-6228-00
- The topic and page number related to your comment
- The text of your comment

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 submit to IBM.

If You Have a Technical Problem

Do not use the feedback methods listed above. Instead, do one of the following:

- Contact your IBM service representative.
- Contact IBM technical support.
- Visit the z/VM support Web page at www.vm.ibm.com/service/
- Visit the IBM mainframes support Web page at www.ibm.com/systems/support/z/

Summary of Changes

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the changes. Some program updates might be provided through z/VM service by program temporary fixes (PTFs) for authorized program analysis reports (APARs), which also might be available for some prior releases.

SC24-6228-00, z/VM Version 6 Release 1

This edition supports the general availability of z/VM V6.1. Changes to information include:

- Specialty engine support. See “Specialty Engine Support” on page 11.
- Cryptographic support. See “z/VSE Hardware Crypto Support” on page 39 and “Crypto Definitions” on page 65.

Part 1. Overview of Guest Environments in z/VM

This part of the book develops the fundamental concepts of *guest support* in z/VM.

Chapter 1. The Fundamentals of Guest Support in z/VM

This chapter contains a general summary of those virtual machine concepts that have a particular bearing on running guest operating systems.

The Advantages of Guest Support in z/VM

You can use *guest support* in z/VM to develop, maintain, manage, and migrate *other operating systems* that make use of IBM Enterprise Systems Architecture/390 (ESA/390) or IBM z/Architecture®. In fact, these tasks constituted the original VM mission. When performing these tasks, system programmers and application programmers often solved the problems they encountered by using the solutions that were already integrated or implicit in VM. In time, VM developed a large collection of handy, useful tools. So much so, that today many IBM customers choose to run their systems in virtual machines as guests of z/VM. If you ask them why, they offer an impressive list of reasons:

System Simulation

Sooner or later, every computer user yearns for a spare system to try out an idea or to isolate a task. z/VM makes virtually limitless “spare systems” available in virtual machines, each simulating a real machine with very high fidelity. The advantages of making one system look like many need hardly be explained here, but in the present context it is worth mentioning at least three of them:

- You can create an exact replica of your production system on which you can test your new programs, services, and procedures. This is a relatively inexpensive way to have your own test system (or as many as you’d like). It is also a safe way to test new function, because your real production system, its applications, and data are protected from any damage the new function might cause if you tested it in the host.
- You can have a flexible migration system. This eliminates the usual inconvenience to users which migration usually causes.
- You can create a multiple production system environment. For example, your installation might run VSE applications, be migrating to z/OS®, and have a new vendor package that runs under Linux. As guests of z/VM, all three can run efficiently while sharing one processor.

System Efficiency

z/VM’s guest support helps you to run systems at lower cost but with greater efficiency. For example:

Performance benefits: Guest systems may see performance improvements by exploiting z/VM features. For example, both virtual disk in storage and minidisk cache allow guests to avoid real I/Os by using data in storage and caching techniques.

Reduced hardware cost: Guest systems can share devices such as channels, printers, and DASD, which z/VM efficiently manages. Indeed, z/VM adds new value to such devices merely by the way it manages them. A good example is z/VM’s minidisk support, which allows one real disk to function as if it were several smaller disks (such as multiple IPL-able minidisks).

And because z/VM simulates some devices (such as unit record devices and CTC adapters), your installation may be able to avoid buying real ones.

Operations management: With PROP (z/VM's PRogrammable OPerator) you can cut your console traffic substantially. That saves time and reduces errors.

Recovery management: You can build guest virtual machines to simulate systems at your organization's other sites. So, z/VM can become a disaster-recovery backup site.

Interactive Computing:

Although it is not normally considered a guest function, CMS is the development and interactive platform-of-choice in many guest environments, even where the guest may offer an alternative. One reason for this is that CMS offers many practical features, including:

Interactive program development tools: CMS, XEDIT, and REXX provide elegant, powerful, and convenient services to help you write programs.

Debug and trace tools: Debugging under CMS is much easier than in a batch environment, because you can see the results right away and make changes easily. With the advent of INSPECT and its truly interactive symbolic debugging, you can examine your code, executing one instruction at a time, making any necessary changes as you go along. These CMS-supported tools speed up the guest application development process, too.

Interactive data analysis and reduction: CMS Pipelines, REXX, XEDIT, and DB2® constitute a powerful toolbox for reducing and analyzing guest data, including data that was generated by a guest's applications.

Access to CMS applications: z/VM supports many applications, such as DB2 and CADAM, that are attractive to guests. Because applications run in virtual machines separate from the guests, you can add these applications without disrupting the whole system. In other words, no migration is necessary.

Because guest support is one of z/VM's most mature roles, enhancements to it tend to come incrementally and with little ostentation. That's why many users are unaware of z/VM's strengths in this area, such as:

DB2 guest sharing and improvements: With DB2, VM/VSE users can now share a DB2 database among several VSE guests. For most customers, consolidating several guest databases into one DB2 database reduces administrative work, simplifies operation, increases data integrity, and improves performance. DB2 guest sharing also streamlines access to operational data by decision support personnel, who often use CMS-based tools. Furthermore, sharing one DB2 database gives VSE applications access to remote data by way of Distributed Relational Database Architecture™ (DRDA®) support.

Multiple 3270 Session Support: VM/VSE users often do several things simultaneously on different operating systems: operators can manage consoles for multiple guest virtual machines, while programmers can move between CMS sessions and virtual test systems. Not surprisingly, users want to be able to run several sessions at each terminal simultaneously to simplify their work. z/VM Pass-Through Facility (PVM) supports several concurrent sessions per user with the ability to switch from session to session using a command or hot-key.

Real and Virtual Storage Support in z/VM

z/VM uses real storage to hold CP code, CP owned objects, and user space. It simulates, among other things, potentially large amounts of storage for the virtual machine's benefit. CP considers this simulated storage to be *second-level storage;* the virtual machine, however, regards this storage as its own *first-level* or "real" storage.

Note: For each virtual machine, CP creates dynamic address translation (DAT) tables in real storage to reference the virtual machine storage. The size and number of these tables will increase as the defined virtual machine storage size is increased. Therefore, if many virtual machines have very large storage sizes, or if many large virtual machines are very active, it could affect real storage availability. For more information, see *z/VM: CP Planning and Administration*.

An XC virtual machine can address up to 2047 MB of second level storage in the base address space plus multiple VM data spaces of up to 2047 MB each. An ESA or XA virtual machine can obtain more than 2047 MB of second level storage. However, the real processor might have a limit that cannot be exceeded. If a directory user definition or a DEFINE STORAGE, LOGON, or XAUTOLOG command exceeds the base address space storage size supported by the real processor, message HCP093E will result.

Only those portions of the virtual storage of the virtual machines currently needed are in that part of host real storage called the dynamic paging area (DPA). The remaining pages of the virtual storage of these virtual machines reside in auxiliary storage until needed. They are then brought into the DPA.

Virtual storage, whether in the DPA or in auxiliary storage, is called second-level storage because the addresses in second-level storage are relative. As such, they must be translated to host storage addresses (that is, first-level storage addresses) before the locations to which they refer can be accessed. Also, because they are brought in only when needed, contiguous locations in second-level storage may not necessarily be contiguous in host storage. Storage addresses in a virtual machine are always virtual.

For example, suppose that a virtual machine is executing a program that contains a branch to virtual address 1000. If virtual location 1000 is in host storage location 202000, address translation must cause that branch to be performed to execute the instruction at host storage location 202000.

It is important to remember that an operating system in a virtual machine may create virtual storage. This appears as *third-level* storage to CP. In fact, both z/OS and z/VM create virtual storage. If such an operating system were a guest of z/VM, then the virtual storage it created would be third-level storage from the point of view of the host z/VM system. On the other hand, whether VSE creates virtual storage depends on how it is generated.

The amount of third-level storage supported depends on the capabilities of the second-level operating system. A z/VM system running as a guest of a z/VM system could generate third-level storage for virtual machines.

Virtual processors can be dedicated to real processors, but you should be aware that doing so does not guarantee performance improvement for a guest. If real storage is not available to run the virtual machine, the virtual machine is delayed

until real storage becomes available. To reduce or eliminate the chance that a virtual machine is delayed, you can use the SET RESERVED or LOCK commands. But use these commands with caution, because using them for too many virtual machines can affect the overall performance of your z/VM system.

The minidisk cache feature of z/VM may provide a significant performance improvement to a virtual machine. The minidisk cache feature will use host real storage or expanded storage to cache data for virtual I/Os, thus avoiding real I/Os to DASD. z/VM has a fair share algorithm for inserting data into the cache. This prohibits any one virtual machine from having more than its fair share of data in the cache.

However, it may be desirable to give certain guests priority for minidisk cache. This can be accomplished by using the NOMDCFS setting on the OPTION directory control statement. The NOMDCFS setting stands for no minidisk cache fair share. A guest with this setting will not be limited as to the amount of data it can insert into the cache.

You can issue the DEFINE STORAGE command to configure reserved and standby storage before you IPL an operating system in the virtual machine. This allows the second-level operating system to exploit the storage reconfiguration capabilities of the Service Call (SERVC) instruction. For example, the operator of a second-level z/VM system can issue the SET STORAGE command to increase the amount of “real” storage (if additional storage is available).

Attention: z/VM supports any virtual configuration established through the DEFINE STORAGE command. In addition, it supports any “real” storage configuration set up by a SET STORAGE command issued by a previous instance of z/VM running in the virtual machine. However, if you choose to run z/VM at second level in the same virtual machine where a non-z/VM operating system was previously running, it is suggested that you log off or issue DEFINE STORAGE commands to trigger a SYSTEM RESET CLEAR (to reestablish the original storage configuration) before you IPL z/VM.

Device Considerations

CP will allow a guest considerable control over DASD operation. Caution should be exercised when deciding the scope of control a guest will be allowed. CP will allow the issuance of CCW’s, on a control unit with mixed ownership, that have a global affect. An example is the “Diagnostic Control CCW” available with 3390’s. In this case a guest will be allowed to use the CCW to do operations such as a “Diagnostic Initialize Subsystem”. This can reset the entire cache for a control unit. If there is a mixture of CP owned DASD and DASD dedicated to a guest the results can be unpredictable. Virtual machines may check stop or CP abend because pages critical to system operation have been purged.

The scope of control can be limited through the use of directory options and CP commands. The directory options are DASDOPT, and MAINTCCW on the OPTION statement. The CP commands that can affect the overall operation of a string of DASD are :

- SET CACHEFW
- SET CACHE
- SET DASDFW
- SET NVS
- ATTACH

For a more detailed explanation of the commands and options mentioned above see *z/VM: CP Planning and Administration* and the *z/VM: CP Commands and Utilities Reference*.

Time of Day Clock Considerations

By default, the guest will see the real Time-of-Day (TOD) clock value of the real machine. Specify OPTION TODENABLE in the directory entry for guests that need to use the SCK instruction to set the virtual machine's TOD clock. OPTION TODENABLE is not necessary to specify a time zone offset in a guest virtual machine.

Virtual Machine Architecture

Your virtual machine's user directory entry specifies the architecture of your virtual machine: ESA, XA, or XC. To IPL any operating system in your virtual machine, your virtual machine must simulate the appropriate architecture.

- ESA/390 operating systems run in ESA virtual machines that are in ESA/390 mode (the default mode).
- z/Architecture operating systems (for example, z/OS or z/VM) run in ESA virtual machines that have been switched into z/Architecture mode through an instruction issued by the operating system.
- The XA virtual machine designation has been retained for compatibility; an XA virtual machine is functionally equivalent to an ESA virtual machine.
- XC is a virtual machine architecture only.
- CMS is supported in XA or XC virtual machines.

Note: Because many operating systems interrogate the configuration when they IPL, they do not always recognize resources created after they IPL. Therefore, all of a guest operating system's resources should be defined before it IPLs. This includes all processors, expanded storage, cryptos, and so forth.

To determine the architecture mode of your virtual machine, enter:

```
query set
```

Part of CP's response is `MACHINE ESA`, `MACHINE XA`, or `MACHINE XC`.

Multiprocessing Considerations

With z/VM you can run virtual machines in certain multiprocessing environments. A real multiprocessing environment is created using an *n*-way processor complex: a two-way processor complex consists of two processors sharing a common real storage; a three-way processor complex consists of three processors sharing a common real storage; and so on.

To set up a virtual multiprocessing environment, your installation does not have to have a multiprocessor complex. You can do virtual multiprocessing even if your real processor is a uniprocessor. With a uniprocessor, however, you cannot dedicate a real processor to a virtual machine.

Notes:

1. Defining more virtual multiprocessors than real processors results in poor guest performance. This practice is recommended only for testing.

2. Queries of the I/O configuration must be entered from the processor that has I/O capability.

Serialization

In order to ensure the correct operation of DB2 Server for VSE & VM in a real multiprocessing environment, DB2 Server for VSE & VM employs several serialization techniques.

Serialization of data is required to:

- Prevent more than one processor from simultaneously running CP code that cannot be reentered
- Ensure serial processing of shared, serially reusable functions
- Ensure serial operation of functions that cannot operate correctly in parallel.

The serialization techniques used by DB2 Server for VSE & VM include locking, forced execution on a particular processor, virtual-machine-definition-block dispatch lock, and execution of a unique task.

Defining Virtual Processors

This section describes the steps necessary to define virtual processors in virtual machines.

To define virtual processors in a virtual machine (VM1), you must first specify the maximum number of allowable virtual processors in the MACHINE directory statement:

```
MACHINE ESA 2
```

specifies that VM1 is an ESA virtual machine that can define up to two virtual processors. To define more than the initial processor, VM1 must enter the DEFINE CPU command (or have the appropriate CPU statement coded in its directory entry). For example, VM1 can enter:

```
define cpu 02
```

The maximum number of virtual processors you can define with a CPU statement is determined by the number you specify on the MACHINE directory statement. (If you do not include a MACHINE statement, the maximum number of virtual processors you can define is determined by the number of CPU statements in the directory entry of the virtual machine).

Assume that a virtual machine (VM2) wishes to define an additional virtual processor. (When VM2 is logged on, it receives one virtual processor at virtual processor address 00.) To specify the maximum number of virtual processors VM2 can define, code VM2's MACHINE directory statement as:

```
MACHINE XA 2
```

Now VM2 can add a virtual processor to its configuration with the DEFINE CPU command. If VM2 enters:

```
define cpu 02
```

CP defines a virtual processor at VM2's virtual processor address 02.

Allocating Real Processors

This section illustrates the different multiprocessing environments that are available with z/VM based on the type of processor you have.

Uniprocessor

When you run z/VM on a uniprocessor, all virtual machines as well as CP contend for one processor. You can use the SET SHARE command to adjust the share of processor time made available to each virtual machine.

n-Way Processor

You can set up any of several multiprocessing environments on an *n*-way processor, with any of the following combinations:

1. A virtual machine running on two or more processors
2. All dedicated processors removed
3. A mix of dedicated and nondedicated processors.

The following examples use a 4-way processor to show these *n*-way environments.

Figure 1 shows the first type of 4-way environment. With the DEDICATE CPU command, you can dedicate up to three processors to one virtual machine to optimize its performance.

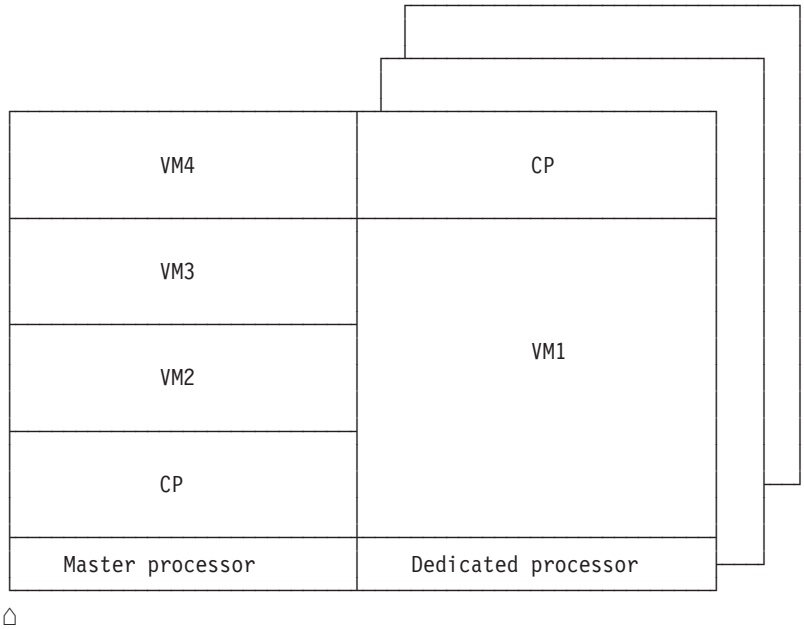

Figure 1. z/VM on a 4-Way Processor with Three Processors Dedicated to One Virtual Machine. Up to three processors are dedicated to a virtual machine to optimize performance.

The second type of 4-way environment (Figure 2 on page 10) contains no dedicated processors. All virtual machines receive about the same amount of processor time. Use the SET SHARE command to adjust the processor share of each virtual machine.

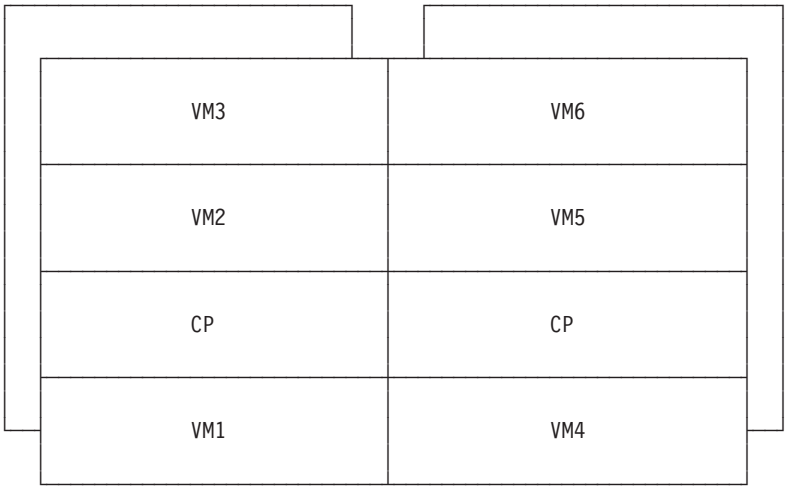

Figure 2. z/VM on a 4-Way Processor with No Dedicated Processors

In a third type of 4-way environment (Figure 3), two processors are dedicated to individual virtual machines and two processors are not dedicated. Just as in the other environments, you can adjust the share of processor time the virtual machine receives using the SET SHARE command.

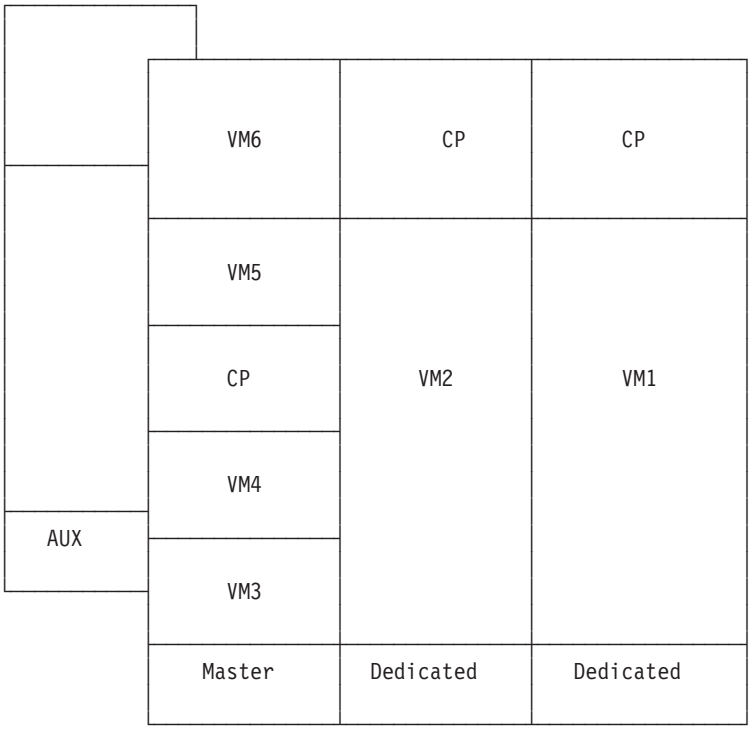

Figure 3. z/VM on a 4-Way Processor with Mixed Dedicated and Nondedicated Processors

On larger *n*-way processors, you can dedicate the other processors to virtual machines, or you can let CP use the extra processors, or any other combination to suit your installation's needs.

Specialty Engine Support

z/VM provides guest support for IBM mainframe specialty engines, including support for:

- The IBM System z® Application Assist Processor (zAAP) with its specialized Java™ execution environment
- The IBM System z Integrated Information Processor (zIIP) with its data-serving capabilities.

Guest support is also provided for:

- The IBM Integrated Facility for Linux (IFL)
- The Internal Coupling Facility (ICF).

There are two variations for this z/VM guest support:

- Simulation support, where z/VM simulates specialty engines for a guest virtual machine but dispatches them on real central processors (CPs).
- Virtualization support, where z/VM simulates specialty engines for guest virtual machines and dispatches them on the real specialty engines that match their type, if the specialty engines are available in the processor configuration.

Virtual specialty engines are defined for the virtual machine using the CP DEFINE CPU command and specifying the TYPE operand.

The following terms are used to explain the rules for defining virtual specialty engines:

- **Primary Real CPU Type** - the type of IPL processor
- **Primary Virtual CPU Type** - the type of the base virtual CPU (created automatically at logon time but can be redefined later)
- **Secondary CPU Type** - any type other than the primary CPU type (applies to real and virtual CPUs).

The term “real” as it applies to processors and CPUs refers to the processor units that are available to z/VM. These processor units are actually logical CPUs, since all mainframe machines that support a mixture of processor types are run in LPAR mode.

The types of virtual CPUs that can be defined depend on the hardware configuration, the logical CPU configuration, the logical partition mode, and the virtual configuration mode.

The hardware configuration might limit the CPU types allowed because only the specialty processors that are supported on the real machine can be defined as virtual specialty processors.

The logical partition mode indicates which types of logical CPUs can be allocated to the logical partition.

- A Linux-Only mode logical partition can be allocated either CPs or IFLs.
- An ESA/390 mode logical partition can be allocated one or more CPs with a combination of zIIPs and zAAPs.
- A z/VM mode logical partition can be allocated one or more CPs with a combination of zIIPs, zAPPs, IFLs, and ICFs.

- ESA/390 TPF and Coupling facility are other modes that appear on the selection screen when customizing logical partitions, but these modes are not supported for running z/VM.

To see the logical partition mode, use the CP QUERY PROCESSORS EXPANDED command.

The purpose of the z/VM mode logical partition is to allow a single z/VM image to support a mixed workload that includes:

- Linux on System z on IFL and general purpose CP engines
- z/OS on general purpose CP engines with zIIPs and/or zAAPs, if desired
- Coupling Facility virtual machines in support of Guest Sysplex on ICFs or general purpose CP engines
- z/VM program products (other than IPLA programs) on general purpose engines.

A virtual configuration mode setting supports specialty engines in a z/VM mode logical partition. This setting is similar to the logical partition mode setting and indicates the type of workload planned for the guest virtual machine. Each virtual configuration mode setting determines the set of virtual CPU types that are allowed to be defined. The virtual configuration mode is set by z/VM when a user logs on, but can be changed with the SET VCONFIG MODE command.

The valid virtual configuration mode settings are: **ESA390**, **LINUX**, **VM**, and **CF**. The **CF** mode setting is a special case. Users cannot set this mode through the SET VCONFIG MODE command; the setting is done automatically for CFVM users when they are autologged.

Use the CP QUERY VCONFIG command to display the virtual configuration mode for the virtual machine.

The default settings for virtual configuration mode are:

Table 1. Virtual Configuration (VCONFIG) Defaults at Logon

Logical Partition Mode	Primary Real CPU Type	VCONFIG MODE Default	Notes
ESA/390	CP	ESA390	except for CFVM
Linux only	CP or IFL	Linux	except for CFVM
z/VM	CP	ESA390	except for CFVM
all	CP or ICF	CF	CFVM

Valid virtual configurations are:

Table 2. ESA/390 Logical Partition

Primary Real CPU Type	VCONFIG mode	Primary Virtual CPU Type	Valid Virtual CPU Types
CP	ESA/390	CP	CP, zIIP, zAAP **
CP	CF *	CP	CP
CP	LINUX	IFL	IFL
CP	LINUX	CP	CP
CP	VM	CP	CP, zIIP, zAAP, IFL, ICF **

* Indicates OPTION CFVM was specified in the user directory entry.

** Indicates that zIIPs and zAAPs are supported as secondary virtual CPU types as long as they are supported on the underlying machine model.

Table 3. LINUX-only Logical Partition

Primary Real CPU Type	VCONFIG mode	Primary Virtual CPU Type	Valid Virtual CPU Types
CP	LINUX	CP	CP
CP	LINUX	IFL	IFL
IFL	LINUX	IFL	IFL

Table 4. z/VM Logical Partition

Primary Real CPU Type	VCONFIG mode	Primary Virtual CPU Type	Valid Virtual CPU Types
CP	ESA/390	CP	CP, zIIP, zAAP
CP	CF *	ICF	ICF
CP	LINUX	IFL	IFL
CP	LINUX	CP	CP
CP	VM	CP	CP, zIIP, zAAP, IFL, ICF

* Indicates OPTION CFVM was specified in the user directory entry.

The TYPE option of the CP DEFINE CPU command allows a user to define virtual processors of a type other than the type of the primary virtual CPU. These other types can be IFL, zAAP, zIIP, or ICF.

The CP SET CPUAFFINITY command is used to indicate whether simulation or virtualization support is desired for secondary processors. CPUAFFINITY is set on by default for all virtual machines. The CPUAFFINITY setting applies to all virtual CPUs in the virtual configuration. When CPUAFFINITY is set on for a user, secondary virtual CPUs will be dispatched on real processors of matching type if available. When CPUAFFINITY is set off for a user, secondary virtual CPUs will be simulated and dispatched on real processors of the primary real type.

In addition to setting CPUAFFINITY on or off, there is also a “suppressed” state that applies to individual virtual CPUs. Although “suppressed” is not an option on the CP SET CPUAFFINITY command, the CPUAFFINITY setting for a virtual CPU is

placed in the suppressed state when CPUAFFINITY has been set on for the user but no real CPU of matching type exists in the hardware configuration. When a virtual CPU is in the suppressed state, it will be simulated and dispatched on a real processor of the primary real type.

The suppressed state has a different meaning for primary virtual CPUs and secondary virtual CPUs. When a secondary virtual CPU is in the suppressed state, it is possible that the state might change to not-suppressed if a real CPU of the corresponding type becomes available. This situation is true for zIIPs or zAAPs in an ESA/390 mode logical partition, and for zIIPs, zAAPs, IFLs and ICFs in a z/VM mode logical partition. This situation is not true for virtual IFLs or virtual CPs in a Linux-Only logical partition for these reasons:

- When the primary real type is CP and CPUAFFINITY is set on, virtual IFLs will always be in the suppressed state because there can be no real IFLs in the hardware configuration on which to dispatch them.
- When the primary real type is IFL and CPUAFFINITY is set on, virtual IFLs will never be in the suppressed state because there always will be a real IFL on which to dispatch them.
- Virtual CPs will never be in a suppressed state because there are always real CPs when virtual CPs are defined.

The CP QUERY CPUAFFINITY command can be used to determine the CPUAFFINITY setting for the virtual machine. The CP QUERY VIRTUAL CPUS command can be issued to determine the state of CPUAFFINITY for individual virtual CPUs. When a virtual CPU is in the suppressed state, the output will indicate “CPUAFF SUPP” for that CPU. When a virtual CPU is in the not-suppressed state, the output will indicate “CPUAFF ON” for that CPU. When CPUAFFINITY is set OFF for the virtual machine, the output will indicate “CPUAFF OFF” for all virtual CPUs.

The CP INDICATE LOAD command displays the types of the processors in the logical partition. The CP INDICATE USER command displays the types of the virtual CPUs for a user and the EXPANDED option of this command displays the virtual configuration mode.

The CP SET SHARE command may be used to set a user’s dispatch share according to processor type. There are several consequences that must be considered when setting the SHARE of a userid with virtual specialty engines defined. For a detailed discussion of the effects of a virtual MP configuration on a virtual machine's share of CPU resources, refer to the SET SHARE command in *z/VM: CP Commands and Utilities Reference*.

Setting up Guest Specialty Engines

To define a virtual configuration that uses virtual zIIPs and zAAPs:

1. If you want virtualization support, ensure that a real zIIP or zAAP is available in the processor configuration. From a user with privilege class A, B, C, or E, issue CP QUERY PROCESSORS EXPANDED to view the types of real processors available and the logical partition mode. For example:

```
query processors expanded
PROCESSOR 00 MASTER CP
PROCESSOR 01 ALTERNATE CP
PROCESSOR 02 ALTERNATE CP
```

```
PROCESSOR 03 ALTERNATE ZIIP
PROCESSOR 04 ALTERNATE ZAAP
PROCESSOR 05 ALTERNATE ZAAP
PARTITION MODE ESA/390
```

The logical partition mode must be either ESA/390 or z/VM.

2. Log on the z/VM guest.

   The virtual configuration mode defaults to ESA390. Check this by issuing:

   ```
   query vconfig
   MODE = ESA390
   ```

3. Use the CP DEFINE CPU command to define the virtual specialty engines:

   ```
   define cpu 2 type ziip
   define cpu 5 type zaap
   define cpu 6 type zaap
   define cpu 8 type cp
   ```

4. Issue CP QUERY VIRTUAL CPUS to view the virtual CPUs that are defined and the CPUAFFINITY status. For example:

   ```
   query cpus
   CPU 00  ID  FF319B9E20948000 (BASE) STOPPED CP   CPUAFF ON
   CPU 02  ID  FF319B9E20948000 STOPPED ZIIP CPUAFF ON
   CPU 05  ID  FF319B9E20948000 STOPPED ZAAP CPUAFF ON
   CPU 06  ID  FF319B9E20948000 STOPPED ZAAP CPUAFF ON
   CPU 08  ID  FF319B9E20948000 STOPPED CP CPUAFF ON
   ```

5. IPL z/OS or other guest operating system that will make use of the zIIPs and/or zAAPs.
6. If you want to switch between simulation and virtualization, issue CP SET CPUAFFINITY from a user with privilege class A. For example:

   ```
   set cpuaffinity off for user test1
   set cpuaffinity on for user test1
   ```

To define a virtual configuration that uses virtual IFLs:

1. If you want virtualization support, ensure that a real IFL is available in the processor configuration. From a user with privilege class A, B, C, or E, issue CP QUERY PROCESSORS EXPANDED to view the types of real processors available and the logical partition mode. For example:

   ```
   query processors expanded
   PROCESSOR 00 MASTER CP
   PROCESSOR 01 ALTERNATE CP
   PROCESSOR 02 ALTERNATE ZIIP
   PROCESSOR 03 ALTERNATE ZAAP
   PROCESSOR 04 ALTERNATE IFL
   PROCESSOR 05 ALTERNATE ICF
   PARTITION MODE z/VM
   ```

2. Log on the z/VM guest. The virtual configuration mode must be LINUX. Issue QUERY VCONFIG to check the virtual configuration mode and SET VCONFIG MODE to change it, if necessary. For example:

   ```
   query vconfig
   MODE = ESA390

   set vconfig mode linux
   MODE = LINUX
   Storage cleared - system reset.
   ```

3. Issue the CP DEFINE CPU command to define the virtual IFLs. If the base CPU is not type IFL, it must be redefined to be type IFL. For example:

   ```
   q cpus
   CPU 00 ID  FF03233F20640000 (BASE) CP  CPUAFF ON
   Ready;
   ```

```
def cpu 0 type ifl
CPU 00 redefined as TYPE IFL
Storage cleared - system reset.

def cpu 3 type ifl
00: CPU 03 defined

00: def cpu 5 type ifl
00: CPU 05 defined
```

4. Issue CP QUERY VIRTUAL CPUS to view the virtual CPUs that are defined and the CPUAFFINITY status. For example:

```
00: q cpus
00: CPU 00 ID  FF03233F20640000 (BASE) STOPPED IFL  CPUAFF ON
00: CPU 03 ID  FF03233F20640000 STOPPED IFL  CPUAFF ON
00: CPU 05 ID  FF03233F20640000 STOPPED IFL  CPUAFF ON
```

5. Load (IPL) Linux to make use of the IFLs.

To define a virtual configuration for test purposes which simulates a z/VM mode logical partition:

1. If you want virtualization support, ensure that real specialty engines are available in the processor configuration. From a user with privilege class A, B, C, or E, issue CP QUERY PROCESSORS EXPANDED to view the types of real processors available and the logical partition mode. For example:

```
query processors expanded
PROCESSOR 00 MASTER CP
PROCESSOR 01 ALTERNATE CP
PROCESSOR 02 ALTERNATE ZIIP
PROCESSOR 03 ALTERNATE ZAAP
PROCESSOR 04 ALTERNATE IFL
PROCESSOR 05 ALTERNATE ICF
PARTITION MODE z/VM
```

2. Log on the z/VM guest. The virtual configuration mode must be VM. Issue QUERY VCONFIG to check the virtual configuration mode and issue SET VCONFIG MODE to change it, if necessary. For example:

```
query vconfig
MODE = ESA390

set vconfig mode vm
MODE = VM
Storage cleared - system reset.
```

3. Issue the CP DEFINE CPU command to define the desired virtual CPUs. For example:

```
00: def cpu 1 type zaap
00: CPU 01 defined
00: def cpu 2 type ifl
00: CPU 02 defined
00: def cpu 3 type icf
00: CPU 03 defined
00: def cpu 4 type ziip
00: CPU 03 defined
```

4. Issue CP QUERY VIRTUAL CPUS to view the virtual CPUs that are defined and the CPUAFFINITY status. For example:

```
00: q cpus
00: CPU 00  ID  FF046F5A20848000 (BASE) STOPPED CP   CPUAFF ON
00: CPU 01  ID  FF046F5A20848000 STOPPED ZAAP CPUAFF ON
00: CPU 02  ID  FF046F5A20848000 STOPPED IFL  CPUAFF ON
00: CPU 03  ID  FF046F5A20848000 STOPPED ICF  CPUAFF ON
00: CPU 04  ID  FF046F5A20848000 STOPPED ZIIP  CPUAFF ON
```

5. Load (IPL) z/VM from the guest virtual machine. You can then log on users of this z/VM image that can take advantage of the virtual specialty engines.

To define a guest coupling virtual configuration that uses virtual ICFs:

1. If you want virtualization support, ensure that a real ICF is available in the processor configuration. From a user with privilege class A, B, C, or E, issue CP QUERY PROCESSORS EXPANDED to view the types of real processors available and the logical partition mode. For example:

   ```
   query processors expanded
   PROCESSOR 00 MASTER CP
   PROCESSOR 01 ALTERNATE CP
   PROCESSOR 02 ALTERNATE ZIIP
   PROCESSOR 03 ALTERNATE ZAAP
   PROCESSOR 04 ALTERNATE IFL
   PROCESSOR 05 ALTERNATE ICF
   PARTITION MODE z/VM
   ```

   The logical partition mode must be z/VM in order to run a Coupling Facility virtual machine with virtual ICFs.
2. Autolog the CFVM guest. In a z/VM mode logical partition, the CPU type for all virtual CPUs defined for the CFVM virtual machine is changed to ICF automatically during autolog processing. This automatic change allows Coupling Facility workload to be dispatched on real ICFs, because CPUAFFINITY is set on by default for all users during logon.
3. If you want the Coupling Facility workload to be dispatched on real CP processors, issue CP SET CPUAFFINITY from a user with privilege class A. For example:

   ```
   set cpuaffinity off for user cfvm1
   ```

Using the Device Support Facilities from a Guest Virtual Machine

The Device Support Facilities (ICKDSF) is a program that supports disk volumes so that they can be accessed by other programs. For example, ICKDSF initializes disk volumes and manages track assignments.

ICKDSF operation and command support for z/VM users is determined by whether the operation is on a minidisk or on an attached device.

z/VM supports the following versions of ICKDSF:

The CMS Version of ICKDSF

The CMS version of ICKDSF provides support for all minidisks, including media maintenance and device initialization.

The Stand-Alone Version of ICKDSF

In the stand-alone version of ICKDSF, the support for minidisks is limited to the functions that are required to initialize a minidisk for operating system use. The support is provided by the MIMIC(MINI) parameter. All devices are treated as ATTACHed if MIMIC(MINI) is not specified. Media maintenance functions (INSPECT, for example) do not operate correctly on minidisks.

A z/VM Guest Using the MVS or VSE Version of ICKDSF

The MVS or VSE version of ICKDSF does not support minidisks. Because ICKDSF is unaware that it is running as a z/VM guest, all devices are treated as real

devices. If a minidisk is to be used by the operating system it should be initialized (using the INIT command) by the CMS or stand-alone version of ICKDSF before IPLing the guest operating system. Media maintenance cannot be performed on these minidisks. For details, see the *ICKDSF User's Guide*.

I/O Reconfiguration in z/VM

z/VM supports the System/390® hardware dynamic I/O configuration facility. This facility allows one to dynamically add, delete or modify the I/O configuration of the processor without requiring a power-on reset of the processor or IPL of z/VM.

z/VM's support allows the system administrator or system operator to issue privileged CP commands to dynamically alter the I/O configuration of the processor without the need of an IPL of z/VM or a power-on reset of the processor.

In addition to the CP commands to dynamically alter the I/O configuration of the machine, other I/O commands allow you to:

- Query the logical partitions on a machine (if the machine is in LPAR mode)
- Query the channel paths
- Query the status of channel paths to devices.

For more information on z/VM's support of dynamic I/O configuration, please see the *z/VM: I/O Configuration*.

Terminating a Guest

A guest's operation can be terminated abruptly by logging it off. This is equivalent to turning off its power. z/VM also allows you to shut down a guest by sending it a shutdown signal and giving it time to respond to that signal. A signal can be sent to a guest using the SIGNAL command.

Only some guest operating systems have programming support to enable them to receive and react to shutdown signals. For example, z/VM itself has this support. The QUERY SIGNALS command can be used to determine whether a guest is enabled to receive shutdown signals. For more information about terminating a guest, refer to the descriptions of the QUERY SIGNALS, SIGNAL, FORCE and SHUTDOWN commands in the *z/VM: CP Commands and Utilities Reference*.

Accounting

It is possible to track a guest's use of virtual network devices by enabling the creation of network account records.

To begin utilizing network account records, a system administrator needs to identify the users account records need to be created for. To activate the creation of account records for a particular user, NETAccounting needs to be added to that user's OPTION statement in his user directory. For example:

```
OPTION NETA
```

Note: Guest LANs that have accounting turned off will not produce accounting records even if the option NETA or option NETR are included in a users directory.

Additionally, if the system has a guest LAN and the administrator wishes to account for traffic to routers separately, NETRouter needs to be added to the router's

OPTION statement in the router's user directory. It is expected that the users who represent a path to a physical network would be designated as NETROUTERs. For example:

```
OPTION NETR
```

The creation of account records for a particular guest LAN may be controlled using CP commands and configuration statements.

The default LAN setting is to not create accounting records involving system LANs, and to not create accounting records involving user LANs.

The system default may be changed with the VMLAN statement in the system configuration file, or the SET VMLAN command. For example, the system configuration file might contain:

```
VMLAN ACCOUNTING SYSTEM ON USER OFF
```

The default accounting setting for individual LANs may be overridden at definition time by specifying ACCOUNTing ON|OFF on the DEFINE LAN system configuration statement, or by specifying the Class B option ACCOUNTing ON|OFF on the CP DEFINE LAN command.

The account setting for existing LANs may be changed by a Class B user by specifying ACCOUNTing ON|OFF on the CP SET LAN command.

Network transmission account records are collected in the same manner as other account records. For more information on setting up a service virtual machine, see *z/VM: CP Planning and Administration*.

Part 2. z/VSE under z/VM

This part of the book shows you how to plan for and how to operate z/VSE as a guest running under z/VM. It assumes that you have at least a working knowledge of both z/VM and z/VSE.

Throughout this document, “VSE” is shorthand for z/VSE.

Chapter 2. Planning to Run VSE under z/VM

This topic discusses how to plan for a VSE guest to run under z/VM.

VSE as a Guest of z/VM

A virtual machine provides an easy, convenient way to run guest operating systems. When you run VSE under z/VM, you get the functional equivalent of a real processor, main and auxiliary storage, and I/O devices. When you run a guest VSE system, VSE runs under the control program (CP) of the z/VM system.

Support Overview

Table 5 shows the VM support and system functions associated with the VSE Supervisor mode ESA.

Table 5. VSE Supervisor Mode, VM Support, and Major System Functions

VSE-MODE	**MODE=ESA**
Runs under PR/SM™ (LPAR mode)	YES
VM linkage enhancements	ALL, (see note 1)
Number of static partitions	12
Number of dynamic partitions	maximum 150-200 (NPARTS-12, SYS command)
Number of virtual address spaces (see note 1)	12 + number of dynamic partitions
VSIZE maximum (see note 1)	*nn*GB (see note 2)
SVA location	(see note 3)
Real storage maximum	*n*GB (see note 4)
Dynamic channel subsystem	YES
Max. number of devices (see note 1)	1024

Note:

1. This function may be dependent on the VSE release.
2. The theoretical limit is 90GB, but the practical limit is determined by the storage capacity of the DASD. The PDS has a maximum of 15 extents, and 15 extents on a 3390-3 would give 36GB. The VSIZE must be defined with the CP DEFINE STORAGE command before the VSE guest is IPLed.
3. ESA mode has a 24 bit SVA that is located low and a 31 bit SVA that is located high.
4. The theoretical limit is 2GB, but the practical limit is determined by the maximum storage size of the uniprocessor.

An Example

The following pages show an example of three VSE guests running under VM. Assume the following hardware:

- IBM 3390 DASD
- Separate system consoles are assumed for the VM system and each VSE guest.

Figure 4 on page 24 shows how the IBM 3390 disk devices can be accessed by the VSE guest systems of our example.

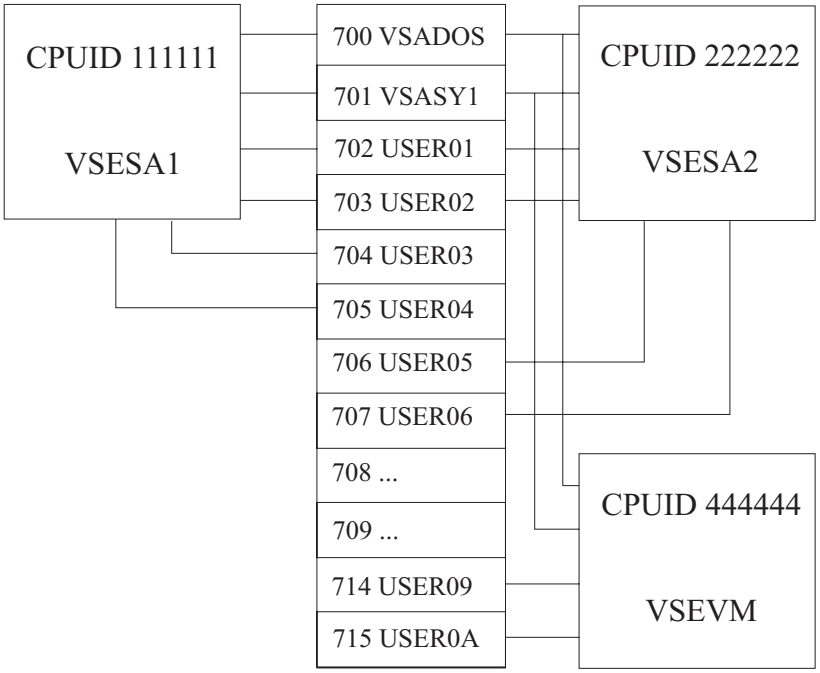

Figure 4. VSE Guest Systems Access to Disk Devices

Notes on Figure 4:

- 702 and 703 are full pack minidisks.
- 700, 701, 702, and 703 are shared among VSESA1 and VSESA2, 700 and 701 also by VSEVM.
- 700 and 701 are known to VM as VSADOS and VSASY1. On it reside the VSE system disks (DOSRES and SYSWK1) for VSESA1, VSESA2, and VSEVM.

On VSADOS and VSASY1, the VSE areas start on cylinder 1 (MDISK statements 240 and 241 for VSESA1, or LINK statements for VSESA2 and VSEVM). Refer to the directory entries shown on the following pages.

Using a Virtual Disk in Storage for Temporary Files

This example also uses the CP virtual disk in storage function to store temporary data, such as:

- Label information areas for the VSE guests
- The cross-system communication file (“lock file”) for the guests that are sharing data (VSESA1, VSESA2, and VSEVM).

A virtual disk in storage is a temporary simulation of an FBA minidisk in host storage instead of being allocated on a real DASD. Having a real FBA DASD in the system configuration is not required. The advantage of using a virtual disk in storage to store these temporary files instead of a permanent minidisk is that the I/O overhead of writing to a real device is eliminated.

This example sets up a server machine called VSESETUP as the owner of the virtual disk in storage. VSESETUP formats the virtual disk in storage and then autologs the VSE guests.

Preparing the Host z/VM System

Preparing the host z/VM system involves creating the directory entries for the server machine and the VSE guests. You may also need to make changes to the system configuration file or other system-dependent files.

The z/VM Directory Entries

Log on to your z/VM system and enter the directory entries for the VSE setup server machine and the VSE guest virtual machines as needed. Following are sample directory entries for the server and the three guest machines in our example.

Directory for VSESETUP Server Machine

```
USER VSESETUP password 32M 64M ABG
ACCOUNT acctno BIN1
IPL CMS
CONSOLE 009 3215
SPOOL 00C 3505 R
SPOOL 00D 3525 P
SPOOL 00E 1403
LINK MAINT 190 190 RR
LINK MAINT 19E 19E RR
MDISK 191 3390 1 1 UDISKA WR readpw writepw
MDISK 900 FB-512 V-DISK 6216 MWV readpw writepw multipw
```

Note: The virtual disk will be used by VSE as a D/T9336 and should be defined in multiples of 777 blocks.

Directory for VSESA1 Guest (z/VSE), MODE=ESA

```
USER VSESA1 password 32M 64M G
OPTION MAINTCCW LNKNOPAS CPUID 111111
MACHINE ESA
IPL 240
ACCOUNT ### SYSPROG
  DEDICATE 244 704
  DEDICATE 245 705
  CONSOLE 009 3215 T OPERATOR
  SPECIAL 080 3270
    "         "
    "         "
  SPECIAL 01F 3270
```

* See note 1 on page 26

```
  SPOOL   00C 3505 A
  SPOOL   00D 3525 A
  SPOOL   00E 4248 A
  SPOOL   02E 4248 A
  SPOOL   05D 3525 A
  SPOOL   05E 1403 A

       * Link to optional disks as needed....
       * Link to Executable CMS Code

  LINK MAINT 190 190 RR

       * Link to Program Products (Y Disk)

  LINK MAINT 19E 19E RR

       * Link to Virtual Disk in Storage

  LINK VSESETUP 900 900 MW
```

* See note 2

```
MDISK 220 3390 0 END VSADOS MW
MDISK 221 3390 0 END VSASY1 MW
MDISK 240 3390 1 END VSADOS MWV
MDISK 241 3390 1 END VSASY1 MWV
MDISK 242 3390 1 END USER01 MWV
MDISK 243 3390 1 END USER02 MWV
```

Notes:

1. Any appropriate display station can be used as system console. Just type

   ```
   dial vsesa1 01f
   ```

   and press ENTER twice.
2. When initializing the disks VSADOS and VSASY1, the Device Support Facilities (ICKDSF) create the VOLSER for z/VM. ICKDSF creates the VOLSER for the guest machine. VSADOS at 240 is the DOSRES device for VSE, and VSASY1 at 241 is the SYSWK1 device.

Directory for VSESA2 Guest (z/VSE), MODE=ESA

```
USER VSESA2 password 32M 64M G
OPTION MAINTCCW LNKNOPAS CPUID 222222
MACHINE ESA
IPL 240
ACCOUNT ### SYSPROG
  DEDICATE 244 706
  DEDICATE 245 707
  CONSOLE 009 3215 T OPERATOR
  SPECIAL 080 3270
    "         "
    "         "
  SPECIAL 01F 3270

        * See directory description for first guest (VSESA1)

  SPOOL   00C 3505 A
  SPOOL   00D 3525 A
  SPOOL   00E 4248 A
  SPOOL   02E 4248 A
  SPOOL   05D 3525 A
  SPOOL   05E 1403 A

        * Link to optional disks as needed....
        * Link to Executable CMS Code

  LINK MAINT 190 190 RR

        * Link to Program Products (Y Disk)

  LINK MAINT 19E 19E RR

        * Link to system disks

  LINK VSESA1 240 240 MW
  LINK VSESA1 241 241 MW
  LINK VSESA1 242 242 MW
  LINK VSESA1 243 243 MW

       * Link to Virtual Disk in Storage

  LINK VSESETUP 900 900 MW
```

Directory for VSEVM Guest (z/VSE), MODE=ESA

```
USER VSEVM password 32M 64M G
OPTION MAINTCCW LNKNOPAS CPUID 444444
MACHINE ESA

IPL CMS PARM AUTOLOG
ACCOUNT ### SYSPROG
  CONSOLE 009 3215 T OPERATOR
  DEDICATE 242 714
  DEDICATE 243 715
  SPECIAL 080 3270
    "         "
    "         "
  SPECIAL 01F 3270

        * See directory description for first guest (VSESA1)

  SPOOL   00C 3505 A
  SPOOL   00D 3525 A
  SPOOL   00E 4248 A
  SPOOL   02E 4248 A
  SPOOL   05D 3525 A
  SPOOL   05E 1403 A

        * Link to optional disks as needed....
        * Link to Executable CMS Code

  LINK MAINT 190 190 RR

        * Link to Program Products (Y Disk)

  LINK MAINT 19E 19E RR

        * Link to system disks

  MDISK 191 xxxx x END xxxxxx MR
  LINK VSESA1 240 240 MW
  LINK VSESA1 241 241 MW

       * Link to Virtual Disk in Storage

  LINK VSESETUP 900 900 MW
```

Note: The PROFILE EXEC for this guest contains the

```
IPL 240
```

statement. For details about using a PROFILE EXEC, see “Using EXEC Procedures” on page 35.

Description of Directory Control Statements

References are usually made to entries in the directory of the VSE guest machine named VSESA1.

For a complete list of z/VM user directory control statements, see *z/VM: CP Planning and Administration*.

The USER Statement: The USER statement defines a user ID and other virtual machine characteristics:

```
USER VSESA1 password 32M 64M G
```

VSESA1
Defines the user ID as VSESA1.

password
Can be changed to the password of your choice.

32M 64M
The 32M entry defines the virtual machine's storage size at log on. The 64M entry defines the maximum virtual machine storage size this user can request after logging on.

G Privilege class G general users control their own virtual machines. The VSE virtual machine is usually given class G privileges, which means all CP commands issued by the VSE operator are restricted to the operator's virtual machine environment.

The OPTION Statement: The OPTION statement specifies optional services available to the virtual machine. For example:

```
OPTION MAINTCCW LNKNOPAS CPUID 111111
```

MAINTCCW
The MAINTCCW operand lets a virtual machine initialize any DASD it uses.

LNKNOPAS
The LNKNOPAS operand specifies that this virtual machine can link to any other virtual machine's DASD without password authorization. This option takes effect only if no access control manager (like RACF®) is active.

CPUID nnnnnn
The CPUID *nnnnnn* operand specifies the processor identification number assigned to the VSE guest. This identification number must match the number specified on the CPU statement in the $ASIPROC. See "$ASIPROC Considerations" on page 32.

The MACHINE Statement: The MACHINE statement defines the virtual machine mode ESA.

The ACCOUNT Statement: The ACCOUNT control statement defines an account number and a distribution identification (SYSPROG). For example:

```
ACCOUNT ### SYSPROG
```

The account statement is optional. If omitted, both the account number and the distribution code default to the user ID. The ACCOUNT statement must follow the USER statement.

The IPL Statement: The IPL statement automatically IPLs a system (in our example a VSE system) by name or by device address. For example:

```
IPL 240
```

The sample directory for VSESA1 IPLs the VSE system. If you want to execute a PROFILE EXEC to set up the virtual environment, you need to IPL CMS before bringing up the VSE system.

The CONSOLE Statement: The CONSOLE statement specifies a virtual machine console. For example:

```
CONSOLE 009 3215 T OPERATOR
```

In the VSE environment, the way you define the VSE console depends upon whether:

- z/VM and VSE have separate consoles
- The VSE console supports the z/VM operations

- VSE is autologged
- VSE is logged on manually.

In our example, we assume that the VSE system is autologged by AUTOLOG1. The VSE operator can then use the DIAL command from any terminal to the VSE virtual machine specifying the VSE console address, in our example, 01F. Refer also to “Initialization Sequence for Example Environment” on page 33 for further details.

The console statement specifies that the OPERATOR is the secondary (VM) user ID receiving all CP messages for the VSE virtual machine when the primary (VSE) user ID is running disconnected.

The secondary user ID (OPERATOR) can send CP commands to the disconnected VSE machine. For example, to send an external interrupt command from the secondary user ID, enter:

```
send vsesa1 cp external 40
```

The DEDICATE Statement: The DEDICATE statement specifies that a real device is to be dedicated to the virtual machine. For example:

```
DEDICATE 244 704
```

Naturally, a real device can be dedicated to only one user at a time. Start interpretive execution (SIE) supports dedicated disks. These disks should appear in the directory before any SPOOL, LINK, MDISK, or SPECIAL statement to ensure that the virtual and real device numbers correspond. You can also dedicate a console (01F) to the VSE system. When the console is dedicated, the z/VM operator must handle all CP requests for the VSE virtual machine, but only as long as the VSE machine is in disconnected mode. The disadvantage of this is that you must have a second screen available in case VSE malfunctions.

The SPECIAL Statement: The SPECIAL statement defines a virtual device with device type and virtual address. For example:

```
SPECIAL 01F 3270
```

Terminal addresses defined with the SPECIAL statement do not have to be available on the system because they are not real addresses. With the SPECIAL statement in the directory, the DIAL command can be used to gain access to the guest machine and to take over the VSE operator console. SPECIAL entries can also be provided to give access from any terminal to VSE subsystems such as CICS/VSE®.

The SPOOL Statement: The SPOOL statement specifies the unit record device that is to be spooled. Several readers, punches, and printers may be specified, each on a separate SPOOL statement:

```
SPOOL   00D 3525 A
SPOOL   02E 4248 A
SPOOL   05E 1403 A
```

An entry in the directory is necessary for each unit record device that is used to spool output to VM or a VM user. You should have matching device type definitions for z/VM and for VSE in the ADD statements of the IPL procedure. If the definitions do not match, the VSE recorder file fills up quickly. VSE then displays the

```
RECORDER FILE FULL - RUN EREP
```

message, indicating that repetitive error handling with nonmatching devices has filled up the RECORDER FILE.

Note: For some devices, like 3262s, matching definitions are impossible.

READER

```
SPOOL 00C 3505 A
```

Other virtual reader devices require that you enter READY *cuu* under some circumstances. They also require that you spool the reader continuously.

PRINTER

```
SPOOL 00E 4248 A
```

You should start your POWER® print writers with the VM parameter unless you are using a dedicated printer.

For any print writer started with the VM parameter, POWER always sends the FCB as the first part of every print file. Therefore, it does not matter in what order the files are printed; the correct FCB is always used.

The LINK Statement: The LINK statement makes a device that belongs to another user ID available to this virtual machine at log on. If you want to make one volume available to several virtual machines:

- Define the volume for one of the virtual machines with an MDISK statement.
- Define a link to that volume, using the LINK statement for all other virtual machines that use the volume.

Later, if you must move or change that volume, you need only update the one MDISK statement; the LINK statements need not be updated.

For VSESA2, the link is to the DOSRES disk owned by VSESA1 with multiple write authorization:

```
LINK VSESA1 240 240 MW
```

All four guests are linked to the virtual disk in storage owned by the VSESETUP server machine:

```
LINK VSESETUP 900 900 MW
```

The MDISK Statement: The MDISK statement describes a permanent minidisk or virtual disk in storage that belongs to the virtual machine. A permanent minidisk is allocated from an area of a real DASD. A virtual disk in storage is a temporary simulation of a minidisk in host storage; it is not mapped to a real DASD.

The following MDISK statement defines a permanent minidisk (with a size of 1112 cylinders) which can be initialized as DOSRES (the *volume ID* of minidisk) on VSADOS. A similar example is shown using an FBA device with a size of 350,000 blocks.

```
MDISK 240 3390 1 1112 VSADOS MWV

MDISK 240 FB-512 1 350000 VSADOS MWV
```

CP does not check for overlapping extents in the MDISK statement. If you define overlapping cylinders among your minidisks, you are responsible for maintaining data integrity.

In addition to supporting minidisk extents on CKD and ECKD™ devices, z/VM also lets minidisks be defined on fixed-block architecture DASD (FBA). All virtual disks in storage are defined as FBA minidisks; however, having a real FBA DASD is not required.

DASD can be assigned to a virtual machine as an entire volume or as part of a volume. If you want to assign an entire volume, you can use either the DEDICATE or the MDISK statement. In deciding which statement to use, be aware that the DEDICATE statement lets only one user access the disk drive through that *cuu* address, whereas the MDISK statement lets the disk be shared among virtual machines. If you want to allocate part of a volume, use the MDISK statement. It is possible to allocate part of a z/VM volume to VSE and part of a VSE volume to z/VM. To do this, set up an MDISK statement for the part of the volume you want VSE to own. Then, initialize the VSE minidisk using ICKDSF.

Updating the Directory

If you made additions or changes to any directory entry, you must file the directory and start the DIRECTXA utility. The DIRECTXA utility processes the directory file to see if it follows the required format. To actually change or swap the current active z/VM directory, you must have write access to the system-owned (system residence or IPL device) volume that contains the current directory up to and including the directory cylinders or pages, or to the volume that is to contain the new directory.

Enter:

```
directxa filename
```

Note: Make sure that your z/VM virtual devices match the devices of the VSE system. Make sure that the devices you define in the z/VM directory entry for the VSE user ID match those in the procedure you use to IPL VSE.

Using Unsupported Devices

z/VM does not support every device that its many potential guests support. Nevertheless, if a guest supports a particular device, you can attach it to the guest even if z/VM does not support it. However, you cannot IPL your guest if its IPL volume resides on a DASD that z/VM does not support. For information about what devices are supported by z/VM, see *z/VM: General Information*.

Updating the System Configuration File

If your VSE system is using devices not supported by CP, you have to make changes to the system configuration file.

When you have unsupported devices, you must specify them as unsupported in the system configuration file and dedicate them to the VSE system in its DIRECTORY entry. In the system configuration file you might have:

```
RDEVICE raddr TYPE UNSUPPORTED DEVCLASS devclass
```

where *raddr* is the real address of the unsupported device. These devices must have matching entries in the IPL procedure. For unsupported device types you must specify a device subclass in the DEVCLASS operand. For a complete listing of the available subclasses, see *z/VM: CP Planning and Administration*.

Updating the User Directory File

Along with the changes in the system configuration file, the unsupported device should have a matching entry in the directory. In the sample directory, the entry for an unsupported device would be:

```
DEDICATE vaddr raddr
```

where *raddr* is the real address and *vaddr* is the virtual address. As always, the VSE virtual machine is also responsible for error recording.

Preparing the Guest VSE Virtual Machine

Manuals should be used to prepare your VSE system for running as a guest in a virtual machine. The supervisor names and IPL procedures used for the example are shown under "VSE IPL Procedures for Example Environment" on page 34.

Initializing VSE

There are three ways to initialize VSE:

1. *$ASIPROC (ASI master procedure).*

   The search for $ASIPROC.PROC in IJSYSRS.SYSLIB is always the first step performed by the IPL routine. The IPL routine searches for $ASIPROC. If $ASIPROC is found, the IPL routine looks for an entry that matches the CPUID. If a match occurs, the procedures named are run.

   Note: IBM recommends this method.
2. *$IPLESA and $$JCL which are the default IPL and JCL procedure names.*

   The IPL routine runs procedures with these names (if available on the system and a $ASIPROC is not found).
3. *Interactive IPL through prompts.*

   If neither default procedures nor $ASIPROC are found, the IPL routines prompt the operator for the appropriate IPL/JCL procedure names to be used.

$ASIPROC Considerations

An $ASIPROC lets you put the required control information into a procedure cataloged in IJSYSRS.SYSLIB and lets the system run the commands without operator intervention each time an IPL occurs. To indicate that a virtual machine is running under z/VM, the CPUID must have a prefix of FF. For example:

```
CPU=FF1111119121
```

where:

The first two digits are the prefix

The next six digits are the CPUID

The last four digits are the processor type.

To allow an IPL of VSE both natively and under z/VM, you can catalog an $ASIPROC with two entries for the same processor. The first entry would be for VSE running under z/VM and the second entry for VSE running stand-alone. The entry for the virtual machine must be the first entry within the master procedure as shown below:

```
CPU=022222229121,IPL=$IPLESA,JCL=$$JCL,LOG=YES
```

Note: If you want to interrupt the execution of the VSE $ASIPROC (say, to specify a different supervisor), then you have to use z/VM commands to simulate the actions you would take on the real hardware console if you were running VSE natively. If your VSE system recognizes specifying a LOADPARM on the hardware console, the LOADPARM option on the IPL command can be used:

```
IPL cuu LOADPARM ..P.....
```

If your normal procedure for interrupting the VSE IPL is to press the interrupt key on the real computer console, the EXTERNAL command is used:

```
IPL cuu STOP
EXTERNAL
```

When you specify a unique CPUID in the $ASIPROC, the same CPUID must be specified in the virtual machine directory entry or in a SET CPUID command before IPLing VSE.

Figure 5 shows the $ASIPROC for the VSE guest systems as defined under “Preparing the Host z/VM System” on page 25.

```
 CATALOG $ASIPROC.PROC    REPLACE=YES
CPU=FF1111119121,IPL=$IPLES1,JCL=$$JCL1,LOG=YES
CPU=FF2222229121,IPL=$IPLES2,JCL=$$JCL2,LOG=YES
CPU=FF4444449121,IPL=$IPLES4,JCL=$$JCL4,LOG=YES
/+
```

Figure 5. $ASIPROC for Four VSE Guest Machines

The names of the IPL and JCL procedures shown in the CPU statements are assigned by the user after VSE initial installation is complete. At that point, the names that the system used by default are still active. These are for processors 111111, 222222, and 444444, $IPLESA and $$JCL.

You can change the name of an IPL procedure with the *Tailor IPL Procedure* dialog. Skeletons SKJCL0, SKJCL1, and SKINITNN are provided for changing the JCL procedures.

Initialization Sequence for Example Environment

Here’s what happens during IPL and startup of our example:

1. The initialization process starts with VM log on and AUTOLOG1. The PROFILE EXEC for AUTOLOG1 includes a CP AUTOLOG statement for the VSESETUP server machine. See Figure 8 on page 42.
2. The PROFILE EXEC for VSESETUP contains a call to the ICKDSF program to format the virtual disk in storage. Following the call to ICKDSF, the profile for VSESETUP includes a CP AUTOLOG statement for each of the four VSE guest machines. See Figure 9 on page 43. The CP AUTOLOG statements enable the system to find the corresponding directory for each guest machine and perform IPL (IPL 240).
3. Enter a DIAL command from the display stations that are to become the VSE system consoles:

   ```
   dial vsesa1 01f
   dial vsesa2 01f
   dial vsevm  01f
   ```

   When you press ENTER twice, VSE becomes active and:

   a. Searches for the $ASIPROC. Refer to Figure 5 for the contents of $ASIPROC.
   b. Selects the IPL and JCL procedures as defined in the appropriate CPU statement (identified by its CPUID) of $ASIPROC.
   c. Performs VSE system startup in the same way as it would for a VSE native system by processing the IPL and JCL procedures selected. No operator intervention is required.

You can then log on to a VSE virtual machine in the same way as you would to a native VSE system using user ID and password.

VSE IPL Procedures for Example Environment

The following pages contain the IPL procedures for the VSE guest systems in our example.

IPL Procedure for VSESA1 Guest (z/VSE), MODE=ESA

```
01F $$A$SUPX,VSIZE=24M,VPOOL=64K,VIO=512K
ADD FEC,3505
ADD FFC,3505 ICCF DUMMY DEVICE DON'T DELETE
ADD FFA,3505 ICCF DUMMY DEVICE DON'T DELETE
ADD FED,2520B2
ADD FFD,2520B2 ICCF DUMMY DEVICE DON'T DELETE
ADD FEE,PRT1
ADD FEF,PRT1
ADD FFE,PRT1 ICCF DUMMY DEVICE DON'T DELETE
ADD FFF,CONS DEDICATED CONSOLE DON'T DELETE

        * The IPL procedure may contain other ADD
        * statements.  The IPL procedure may also
        * be able to sense which devices are
        * available.  Thus:

                    ADD 080,3270
                    ADD 01F,3270
                    ADD 00C,3505
                    ADD 00D,3525
                    ADD 00E,4248
                    ADD 240:245,ECKD,SHR  (3390 disk devices)
                    ADD 900,FBA,SHR  (virtual disk in storage)

DEF SYSCAT=DOSRES,SYSREC=SYSWK1
SYS JA=YES
SYS SPSIZE=1024K
SYS NPARTS=24
DLA VOLID=VDSK01,BLK=82,NBLK=320,DSF=N,NAME=AREA1
DLF VOLID=VDSK01
DPD VOLID=DOSRES,CYL=209,NCYL=12,DSF=N
DPD VOLID=SYSWK1,CYL=422,NCYL=12,DSF=N
SVA PSIZE=640K,SDL=300,GETVIS=768K
/+
```

IPL Procedure for VSESA2 Guest (z/VSE), MODE=ESA

```
01F $$A$SUPX,VSIZE=24M,VPOOL=64K,VIO=512K
ADD FEC,3505
ADD FFC,3505 ICCF DUMMY DEVICE DON'T DELETE
ADD FFA,3505 ICCF DUMMY DEVICE DON'T DELETE
ADD FED,2520B2
ADD FFD,2520B2 ICCF DUMMY DEVICE DON'T DELETE
ADD FEE,PRT1
ADD FEF,PRT1
ADD FFE,PRT1 ICCF DUMMY DEVICE DON'T DELETE
ADD FFF,CONS DEDICATED CONSOLE DON'T DELETE

        * Additional ADD statements are added by the system.
        * Refer to the VSESA1 IPL procedure for details.

DEF SYSCAT=DOSRES,SYSREC=SYSWK1
SYS JA=YES
SYS SPSIZE=1024K
SYS NPARTS=24
DLA VOLID=VDSK01,BLK=402,NBLK=320,DSF=N,NAME=AREA2
DLF VOLID=VDSK01
```

```
DPD VOLID=DOSRES,CYL=209,NCYL=12,DSF=N
DPD VOLID=SYSWK1,CYL=422,NCYL=12,DSF=N
SVA PSIZE=640K,SDL=300,GETVIS=768K
/+
```

IPL Procedure for VSEVM Guest (z/VSE), MODE=ESA

Note: This guest is autologged first and initializes the lock file.

```
01F $$A$SUPM,VPOOL=320K
ADD FEC,3505
ADD FFC,3505 ICCF DUMMY DEVICE DON'T DELETE
ADD FFA,3505 ICCF DUMMY DEVICE DON'T DELETE
ADD FED,2520B2
ADD FFD,2520B2 ICCF DUMMY DEVICE DON'T DELETE
ADD FEE,PRT1
ADD FEF,PRT1
ADD FFE,PRT1 ICCF DUMMY DEVICE DON'T DELETE
ADD FFF,CONS DEDICATED CONSOLE DON'T DELETE

        * Additional ADD statements are added by the system.
        * For example:
        *
        *   ADD 240:241,ECKD,SHR
        *   ADD 242:245,ECKD
        *   ADD 900,FBA,SHR
        *
        * Refer to the VSESA1 IPL procedure for details.

DEF SYSCAT=DOSRES,SYSREC=SYSWK1
SYS JA=YES
DLA VOLID=VDSK01,BLK=1042,NBLK=320,DSF=N,NAME=AREA4
DLF VOLID=VDSK01,BLK=2,NBLK=80,DSF=N,TYPE=N
SVA PSIZE=640K,SDL=300,GETVIS=768K
/+
```

Using EXEC Procedures

An exec procedure is a sequence of commands and other statements that can be processed by CMS. Although certain format and syntax rules apply, the process is the same as when you create any other type of CMS file.

When an exec is stored on a minidisk, you can call it simply by mentioning its filename. The PROFILE EXEC file also runs automatically when you IPL CMS.

For a VSE virtual machine, a PROFILE EXEC can be used to establish the proper SET commands appropriate to the machine's authority. Because the PROFILE EXEC runs automatically, you do not have to enter these commands.

Note: To avoid reserving routines in storage that are used only once, such as the initialization routine, do not issue the CP SET RESERVE command from a PROFILE EXEC.

Figure 6 on page 36 is an example of a simple PROFILE EXEC that can be used to set up some CP commands for a virtual machine:

```
/* VSE PROFILE EXEC */
'EXECIO 1 CP (VAR SYSNAME STRING QUERY USERID'
PARSE VAR SYSNAME USER . NODE .
'CP SET SHARE' USER 'ABS 50'
'CP SET QUICKDSP' USER 'ON'
```

Figure 6. PROFILE EXEC for VSE Guests

Stacking CP Commands in the PROFILE EXEC

If you want to automate the IPL of the VSE virtual machine after increasing the virtual storage or changing the virtual machine type, a special technique must be used. This is necessary because changing the virtual storage size or virtual machine mode causes the system to be reset. Once one of these commands is issued, the rest of the EXEC will never be executed. The following example stacks the CP commands so that they are issued one after the other.

```
/* Example of stacking CP commands in an EXEC */
nl = '15'
mach = 'CP SET MACH ESA'
stor = 'CP DEF STOR 64M'
term = 'CP TERM CONMODE 3270 BREAKIN MIN'
ipl = 'CP IPL 240 CLEAR'
cpcom = mach || nl || stor || nl || term || nl || ipl

address command cpcom
exit rc
```

Using the COMMAND Directory Control Statement

As an alternative to using the EXEC procedure described above, you can also use the COMMAND directory control statement to specify a command or a list of commands to be issued automatically whenever the VSE virtual machine is logged on. Note that the commands to be executed may be of any class, and that privileged commands can therefore be executed this way without having to give the virtual machine the associated privilege class.

See the COMMAND directory control statement in *z/VM: CP Planning and Administration* for more information.

Production and Testing

Virtual machines should be set up to avoid double paging and double CCW translation. This is accomplished using the NOPDS option.

Whether a virtual machine is used for production or testing, you should define its virtual storage as the size that the VSE system requires to run natively. For example, a VSE system that runs natively as a 16MB system should be defined as a 16MB virtual machine under z/VM.

Running VSE as a Virtual Machine

When you run VSE as a virtual machine, you specify a MODE=ESA supervisor. This supervisor:

- Supports up to 2GB of virtual storage
- Provides handshaking support available with a MODE=ESA supervisor
- Ensures that paging is done only once (by z/VM)
- Ensures that CCW translation is done only once (by z/VM)

- Supports page release, allowing a virtual machine to use DIAGNOSE X'10' to release pages of virtual storage.

  Note: When necessary, you can use the CP LOCK command to ensure that a subset of the VSE guest's pages remain in storage. You must remember, however, that when VSE subsequently releases one of those locked pages (DIAGNOSE X'10'), that page is unlocked.

Additional Considerations

1. z/VM lets VSE run on processors on which VSE cannot run natively. However, VSE is not able to use all of the processor's features. For example, some levels of VSE cannot use more than one processor. However, z/VM can use more than one processor on behalf of the VSE guest.
2. Because paging is handled differently under z/VM, programs that are dependent on paging and that run correctly natively may not run correctly in a guest machine.
3. When VSE runs under z/VM, you can use the VSE stand-alone dump program. You can also use the VMDUMP command to produce a virtual machine dump.

VSE Performance under z/VM

Understandably, installations that run VSE under VM as a production system (rather than as a test or conversion system) are concerned with performance. In this context, performance refers to how long a production run takes or how long before the system responds to an online request. The following factors affect the performance of a virtual machine running in a VSE guest:

- The amount of real storage available to the guest
- The amount of contention with other guests and CMS for resources such as channels, control units, and devices
- The frequency of real interrupts
- The frequency and type of I/O instructions processed
- The location of reference within virtual storage
- The amount of fixed-head paging space
- The location of the paging areas on DASD
- Whether the minidisk cache feature of z/VM is being exploited.

The performance of both the z/VM system and the individual virtual machines running under it can be measured and evaluated. How well the system responds is of paramount importance to the general user. How efficiently an individual virtual machine uses the storage, processor, and I/O facilities allotted to it is of great importance to the system analyst.

However, performance characteristics are difficult to predict when VSE is running under CP because of several complex factors. These factors can be broadly classified into three groups. These are detailed under:

- "Hardware Configuration Factors Influencing Performance" on page 38
- "Workload Factors Influencing Performance" on page 38
- "Analyzing Performance" on page 38.

Although a specific virtual machine's performance may not equal that of a real, stand-alone system, in some situations the total throughput obtained in the virtual machine environment is equal to or better than the throughput obtained on a real system.

Hardware Configuration Factors Influencing Performance

Certain hardware factors affect performance:

- The amount of real storage available
- The speed, capacity, and number of paging devices
- The amount of channel and control unit competition affecting each paging device
- The interference created between system paging devices and devices for processing a user's I/O requests.

How to Reduce Paging Activity

When a virtual machine refers to virtual storage addresses that are not in host storage, a page fault (and paging activity) occurs. Routines that have widely scattered storage references tend to increase the paging load caused by this virtual machine.

When possible, modules dependent on each other as well as the related reference tables, constants, and literals, should be located in the same 4KB page. Infrequently used routines such as those that handle unusual error conditions should not be placed near main routines. To minimize paging, reentrant coding techniques should be used whenever possible.

Workload Factors Influencing Performance

Many factors affect the performance of a VSE guest running under z/VM:

- The total number of virtual machines running under CP
- The type of work each virtual machine is doing, especially the amount of I/O processing required.

By measuring and evaluating the effects of these workload factors on a specific configuration, you can anticipate their effect on performance.

Analyzing Performance

When analyzing performance of VSE under z/VM, it is important to look at performance from the VM perspective and the VSE perspective. In order to be able to correctly interpret the data, it is crucial to understand how the data is affected by this environment.

From the VSE perspective, you may run the same tools to gather performance data as you would use if you were running natively. When anaylzing the data from these tools, it is important to understand what impact running as a guest has had on the data. For example, the data being gathered might be based on using the hardware interval timer for sampling. In this case, the samples only represent the time which VM considers to be guest's problem state time. CP time and pre-empted time will not be part of the sampled population.

Some tools may use the time-of-day-clock to record events. An example of this could be: the time of an I/O request is recorded correctly, the time of the SIO being issued is recorded correctly, but the time of the I/O interrupt is delayed by the time for CP to handle the real interrupt and reflect it to the guest. It could be further delayed if the guest has been pre-empted. The result could be that I/O counts are accurate, delays while the I/O requests are in the VSE queues by virtual device are accurate, but the device busy times are over stated.

A third possibility is that the CPU Timer might be used to measure CPU time given to a task. Since CP saves and restores the CPU Timer when a guest is pre-empted

and re-dispatched, and whenever the guest voluntarily relinquishes control by loading a wait state PSW, this data is as accurate running VSE native as running under VM.

From a VM perspective, you may run the same tools to gather performance data as you would use in a CMS environment. These tools are included in the Performance Toolkit for VM™. You can use the Performance Toolkit for VM for short-term study and problem-solving as well as long-term trend analysis and capacity planning. For more information, see *z/VM: Performance Toolkit Reference*.

How to Use CP to Measure Performance

After measuring the performance of both CP and the virtual machines it supports, the system analyst and general user can use CP commands to attempt to improve performance of either the system or a specific virtual machine.

z/VM provides certain CP commands (INDICATE and MONITOR, for example) to allow both CP's and the virtual machine's performance to be tracked and measured. Other commands (SET SHARE and SET QUICKDSP, for example) allow the setting of certain options to improve performance. For more information on using the INDICATE, MONITOR, and SET commands, refer to the *z/VM: CP Commands and Utilities Reference*.

Date and Time Zones in the VSE Virtual Machine

VSE SET ZONE= should be set to match the offset currently used by CP so that the VSE time of day will match the z/VM time of day.

If you do not use the VSE SET ZONE command, then VSE uses ZONE= WEST/00/00 and assumes that the hardware TOD clock is set to local time.

You can set the zone value on a guest VSE system by issuing the SET ZONE command any time before you enter the SVA command.

z/VSE Hardware Crypto Support

z/VSE provides hardware-accelerated encryption support by exploiting cryptographic features on System z processors. PCI-X cryptographic features are supported for RSA acceleration, and the CP Assist for Cryptographic Function (CPACF) provides symmetric algorithms such as Triple-DES, AES, or SHA. Cryptographic hardware is transparently used by TCP/IP for VSE/ESA and applications like Encryption Facility for z/VSE.

For implementation details, refer to the z/VSE documentation and the z/VSE home page at:

www.ibm.com/servers/eserver/zseries/zvse/

Using the Hardware Crypto Support with a z/VSE Guest under z/VM

To run z/VSE as a guest under z/VM, you need to define the crypto capability to the z/VSE system with the following statement in the z/VM directory:

```
CRYPTO APVIRT
```

This CRYPTO APVIRT statement provides access to the crypto hardware and allows you to use crypto-specific instructions. z/VM manages a pool of hardware

crypto queues which are shared among all of the APVIRT guests. Note that these guests may outnumber the actual number of hardware queues available.

You can use z/VM commands to query the crypto hardware and to check the currently assigned crypto domain and device number of your z/VSE guest, as follows:

- Query the status of the cryptographic units in the processor configuration and the status of the AP queues:

  ```
  QUERY CRYPTO
  QUERY CRYPTO APQS
  ```

- Check the currently assigned crypto domain and device number of your z/VSE guest:

  ```
  QUERY VIRTUAL CRYPTO
  ```

z/VM assigns the AP (Adjunct Processor) queue numbers randomly, so it is normal for the guest to see a different queue number each time the guest is started.

Collecting Hardware Crypto Status information on z/VSE

On the VSE side, information about hardware cryptographic support can be queried via the STATUS command of the security transaction server, which by default runs in partition FB. In the following example, the command queries the status of the VSE crypto device driver. It shows two available AP numbers 0 and 1. The crypto domain is 4.

```
msg fb,data=status=cr
AR 0015 1I40I  READY
FB 0011 BST223I CURRENT STATUS OF THE SECURITY TRANSACTION SERVER:
FB 0011 ADJUNCT PROCESSOR CRYPTO SUBTASK STATUS:
FB 0011   AP CRYPTO SUBTASK STARTED ......... : YES
FB 0011   MAX REQUEST QUEUE SIZE ............ : 1
FB 0011   MAX PENDING QUEUE SIZE ............ : 1
FB 0011   TOTAL NO. OF AP REQUESTS .......... : 86
FB 0011   NO. OF POSTED CALLERS ............. : 86
FB 0011   AP CRYPTO POLLING TIME (1/300 SEC).. : 1
FB 0011   AP CRYPTO WAIT ON BUSY (1/300 SEC).. : 75
FB 0011   AP CRYPTO RETRY COUNT ............. : 5
FB 0011   AP CRYPTO TRACE LEVEL ............. : 3
FB 0011   TOTAL NO. OF WAITS ON BUSY ........ : 0
FB 0011   CURRENT REQUEST QUEUE SIZE ........ : 0
FB 0011   CURRENT PENDING QUEUE SIZE ........ : 0
FB 0011   ASSIGNED APS : PCICC / PCICA ...... : 0 / 0
FB 0011                  CEX2C / CEX2A ...... : 1 / 1
FB 0011                  PCIXCC ............. : 0
FB 0011     AP 0 : CEX2A   - ONLINE
FB 0011     AP 1 : CEX2C   - ONLINE
FB 0011   ASSIGNED AP QUEUE (CRYPTO DOMAIN)... : 4
FB 0011 CPU CRYPTOGRAPHIC ASSIST FEATURE:
FB 0011   CPACF AVAILABLE ................... : YES
FB 0011   INSTALLED CPACF FUNCTIONS:
FB 0011     DES, TDES-128, TDES-192
FB 0011     AES-128
FB 0011     SHA-1, SHA-256
FB 0011 END OF CPACF STATUS
```

Chapter 3. Operating a VSE Guest under z/VM

This chapter discusses how to operate a VSE guest under z/VM.

Autologging the VSE Virtual Machine

AUTOLOG is a convenient way to start a large VSE guest running with many I/O devices under z/VM. The I/O devices needed by the VSE system require considerable contiguous free storage space for the I/O control blocks established by z/VM. If smaller users have logged on to z/VM before you IPL the large VSE guest, there may be insufficient contiguous free storage space available for the required I/O control blocks. As a result, there may be an insufficient number of I/O devices to accommodate the guest VSE system and its application programs.

To ensure sufficient contiguous free storage space for a large production VSE system, log on the VSE machine immediately after z/VM is loaded. To do this, either:

- Have the z/VM system operator enter the CP AUTOLOG command before enabling user terminals
- Define the AUTOLOG1 virtual machine in the z/VM directory. This is the method we have used in our initialization example. The AUTOLOG1 user ID is the default. You can change the AUTOLOG1 user ID using the STARTUP operand on the SYSTEM_USERIDS statement in the system configuration file.

Operator Issuing CP AUTOLOG Command

Before enabling any user terminals, the z/VM system operator can enter the CP AUTOLOG command for each production guest virtual machine that requires substantial contiguous free storage. The virtual machine being AUTOLOGed must have an automatic IPL defined in its directory. This machine is allowed to issue one read to its virtual console, and it operates in disconnected mode. The same limitations that apply to any disconnected machine apply here, too. To invoke the command, enter:

```
AUTOLOG userid password
```

For more on the AUTOLOG command and its alternative, the XAUTOLOG command, see *z/VM: CP Commands and Utilities Reference*.

Defining AUTOLOG1 in the Directory

To use AUTOLOG1 or any other user ID specified on the SYSTEM_USERIDS statement in the system configuration file to initiate one or more virtual machines, the z/VM directory statement loads CMS into the AUTOLOG1 virtual machine. In the PROFILE EXEC for AUTOLOG1, each CP AUTOLOG command can initiate one virtual machine. Each such virtual machine can contain a guest operating system, such as VSE. When you use the CP AUTOLOG command, the directory entry for the virtual machine referred by the CP AUTOLOG command must contain an IPL statement.

In our example, the PROFILE EXEC for AUTOLOG1 does not directly autolog the VSE guests. Because we are using a virtual disk in storage to store temporary data, we need a user who is not a VSE guest to own the virtual disk in storage. A virtual disk in storage simulates an FBA minidisk and cannot be formatted from VSE. Therefore, the PROFILE EXEC for AUTOLOG1 contains a CP AUTOLOG command for a server machine called VSESETUP, which owns the virtual disk in storage. The

PROFILE EXEC for VSESETUP calls the ICKDSF program to format the virtual disk in storage. Following the call to ICKDSF, the PROFILE EXEC for VSESETUP contains CP AUTOLOG commands for the four VSE guest virtual machines.

Figure 7 shows the AUTOLOG1 entry in the z/VM directory. The IPL statement initializes CMS, causing the execution of the PROFILE EXEC.

```
USER AUTOLOG1 PASSWORD 16M 16M ABG
  ACCOUNT ACCTNO BIN1
  IPL CMS
  CONSOLE 009 3215
  SPOOL 00C 3505 R
  SPOOL 00D 3525 P
  SPOOL 00E 1403
  LINK MAINT 190 190 RR
  LINK MAINT 19E 19E RR
  LINK MAINT 19D 19D RR
  MDISK 191 3390 1 1 UDISKA WR RPASS WPASS
```

Figure 7. AUTOLOG1 Entry in z/VM Directory

Figure 8 shows the PROFILE EXEC for AUTOLOG1 with the CP AUTOLOG command for the VSESETUP server machine. The last CP command in the PROFILE EXEC logs off AUTOLOG1.

```
/* PROFILE EXEC for AUTOLOG1 */

TRACE O
ADDRESS COMMAND
'CP SPOOL CONSOLE START'
'CP SET EMSG ON'
'CP AUTOLOG VSESETUP password'
'CP LOGOFF'
```

Figure 8. PROFILE EXEC for AUTOLOG1 with AUTOLOG Command for VSESETUP

Figure 9 on page 43 shows the PROFILE EXEC for VSESETUP, containing the call to ICKDSF and the CP AUTOLOG command for each VSE virtual machine to be loaded. VSEVM is autologged first because in our example it is the guest that initializes the lock file. The last CP command in the PROFILE EXEC logs off VSESETUP.

```
/* PROFILE EXEC for VSESETUP */

/* Sample exec to ICKDSF virtual disk in storage    */
/* and AUTOLOG VSE guests                           */
/* Assumes virtual disk in storage defined in       */
/* directory as device 900                          */
PUSH 'END'
PUSH 'U'
PUSH 'INIT UNIT(900),NOVERIFY,NOMAP,VOLID(VDSK01),FBAVTOC(start,length)'
PUSH 'CONSOLE'
PUSH 'CONSOLE'
'ICKDSF'
/* The following message will inform the operator that the guest */
/* operating systems are being autologged.                       */
'MSG OP The guest VSE virtual machines are being autologged.'
'ENABLE ALL'
'AUTOLOG VSEVM password' /* This is the guest that formats the lock file */
'SLEEP 30 SEC' /* Allows time to format lock file - adjust as necessary */
'AUTOLOG VSESA1 password'
'AUTOLOG VSESA2 password'
'AUTOLOG ......
'AUTOLOG ......
CP LOGOFF
EXIT
```

Figure 9. PROFILE EXEC for VSESETUP with AUTOLOG Commands

With the CP AUTOLOG command in the PROFILE EXEC, the VSE virtual machine is loaded and runs in disconnected mode. A VSE user can gain access by doing one of the following:

- Logging on with the user ID specified in the CP AUTOLOG command
- Issuing the CP SEND command from the secondary user's console
- Issuing the CP DIAL command and specifying the guest user ID.

When the user logs off, the contiguous storage space is relinquished, and the virtual machine stops. If you want to keep the virtual machine's I/O blocks in contiguous storage and keep the virtual machine running while you temporarily give up use of the virtual machine console, then enter the CP DISCONNECT command. Later, you can enter the CP LOGON command to reconnect the console to the virtual machine.

Issuing CP Commands from the VSE Virtual Machine

While running the VSE guest, use CP commands to:

- Communicate with the z/VM system operator or other virtual machine users
- Query the status of virtual machine devices or spool files
- Attach or detach devices from the virtual machine configuration.

As shown in the sample directories in Chapter 2, "Planning to Run VSE under z/VM," the VM console and the consoles of the VSE virtual machines are on separate devices. That means that only with commands prefixed with *CP (from a VSE virtual machine console) can you communicate with CP. The secondary user ID (OPERATOR) specified in the z/VM directory for VSESA1 is responsible for handling CP requests for the VSE virtual machine, or you can log on VSESA1 and enter CP commands on behalf of the VSE guest.

Note: CP commands cannot be entered with the *CP function during a VSE IPL, because the CCWs used for the console reads expect less data than the length of the *CP command entered. So, the additional data in the *CP command is truncated and lost.

Various Uses of the DEDICATE Statement

The DEDICATE control statement specifies that a real device is to be dedicated to a given user ID. Naturally, a real device can be dedicated to only one user at a time. For example, the following statement dedicates a disk device, as in the directory for the VSESA1 guest:

```
DEDICATE 244 704
```

Note: Apart from the DEDICATE statement, you can also use the CP ATTACH command to perform the following functions.

Tape Devices

A device such as a tape drive can be used by only one virtual machine at a time. You can dedicate the tape drive if only one virtual machine is to use it (for example, for journaling), or different users can use it in turns by issuing the CP ATTACH command.

To dedicate the tape drive to one user, specify it in that user's directory. For example:

```
DEDICATE 181 281
```

This statement allows the operating system to access the device at real address 281 by way of a virtual address of 181.

Unit Record Devices

Usually, spooling represents the most efficient way of handling the unit record input and output of many virtual machines. However, in certain cases, the dedication of a real unit record device to a single virtual machine may be justified.

One such special case is when the virtual machine's operating system does its own spooling, such as VSE/POWER under VSE. To eliminate double spooling of printer output, include a DEDICATE statement in the virtual machine's directory entry. For example:

```
DEDICATE 00E 002
```

This statement causes CP to pass all output for the virtual printer at address 00E to the real printer at address 002.

Important! If you dedicate unit record devices to a guest, you must not have a spool entry for that same device.

Unsupported Devices

You can use the DEDICATE statement to put a device that z/VM does not support into a virtual machine configuration. To dedicate such a device, it must be:

- Physically connected to the z/VM system
- Supported by the VSE operating system.

For example, a directory entry can include the statement:

```
DEDICATE 007 012
```

where real address 012 represents a real device that is unsupported by z/VM but is attached to the processor.

The device is added to VSE as:

```
ADD 007,xxxx,EML
```

Note: The EML parameter is required when defining a device for VSE that is defined to z/VM as unsupported.

Dedicated Terminal Definitions

To be more flexible, our example system does not use dedicated terminals for the system consoles. CICS/VSE applications are an example where dedicated terminals may be used.

Nondedicated Terminal Definitions

Include the SPECIAL statement in the system directory for each terminal that requires access to CMS and CICS/VSE. With the SPECIAL statement in the directory, you can use the CP DIAL command to gain access to the VSE system.

For example, if the z/VM directory entry for the VSE virtual machine has the following SPECIAL statement:

```
SPECIAL 080 3270
```

Then, enter either one of the following:

```
dial vsesa1 080
```

```
dial vsesa1
```

If you enter the CP DIAL command without a specified address, z/VM connects the terminal to the first available line as defined in the SPECIAL control statement. The line belongs to the specified user ID. If no lines are available or if all lines are busy, then z/VM issues an error message and does not make the connection. The end user remains connected until one of the following happens:

- The virtual machine logs off
- The virtual machine is forcibly logged off
- The terminal is turned off then turned back on
- The Normal/Test switch is actuated.

If the CP RESET command is issued from the VSE virtual console or from a user who is authorized to issue the CP RESET command, then the end user is disconnected. Once disconnected, the end user is free to use the DIAL command to connect to another user ID.

Spooling Options

Most multiprogramming operating systems have their own spooling subsystem. In VSE, this subsystem is VSE/POWER. Because z/VM also provides its own spooling, double spooling can occur. If VSE is a guest of z/VM, this raises the question, should an installation use only the VSE spooling subsystem or allow double spooling to occur?

Spooling Recommendations

If an installation has very much printing or punching to do, one of the spooling subsystems probably, though not necessarily, should be eliminated.

Use double spooling if:

- You have large quantities of printed output on standard forms
- DASD space is not a limiting factor
- You want several virtual machines to share a real printer without having to attach and detach the printer to the individual guests.

Many VSE virtual machines spool data and must use a common pool of unit record devices. In this case, double spooling reduces the privilege operations CP must simulate.

For z/VM to do the spooling for the VSE guest (double spooling), the spool statement must exist in the z/VM directory. This statement must agree with the ADD statement in the VSE ASI procedure.

Note: If VSE controls the spooling by either attaching or dedicating unit record devices, you must not have a corresponding SPOOL statement in the z/VM system directory defining the device.

VSE/POWER under z/VM

VSE/POWER behaves the same, whether in a stand-alone VSE system or in a VSE guest of z/VM. VSE/POWER makes no distinction between real or virtual devices, and it executes input and output regardless of the devices used.

Controlling Printed Output

Most of the VSE supported printers use a forms control buffer (FCB) to control the length of form skips. In addition, some printers can be equipped with the universal character set feature that is controlled by the universal character set buffer (UCB).

Loading Universal Character Set Buffer

You can load the UCB from either z/VM or VSE. If you load the UCB from VSE, the printer has to be attached to the VSE machine. The UCB automatically loads at IPL time, or the LUCB command loads the buffer after VSE has been IPLed. For example, the command:

```
lucb 00e,$$bucb00,fold
```

loads the UCB $$BUCB00 on the printer at 00E, and FOLD translates lower case to upper case. VSE provides several UCBs, by default; see *z/VSE System Control Statements*.

If you want to load the UCB under z/VM, use the LOADBUF command:

```
loadbuf 00e ucs p64 fold
```

This command loads UCB P64 into the printer at address 00E, and the FOLD option translates lower case to upper case when printing. You can load the buffer at IPL time by including this LOADBUF command in AUTOLOG1's PROFILE EXEC.

Note: You must drain the printer before loading any buffer under z/VM. Use the following command:

```
drain 00e
```

Loading a Forms Control Buffer

There are two recommended ways to use a printer with VSE:

- Dedicate the printer to the VSE guest and start a POWER print writer without the VM parameter. In this case, POWER loads the FCB.
- Share the printer among the VSE guest(s) and z/VM users. In this case, start the POWER print writer with the VM parameter. POWER sends the FCB as part of the print file, and it passes to z/VM the forms ID and number of copies. You do not have to load the FCBs for any POWER output.

If you want to load the FCB in the real physical printer under z/VM, you must use the z/VM LOADBUF command:

```
loadbuf 00e fcb fcb1
```

This command loads the FCB called FCB1 on the real printer at address 00E. Find a list of the FCBs available with z/VM in *z/VM: CP Planning and Administration*.

If you want to use a special FCB, you must have two identical FCBs with the same name—one created under VSE and the other loaded into the VM nucleus. Confirm that your special FCB is working by dedicating your printer to z/VM.

The LOADVFCB command can be used in installations that do not have an FCB-capable printer. The virtual machine's directory entry must specify a 3203, 3262, 3289E, or 4245 even though both the program and operating system have a real 1403 printer defined. Then the LOADVFCB command can be used to specify a virtual FCB image for 1403 printers so that programs that use printer overflow sensing can be spooled to a disk.

For details on how to load the FCB and UCB in VSE, see the *z/VSE System Control Statements*.

Starting the VSE/POWER Printer

When you run VSE under z/VM, the address you use to start a VSE/POWER printer must be a virtual address.

VSE/POWER loads the correct FCB.

Use the following command to start the printer:

```
s lst,cuu,a,,vm
```

This starts a list-writer task to send spooled output to the virtual printer with address *cuu*. The *vm* operand tells VSE/POWER that the device is a virtual device owned by z/VM.

When *vm* is specified, VSE/POWER:

- Issues NO FORMS CHANGE messages
- Loads the FCB
- Spools to the specified user ID (DEST=(,*user ID*)
- Prints or punches only one copy of the spool file
- Tells z/VM about class, forms name, and number of copies
- Closes the device at the end of entry and names the spool file after the VSE/POWER job name and gives the file the job's number.

For more information, refer to *VSE Power Administration and Operation*.

Varying Devices Offline and Online

Once you IPL the virtual machine, the devices that were not accessible to that machine at IPL are considered offline. However, the operator can attach more devices to this machine and have them placed online as required. The operator can enter the CP VARY, CP GIVE, and CP ATTACH commands to make the devices available for use by a particular virtual machine.

For example, if a graphics device is offline and VSExxxx needs the device, notify the operator to attach real device 080 to VSExxxx at virtual address 080.

```
#cp msg operator Please attach 080 to VSExxxx as 080
```

The operator enters:

```
vary online 080
```

The system responds:

```
080 VARIED ONLINE
```

Operator enters:

```
attach 080 to vsexxxx 080
```

The system responds:

```
080 ATTACHED
```

The operator informs VSExxxx that the graphic device is now attached and ready for use.

Switching Devices between Users

It is possible to switch devices such as tape drives and printers between z/VM users. For example, you might have a z/VM system that has a VSE virtual machine user (for example, VSExxxx) and a CMS user (CMSUSER). At different times, each system might want to use a tape drive or printer. Because a tape drive cannot be shared, the device has to be attached to one user at a time. If CMSUSER has only class G privileges, a message must be sent to the operator to attach 181 to CMSUSER. For example:

```
#cp msg operator Please attach 181 to CMSUSER as 181
```

When the tape drive is attached, the operator notifies CMSUSER with the message:

```
ATTACHED 181 TO CMSUSER AS 181
```

CMSUSER has the tape drive for as long as needed. When CMSUSER is finished, enter the following command:

```
#cp detach 181
```

As soon as the tape drive is detached, it is available to other users through the ATTACH command.

Using the GIVE command, the operator can transfer control of a dedicated tape drive. The RETURN option returns tape drive to the operator after the target user detaches it. The LEAVE option, when used with the RETURN option, leaves the tape in its current position when the target user detaches it. The UNLOAD option rewinds and unloads the tape when the target user detaches it. You can even specify whether the tape drive is to be attached to the target virtual machine in read only mode or in read/write mode.

z/VM can do all the printing for its users, or VSE does its own. For a VSE virtual machine to do its own printing, the printer has to be attached to the VSE machine with the CP ATTACH command. (A user ID with class G privileges cannot enter the CP ATTACH command. A message must be sent to the operator as shown in the example above.)

Note: If z/VM was previously doing the printing, the printer must be drained before attaching it to the VSE guest. The CP DRAIN command brings the spooling system to a controlled halt or halts the activities on a device whose spooling status is to be changed. The operator enters:

```
drain 00e
```

The operator then enters:

```
attach 00e vsexxxx 00e
```

The printer is now available to VSExxxx for as long as needed. When the VSE guest no longer needs the printer, the printer must be stopped before it is detached. Enter the VSE/POWER command:

```
pstop 00e
```

Then, enter the CP command:

```
detach 00e vsexxxx
```

Defining and Using the Virtual Console Facility

If you use the DEDICATE or ATTACH control statements to assign a 3270 terminal to the VSE virtual machine (as shown in sample directory entries above), you need not follow the procedures discussed here. You can use the 3270 defined in the directory in display operator console mode.[1]

Otherwise, to use a 3270 display terminal as the primary console in display operator console mode, either have a SPECIAL statement in the VSE virtual machine's directory entry or enter the CP DEFINE GRAF command after logging on to z/VM:

- If you use the SPECIAL statement, it would appear in the directory as:

  ```
  SPECIAL 01F 3270
  ```

- If you do not use the SPECIAL statement, assume that the z/VM operator has enabled a local 3270 line. Enter the following CP DEFINE command:

  ```
  define graf 01f 3270
  ```

Either way, after you log on z/VM (by using the device specified in the CONSOLE statement) and after you load the operating system into the virtual machine (by using the IPL command), you must enter the CP DIAL command at the 3270 console that is to be used in display mode. This logically connects that 3270 console to the operating system.

Use the CP TERMINAL CONMODE 3270 command to obtain a display mode console for the VSE guest. The console of the VSE virtual machine is defined in the z/VM directory as:

```
CONS 01F 3215 T
```

The following command allows both z/VM and VSE operator to perform work from the same terminal. Before you IPL the VSE guest, define the type of console operation with the following CP command:

```
term conmode 3270
```

Note: If CMS is running when you enter the CP TERM CONMODE command, CMS abends.

1. IBM recommends that you use separate consoles for z/VM and the VSE guest machines and that you use the DIAL command to reach other systems.

The following command prevents CP messages from appearing on the VSE console, thereby giving the operator the impression of running stand-alone.

```
terminal breakin minimal
```

CP messages are displayed only when:

- A high-priority message is involved
- The CP function is requested.

Special Considerations for VSE Users Running under z/VM

When you use the VSE *Display System Activity* dialog and the *Display Channel and Device Activity* dialog to monitor system activity, remember that:

- The SIO per second rate from the *Display System Activity* dialog may seem unusually high. The dialog calculates the total I/O rate on the system, including unit record virtual I/O. To specifically monitor disk or tape device activity, use the *Display Channel and Device Activity* dialog.
- VSE accounting support is used. Therefore, some z/VM—simulated privileged instructions are not accounted for, but the data gives you a better overview of the VSE guest virtual machine.
- All data is valid, *except* for data displayed as the number of events per second (for example, SIO per second). This type of data describes only VSE activity.
- When the NOPDS option is being used, the dialog displays zeros for paging activity. This shows you that only z/VM handles paging.
- When the VSE guest is running MODE=ESA, the paging activity data reflects VSE paging.

See *VSE Operation* for more on these dialogs.

Submitting Jobs to the VSE Virtual Machine

There are two ways to submit a job to the VSE guest:

- "Submitting Jobs under CMS"
- "Submitting Jobs Using SUBVSE EXEC" on page 51.

Submitting Jobs under CMS

If you are under CMS and you wish to send a job to the VSE virtual machine for execution, then create the job stream using XEDIT or the editor of your choice. Before using the virtual punch to create jobs for the virtual machine, take the precaution of clearing any files that remain in it from previous jobs with the following command:

```
spool punch nocont purge
```

Now you can spool your punch to the VSE guest:

```
spool punch vsexxxx cl n
```

where *n* is the spool class of the VSExxxx virtual reader. The class can be changed by issuing the CP SPOOL command on the VSExxxx virtual machine. The default class is A. You can now enter the CMS PUNCH command:

```
punch filename filetype filemode (noh
```

This sends the job stream to the VSE virtual machine. A job stream spooled to the VSE virtual machine remains in the virtual reader until you instruct the reader to

begin reading it. From the VSE console, you must issue the POWER READER TASK START command to the virtual reader:

```
pstart rdr,cuu
```

The virtual device number must match the one specified in the directory entry for VSE virtual machine.

Submitting Jobs Using SUBVSE EXEC

You can submit jobs to a VSE virtual machine by using the SUBVSE EXEC. The SUBVSE syntax varies by VSE release.

Transferring Output with the z/VM Writer Task of VSE/POWER

The z/VM writer task transfers output created in VSE to the CMS user. It is a standard feature of VSE/POWER 2.2 and later versions.

For the z/VM writer task to work, you must use either the DEST parameter on the POWER LST/PUN cards or the PDEST/LDEST parameters on the POWER JOB card. Either way, you can transfer back to the original z/VM user ID all print and punch output submitted through SUBVSE.

Notes:

1. If an interactive computing and control facility (ICCF) user has a z/VM user ID with the same name as one of the DEST parameters, then the z/VM user receives the output from the z/VM writer task.
2. If you start a printer or punch task with the VM parameter, a POWER print writer enters the CP CLOSE command at the end of each file being spooled.

Example of the z/VM Writer Task

The z/VM writer task returns the created output to the CMS user ID USER2 if all of the following are true:

- The CMS user ID USER2 exists on the z/VM system.
- A virtual printer is started with the same class as specified on the POWER LST/PUN cards and with the VM parameter.
- The DEST (or LDEST or PDEST) for the output is the same as the CMS user ID.

You can enter the PSTART command from the VSE console or from the CMS user ID using VMCF as follows:

```
vsecmd vsexxxx pstart lst,05e,v,,vm
```

This assumes that the virtual printer is at address 05e and the files to be printed are of class v.

Initializing Minidisks

VSE always uses DOSRES and SYSWK1 as the volume IDs for its system disks. If under z/VM you run several VSE guest machines that do not share DASD, you will have several DOSRES and SYSWK1 volumes.

z/VM, however, does not accept duplicate volume IDs on real disks. To solve this problem, you must have two volume IDs:

- A unique volume ID on the real disk used by z/VM
- The label DOSRES or SYSWK1 on a minidisk.

To set this up:

1. Use ICKDSF to initialize and format the DASD that are to contain the minidisks. This creates the unique volume labels required by z/VM. For ECKD and CKD devices, the volume label and allocation byte map are on cylinder 0. For FBA devices, the volume label and VTOC are part of the first 16 blocks.
2. In the directory entries for each VSE guest machine, define minidisks on the DASD you initialized and formatted. *Do not* define minidisks that start on block or cylinder 0. For ECKD and CKD devices, minidisks can begin at cylinder 1. For FBA devices, they can begin at a MAX-CA boundary following the initial blocks reserved for use by z/VM.
3. Initialize the minidisks for each VSE guest system. To do this:
   a. Log on to z/VM using the user ID and password for each VSE system.

      Note: To do the following step, the directory entry for each VSE guest must contain an OPTION statement with the MAINTCCW option, or the user ID must have privilege class F.

   b. Use ICKDSF to initialize the minidisks.

VSE Interface

The VM/VSE interface provides functions for interfacing to one or more VSE guest systems from CMS.

The VSE interface routines are distributed in IJSYSRS.SYSLIB. You must obtain these routines from the library and install them on a CMS minidisk.

See *z/VSE Installation* for complete information on the interface routines.

Using the Data Compression Facility with VSE

The VSE/VSAM for VM version 6 release 1 (program number 5686-081) supports Data Compression Services to save DASD space in large customer databases. CMS and GCS will also support the VSE/VSAM for VM version 6 release 1 interface for Data Compression Services. When you use AMSERV to create a VSAM cluster, the COMPRESS parameter of the DEFINE function will allow record data to be compressed when it is written and will expand data when it is read. This parameter automatically lets VSAM know if the data is to be converted by VSAM when it is read or written; no application program changes are necessary.

Note: VSE/VSAM for VM version 6 release 1 (program number 5686-081) was withdrawn from marketing on 30 September 2005 and was withdrawn from service on 28 February 2007.

Using CMS/DOS with VSE

CMS/DOS (a subset of CMS) enables the CMS user to take advantage of the interactive facilities of z/VM, to develop programs, and to execute them in a virtual machine.

CMS/DOS support in z/VM is based on VSE/AF version 1. CMS/DOS contains terms for both CMS (in the form of commands) and VSE (in the form of control cards). It simulates many of the functions of VSE/AF.

How the Library Structure of VSE Restricts CMS Users

z/VSE has different library structures from earlier releases of VSE. It also manipulates the library in different ways. Thus, many CMS/DOS functions do not work with z/VSE unless you use alternative methods.

Using VSE Librarian Functions in CMS/DOS

You cannot use the following VSE librarian functions from CMS when you are using z/VSE:

DSERV

ESERV

SSERV

PSERV

RSERV

Alternatives When Using z/VSE

1. Use the librarian functions from the interactive interface of VSE. Because you cannot use these functions from CMS, log off or disconnect from the CMS virtual machine and DIAL into the VSE system. Do this from terminals defined with the SPECIAL statement in the z/VM directory.

   To end this session, log off from the interactive interface and log on again to CMS.

   To save time in switching between CMS and the interactive interface, you can use the VM/Pass-Through Facility. See “Using the VM/Pass-Through Facility” on page 54.
2. Another option is to create a CMS file with JCL and librarian statements. You can submit this file to the VSE system and route it back with the VM writer task. See “Transferring Output with the z/VM Writer Task of VSE/POWER” on page 51 for more information.

Other CMS/DOS Restrictions

Table 6 on page 54 describes other CMS/DOS restrictions that exist when using z/VSE.

Table 6. CMS/DOS Restrictions Using z/VSE.

Command or EXEC	Restriction	Comments
DOSLKED	You cannot use this command with the libraries of VSE.	You can still use this command when your input is a CMS TEXT file or a CMS DOSLNK file.
FCOBOL	You cannot use this EXEC with VSE libraries.	You can move the DOS/VS compiler to a CMS DOSLIBS. However, the COBOL source program can have no COPY or BASIS statements.
VMFDOS	You cannot use this command to load or scan modules from a VSE distribution library tape.	VSE SYSIN tape format is still supported.
FETCH	You cannot use this command to fetch a phase from a VSE library.	You can still use this command to fetch a phase from a CMS DOSLIB.
SET DOS ON mode	You receive an error message if you use this command with the *mode* operand.	Use SET DOS ON without filemode to activate CMS/DOS.
ASSGN	You cannot assign the following system logical units to a VSE disk: SYSCLB, SYSRLB, SYSSLB with VSE.	You can still use this command to assign all other logical units to input and output devices.
DLBL	IJSYSCL, IJSYSRL, and IJSYSSL are incorrect file names for VSE libraries.	You can still use this command to identify CMS and VSAM files.

Using the VM/Pass-Through Facility

The VM/Pass-Through Facility is a z/VM optional licensed program. It enables a virtual machine on one system to pass through to an operating system or application on the same processor or any other processor defined to the PASSTHRU virtual machine.

To use the VM/Pass-Through Facility, you can either:

- Execute the PASSTHRU EXEC from the active CMS environment
- DIAL into the PASSTHRU virtual machine.

For details, see *VM/Pass-Through Facility Managing and Using*.

How to Switch between CMS and the Interactive Interface

Again, you can use PASSTHRU to reach the interactive interface and pass back to CMS. Use either a PF key or a four-character string (for example, %%%%) to toggle between the interactive interface and CMS.

By using the interactive interface, a CMS user can do many things:

- View the VSE system console from a CMS console
- Display VSE system activity
- Interactively display VTOC information
- Use the online problem determination dialog
- Display VSE/POWER queue entries
- Use VSE/ICCF.

Problem Determination and the VSE Virtual Machine

When a problem arises, observe all the symptoms. Ordinarily, these symptoms are error messages, abnormal ends of jobs, loops, wait states, or program checks. Also note special conditions associated with the problem. Such special conditions are likely to include one or more of the following:

- A new job running
- A recent sysgen
- A change in system configuration
- A new procedure in use
- Something different from the last time the run was made.

These are the kinds of conditions that may be related to the problem. Document the symptoms and conditions you observe. If it appears to be a hardware problem, call IBM Customer Service. If it appears to be a software problem, call the IBM Support Center.

Creating a Dump of the VSE Guest

To get a stand-alone dump of a VSE guest, proceed as though VSE were a native system but with these differences:

- Issue the CP STORE STATUS command instead of doing a machine save. Then enter

  ```
  set run off
  ```

- Attach a tape unit to the VSE machine before you IPL the VSE stand-alone dump program.

If you are using handshaking and if VSE storage is in the virtual machine, then you can use the CP VMDUMP command. However, you cannot use the VSE Info/Analysis program to debug VSE problems, because it requires the VSE DUMP utilities. These utilities allow you to debug problems interactively, create problem reports on the VSE machine, and go into dump scan mode.

Backup and Restore Procedure for the VSE Virtual Machine

It is advisable to back up your system on a regular basis. Use z/VM's DASD Dump/Restore (DDR), TAPE DUMP, MOVEFILE, VMFPLC2 DUMP, and VSE FASTCOPY to back up and restore the z/VM system or the VSE guest. Table 7 summarizes the function of each of these utilities:

Table 7. A Summary of Backup and Restore Functions in VSE

	VM DATA	**VSE DATA**
ALL	DDR	FASTCOPY or DDR
CMS	• TAPE DUMP • MOVEFILE • VMFPLC2 DUMP	
VSAM DATA SETS	Access Method Services	Access Method Services
Note: You cannot use DDR on a file that was backed up using VSE/FASTCOPY. Neither can you use VSE/FASTCOPY on a file that was backed up using DDR.		

To decide which backup method to use, ask yourself:

- What type of tape drive will be used in the backup procedure?
- How often will the system be backed up?

The access method service utilities back up and restore VSAM data sets in both z/VM and VSE.

Note: Before you use DDR to back up VSE, shut the VSE guest down.

Part 3. MVS under z/VM

This part of the book shows you how to plan for and how to operate MVS as a guest running under z/VM. It assumes that you have at least a working knowledge of both z/VM and MVS.

Chapter 4. Planning to Run MVS under z/VM

This chapter describes how to plan for an MVS guest to run under z/VM. It shows you how MVS and z/VM can exist in the same real system and how you can benefit from it.

Note that for the purposes of this discussion, MVS, OS/390®, and z/OS are interchangeable.

Creating Directory Entries

Establish a directory entry for each MVS guest that you want in your z/VM system. This chapter shows a sample directory entry for MVS running as a guest, followed by an explanation of each directory statement. For a complete list of z/VM user directory control statements with descriptions of all their operands and options, see *z/VM: CP Planning and Administration*.

Virtual Machine Configuration for MVS

Figure 10 on page 60 contains an example of a directory entry for an MVS guest.

Note: MVS does not support FBA DASD. You can run a mixed DASD configuration of FBA and CKD on a z/VM with an MVS guest system. However, the MVS guest is restricted to using only CKD DASD, just as it would be if it were running natively.

```
*
*********************************************************************
*                  SYSTEM RELATED USERIDS                           *
*********************************************************************
*
USER MVSTEST password 128M 256M BFG
 MACHINE ESA 2
 OPTION MAINTCCW LNKEXCLU
 ACCOUNT MVS00001 VM-FLOOR
  STDEVOPT DASDSYS DATAMOVER LIBRARY CTL
IPL CMS
XSTORE 1M
* MVS operator's console
 DEDICATE CC0 CC0
* MVS system volumes
 DEDICATE 1A2 1A2
 DEDICATE 1A3 1A3
 DEDICATE 170 170
 DEDICATE 171 171
 DEDICATE 172 172
* Console definition
 CONSOLE 01F 3270 C
* Spooled unit record devices
 SPOOL 01C 3505 A
 SPOOL 01D 3525 A
 SPOOL 010 4248 A
 SPECIAL 1A0 3270
 SPECIAL 1A1 3270
* Minidisk definitions
 MDISK 191 3390 275 002 H34P30 MR
* MVS system residence volume
 MDISK 179 3390 000 3339 SYSRES MW
* Links to CMS
 LINK MAINT 190 190 RR
 LINK MAINT 19D 19D RR
 LINK MAINT 19E 19E RR
* Link to a data repository
 LINK RPRDATA 291 292 ER
```

Figure 10. Sample Directory for MVS Guest

Description of Directory Control Statements

The USER Statement: The USER statement specifies a user ID and other virtual machine characteristics:

```
USER MVSTEST password 128M 256M BFG
```

MVSTEST
Defines *user ID* as MVSTEST.

password
password can be changed to the password of your choice.

128M 256M
The 128M entry defines the virtual machine's storage size at log on time. The 256M entry defines the maximum virtual machine storage size this user can define after logging on to the system.

BFG User class B is assigned so that the MVS virtual machine user can enter CP ATTACH and DETACH commands. User class F is for IBM service. Class G (general) is also assigned to the MVS virtual machine.

The OPTION Statement: The OPTION statement specifies optional services available to virtual machines:

```
OPTION MAINTCCW LNKEXCLU
```

MAINTCCW
The MAINTCCW operand lets a virtual machine initialize any DASD it uses.

LNKEXCLU
The LNKEXCLU operand gives a user the authority to use CP LINK commands that specify stable or exclusive link access to any minidisk for which that user has password authorization.

The MACHINE Statement: The MACHINE statement specifies the virtual machine architecture:

```
MACHINE ESA 2
```

ESA Defines this user as an ESA virtual machine. MVS must run as ESA.

2 Indicates the maximum number of virtual processors this user can define.

The ACCOUNT Statement: The ACCOUNT control statement specifies an account number and a distribution identification:

```
ACCOUNT MVS00001 VM-FLOOR
```

The ACCOUNT statement is optional. If omitted, both the account number and the distribution code are the user ID by default. The ACCOUNT statement must follow the USER statement.

The IPL Statement: The IPL statement automatically IPLs a system either by name (for saved systems) or by device address. For example:

```
IPL CMS
```

IPLing CMS lets you run a CMS PROFILE EXEC. A sample for MVS is shown in Figure 12 on page 68.

The STDEVOPT Statement: The STDEVOPT statement specifies the optional storage device management functions available to the guest virtual machine. The directory entry example contains the statement:

```
STDEVOPT DASDSYS DATAMOVER LIBRARY CTL
```

DASDSYS DATAMOVER
Specifies that the guest virtual machine is authorized to control and process Concurrent Copy and Peer-to-peer remote copy Establish Pair CCW. If the DATAMOVER parameter is not explicitly coded, the default is NODATAMOVER, which specifies that the virtual machine is not authorized for Concurrent Copy sessions and Peer-to-peer remote copy Establish Pair order. CONCOPY and NOCONCOPY are still accepted for compatibility and provide the same authorization control as DATAMOVER and NODATAMOVER, respectively.

LIBRARY CTL
Specifies that the guest virtual machine is authorized to issue tape library commands to control a 3495 Tape Library Dataserver. If the CTL parameter is not explicitly coded, the default is NOCTL, which specifies that the virtual machine is not authorized to control a tape library.

The DEDICATE Statement: The DEDICATE control statement specifies that a real device is to be dedicated to this user ID. An example of one of the DEDICATE statements in the sample directory entry is:

```
DEDICATE CC0 CC0
```

This statement dedicates as virtual device number CC0 the device at real address CC0 (the first CC0 is the virtual address). A real device can be dedicated to only one user at a time.

The CONSOLE Statement: The CONSOLE statement specifies a virtual machine console:

```
CONSOLE 01F 3270 C
```

01F Indicates the virtual address of the console

3270 Indicates the type of virtual console

C Indicates the spool class of the output.

See “Console Definitions” on page 63 for more on defining a virtual console in MVS.

The SPOOL Statement: The SPOOL statement specifies the unit record device that is to be spooled. Several readers, punches, and printers may be specified, each on a separate SPOOL statement.

The SPECIAL Statement: The SPECIAL statement defines special virtual devices which are not connected with real devices at definition time. For example:

```
SPECIAL 1A0 3270
```

1A0 Indicates the virtual address of the device

3270 Indicates the type of virtual device.

The MDISK Statement: The MDISK statement defines minidisks for virtual machines. In our directory entry example, we have:

```
MDISK 179 3390 000 3339 SYSRES MW
```

179 Specifies the virtual device number of the minidisk

3390 Specifies the device type

000 Specifies the starting cylinder number

3339 Specifies the number of cylinders allocated to the minidisk

SYSRES
Specifies the label (volume serial number) of the minidisk

MW Specifies that the access mode for this minidisk is multiple write.

The LINK Statement: The LINK statement is used to obtain access to another user’s minidisk. In Figure 10 on page 60, MVSTEST links to three CMS disks owned by MAINT. For example:

```
LINK MAINT 190 190 RR
```

MAINT
Is the target user ID (the user ID to which the MVSTEST virtual machine is linking).

190 190
The first 190 indicates the virtual address of the target minidisk. The second 190 indicates the virtual address that the linking virtual machine (MVSTEST in this case) uses for the minidisk.

RR Indicates an access mode of read-only.

In another example from Figure 10 on page 60, the LINK statement is used to obtain exclusive read access to a disk owned by RPRDATA:

```
LINK RPRDATA 291 292 ER
```

Notes:

1. Because the LNKEXCLU option was specified with the OPTION statement, the ER suffix is not needed to authorize exclusive read access to the RPRDATA minidisk. The LNKEXCLU grants exclusive access authority for *all* minidisks for which the user has password authorization. The ER suffix is specified here to show an example, even though the RR suffix would be sufficient.
2. A virtual machine that has the LNKEXCLU operand with the OPTION statement in its directory can use LINK to gain stable as well as exclusive access to minidisks.

Because exclusive is the highest level of read authorization, a user with that authority can use the LINK command twice—first to obtain read-only access, and then later, to obtain exclusive read access. For example, you can enter:

```
link rprdata 291 292 rr
```

for read-only access to the RPRDATA minidisk. Later, when you need to ensure that no one else can access that minidisk while you examine some of its files, you enter:

```
link rprdata 291 292 er
```

This second LINK command overrides the first, and you have exclusive access to the RPRDATA minidisk.

If you make any changes to the directory, you must file it. Then, you must use the DIRECTXA utility, which processes the directory file to see if it follows the required format. To actually change or swap the current active z/VM directory, you must have write access to the system-owned (system residence or IPL device) volume that contains the current directory. You must have access up to and including the directory cylinders or to the volume that is to contain the new directory.

Enter:

```
directxa filename
```

Console Definitions

You have two kinds of virtual console to consider when running MVS under z/VM:

- The *log on console* is the virtual machine console. Use this console to enter CP commands.
- The *MVS operator console* communicates with MVS. Use this console to enter MVS commands.

You can use two separate terminals as your log on and MVS operator console, or you can use one terminal for both.

If you use separate terminals, your directory entry should look like this:

```
CONSOLE 01F
DEDICATE CC0 CC0
```

The console at virtual address 01F is your log on console. From there, you can log onto your guest virtual machine and IPL the guest. You can then disconnect the log on terminal. The terminal at real address CC0 is now your MVS operator console.

If you want to use the same terminal as your log on and MVS operator console, create a PROFILE EXEC that automatically sets up a 3270 console for you. See "Using a CMS PROFILE EXEC to Automatically IPL MVS" on page 68.

```
CONSOLE CC0 3270
```

Your log on console is at virtual address CC0. The 3270 specification lets the MVS guest share a locally attached terminal controlled by CP. The MVS guest can use the terminal in full-screen mode, while CP shares the terminal and uses it as a line device.

To use this terminal as both a log on console and an MVS operator console, proceed as follows:

1. Log onto the guest's virtual machine.
2. The address specified in the CONSOLE statement should match one of the console addresses defined in a CONSOLxx member of SYS1.PARMLIB that will be selected for the MVS IPL as a console or alternate console. If the CONSOLE statement in the directory does not match a console address specified in the CONSOLxx member of SYS1.PARMLIB to be selected for the MVS IPL, use the CP DEFINE command to correct it.
3. *Do not* disconnect the virtual machine.
4. Enter:

   ```
   #cp terminal conmode 3270
   ```

5. IPL the guest operating system.

Consider the following before you make your directory definitions for consoles:

The terminal type and virtual address of the MVS operator console (in this case, a 3278 at real address CC0) must be the same as specified in the CONSOLxx member of SYS1.PARMLIB.

```
CONSOLE   DEVNUM(CC0) ALTERNATE(CC1) ROUTCODE(ALL)
      UNIT(3270-X)
      AUTH(MASTER)
      AREA(NONE)
      DEL(RD)
      MFORM(J)
      RNUM(19)
      RTME(2)
      SEG(19)
```

and your directory definitions might look like this:

```
DEDICATE E20 E20
DEDICATE E21 E21
DEDICATE E22 E22
```

Tape Definitions

Tape drives can be used by only one virtual machine at a time. Dedicate a tape drive to a virtual machine (usually a production machine) this way:

```
DEDICATE 580 580
```

where the first 580 is the address of the virtual tape drive and the second 580 is the address of the real tape drive.

It is possible to switch devices such as tape drives and printers among virtual machines. Because these devices cannot be shared, a device must be attached to one user at a time. You can do this by entering the CP ATTACH, CP GIVE (for tape drives), and CP DETACH commands.

Using the GIVE command, the operator can transfer control of a dedicated tape drive. The RETURN option specifies that the tape drive is to be returned after it is detached by the receiver. The LEAVE option, when used with the RETURN option, specifies that the tape is to be left in its current position when it is detached by the receiver. Use the UNLOAD option to specify that the tape is to be rewound and unloaded when it is detached. Other options specify whether the tape drive is to be attached to the target virtual machine in read only mode or in read/write mode.

Crypto Definitions

To run Integrated Cryptographic Service Facility (ICSF) applications, MVS guests must be configured to use the z/VM guest support for crypto. MVS guests may use the following types of cryptographic hardware:

- The IBM CP Assist for Cryptographic Function (CPACF), which is available on the processor board
- One or more IBM optional cryptographic features, with PCI-X adapters that can be individually configured as a secure coprocessor or as an accelerator for SSL.

The following directory control statements and commands are necessary to use z/VM guest support for cryptographic hardware. For more information on the directory control statements, see *z/VM: CP Planning and Administration*. For more information on the commands, see *z/VM: CP Commands and Utilities Reference*.

The CRYPTO Directory Statement

This statement is required and defines which APs or AP the guest will be using.

DOMAIN
Defines AP queues, if available.

APDED
Specifies which APs the guest may use.

APVIRT
This option is not used for z/OS guests.

Note: Take care when defining the same domains for more than one user in the CP Directory. The first user with the domains defined who logs on will get them. This also applies to the corresponding AP queues.

In this example, the guest will have dedicated access to AP queues 4, 5, and 6 on APs 2 and 3:

```
CRYPTO  DOMAIN  4 5 6  APDED 2 3
```

The APs specified must be selected from the set of APs selected on the PCI Cryptographic Online List on the Crypto Image Profile Page for the logical partition. The DOMAINs specified must be selected from the set of domains specified on the Usage Domain Index selections on the Crypto Image Profile Page for the Logical Partition. For more information about the Crypto Image Profile Page, see the *Processor Resource/Systems Manager Planning Guide*, SB10-7041.

The QUERY VIRTUAL CRYPTO Command

This command queries the status of the virtual cryptographic facilities defined for the virtual machine:

```
QUERY VIRTUAL CRYPTO
```

The QUERY CRYPTO Command

This command queries the status of the cryptographic hardware and the status of installed AP queues:

```
QUERY CRYPTO
QUERY CRYPTO APQS
```

Recommendations on the Configuration

Follow these recommendations when running MVS as a guest of z/VM:

- Define the MVS master console on a different control unit from the MVS log on console. Otherwise, you may experience console lockouts.
- For any MVS virtual machine, there is the choice as to whether to SET SVC76 CP or to SET SVC76 VM.
  - SET SVC76 CP lets z/VM collect all error records in one location. It also lets CP translate your virtual machine's storage addresses into host addresses, and your virtual machine's I/O device addresses into real device numbers.
  - SET SVC76 VM sends the error records to MVS's SYS1.LOGREC data set, and address translation does not occur.

    Note: SET SVC76 CP is convenient because all error records are stored in one location, but since the error records are not written to SYS1.LOGREC, MVS error recovery may be adversely affected.

  For detailed information on SET SVC76 see the *z/VM: CP Commands and Utilities Reference*.
- z/VM simulates the hardware load parameter facility for virtual machines. This lets the MVS virtual machine operator pass a load parameter (up to 8 bytes of data) when using the IPL command to load an alternative nucleus.
- MVS cannot operate in an XC virtual machine.

Limiting the Resources of MVS Guests

When MVS IPLs, WTOR messages can appear. If the operator does not answer each of these messages *promptly*, then MVS might consume all of the resources it is authorized to use. This can cause inconvenience throughout the system. Here are a couple of ways to avoid this problem:

- Limit the resources to which your MVS guests have access. For example, you might logically partition the processor and use LPAR resource capping.
- If you prefer to IPL MVS manually, be certain that a responsible operator monitors the console and responds to the IPL messages correctly and promptly.

Virtual Multiprocessing

MVS can run in a multiprocessor (MP) environment and can have many more virtual processors defined than are available in the real system. For the sake of performance, however, you should not define more virtual processors than the hardware offers.

See "Multiprocessing Considerations" on page 7 for more information on virtual and real multiprocessing.

Preparing to IPL the MVS Virtual Machine

If you use the same console as both the log on console and the MVS operator console, enter the CP TERM commands shown in Figure 11 before you IPL MVS. Or, use the PROFILE EXEC in Figure 11 to do this automatically.

```
/*********************************************/
/*                                           */
/*   EXEC to set up CP commands for MVS      */
/*                                           */
/*********************************************/

TRACE C
ADDRESS COMMAND

'CP TERM BRKKEY PF12'
'CP TERM BREAKIN MINIMAL'
'CP TERM CONMODE 3270'
```

Figure 11. Sample PROFILE EXEC for MVS

Note: Refer to usage notes for the TERMINAL command in the *z/VM: CP Commands and Utilities Reference*, if you use an SNA/CCS terminal.

`CP TERM BRKKEY PF12` lets you set the CP break key (normally PA1) to another program function (PF) key. This is helpful because MVS uses PA1 to retrieve the last command entered. If you do not set the BRKKEY to something other than PA1, you drop into a CP READ state if you press PA1 instead of retrieving the last command.

`CP TERM BREAKIN MINIMAL` prevents CP from interrupting the screen when a message must be displayed at your terminal. For example, if someone sends you a message, the terminal beeps; but, your MVS console is not cleared. When you want to display the message, press the break key to drop into CP READ.

`CP TERM CONMODE 3270` places your virtual console in 3270 mode. You must include this command when you run a PROFILE EXEC because CMS internally sets your virtual console to a 3215 when it is IPLed. CMS can no longer run once the command is entered, but CP can.

You can include other CP commands in a PROFILE EXEC, such as the IPL command. For more information, see IPLing the MVS Virtual Machine.

IPLing the MVS Virtual Machine

You can IPL MVS with or without using a PROFILE EXEC. The first method features the MVS guest (as defined in Figure 10 on page 60) being IPLed without using a PROFILE EXEC. In the second method, the same guest IPLs automatically via a PROFILE EXEC.

IPLing MVS without a PROFILE EXEC

After you have logged on to your virtual machine and before you IPL MVS, make sure your virtual machine size is adequate to your task. In the sample directory entry, the USER control statement is:

```
USER MVSTEST PASSWORD 128M 256M BFG
```

so user MVSTEST receives 128MB of storage at log on, and has a maximum storage capacity of 256MB. If you want, for example, 256MB, enter:

```
define storage as 256m
```

To IPL the guest, enter the command:

```
ipl 179
```

179 is the address of the MVS system residence volume as defined in the directory entry for the guest.

Note: If the MVS system recognizes specifying a LOADPARM on the hardware console, the LOADPARM option can be used on the IPL command. For more information on the LOAD parameter refer to MVS/ESA *System Commands*.

Using a CMS PROFILE EXEC to Automatically IPL MVS

You can use a PROFILE EXEC, a sample of which is shown in Figure 12, to IPL MVS for you when you log on the VM virtual machine. This PROFILE EXEC is set up for the user ID MVSTEST (defined in Figure 10 on page 60), which has an MVS system defined at address 179.

```
/*******************************************/
/*                                         */
/*   PROFILE EXEC for user ID MVSTEST       */
/*                                         */
/*******************************************/

TRACE C
ADDRESS COMMAND

'CP MESSAGE OPERATOR *** IPL NOW IN PROGRESS'
    /* Informs the operator of the IPL */

nl='15'X
    /* Defines a new line character */

'CP COUPLE 5D0 TO MVSTEST2'
    /* Couples MVSTEST's channel-to-channel adapter to
    another user ID, in this case MVSTEST2 */

'CP TERM CONMODE 3270' || nl || 'IPL 179'
   /* Both the TERM and IPL commands must be issued at this point. */
   /* However, the TERM command disables CMS.                      */
   /* The two commands are concatenated.                           */
   /* If there was no concatenation the IPL 179 command should     */
   /* have been preceeded by CP and it would have worked.          */
```

Figure 12. Sample PROFILE EXEC for Automatic IPL of MVS

Using the COMMAND Directory Control Statement

As an alternative to using the EXEC procedure described above, you can also use the COMMAND directory control statement to specify a command or a list of commands to be issued automatically whenever the VM virtual machine is logged on. Note that the commands to be executed may be of any class, and that privileged commands can therefore be executed this way without having to give the virtual machine the associated privilege class.

See the COMMAND directory control statement in *z/VM: CP Planning and Administration* for more information.

Using the CP SET RUN ON Command

Unless you are using CP debug facilities, you should enter the CP SET RUN ON command. This lets you enter CP commands without stopping your virtual machine.

Virtual Channel-to-Channel Adapters

A virtual channel-to-channel adapter (3088) is defined for a virtual machine through a SPECIAL statement in its z/VM directory entry, the DEFINE CTCA command, or the DEFINE 3088 command. You must specify the virtual device number to be assigned to the virtual channel-to-channel adapter. The user ID of the virtual machine allowed to connect with the virtual channel-to-channel adapter through the COUPLE command is optional.

A virtual channel-to-channel adapter can be defined whether or not a real channel-to-channel adapter is present in the real machine in which CP runs. This facility can be used to test multisystem operating systems and configurations (for example, multiple MVS/JES3 systems) using only one real machine. It can also be used in a multisystem environment to perform multisystem operating system testing or new release testing concurrent with usual production operations.

Figure 13 illustrates an MVS/JES3 multisystem configuration consisting of two real machines in which both MVS/JES3 system testing and production work are being done. CP simulates the existence of two MVS/JES3 configurations. CP runs in one real machine to control the operation of three virtual machines.

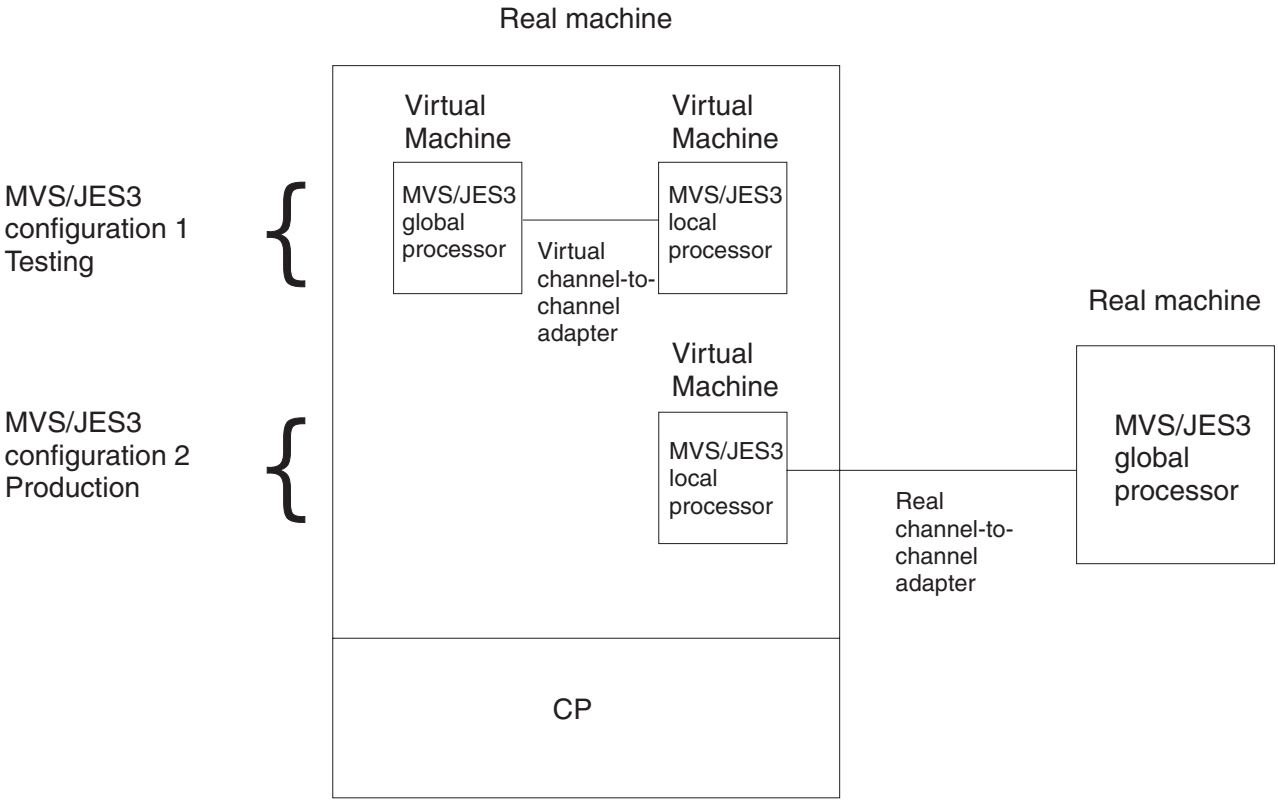

Figure 13. CP Support of Two MVS/JES3 Configurations

Two of the virtual machines are connected by a virtual channel-to-channel adapter to form the two processors in the MVS/JES3 configuration that is being used for

testing. The CP COUPLE command must be entered to connect these two virtual machines by the virtual channel-to-channel adapter defined for each virtual machine.

The third virtual machine is connected to the second real machine by a real channel-to-channel adapter to form the processors of the second MVS/JES3 configuration, which is doing production work. While CP can be used within an MVS/JES3 multiprocessing configuration, as shown, CP itself does not support loosely-coupled real-machine multiprocessing configurations.

The Hardware Console Integration Facility

MVS uses the hardware system console as an IPL and error recovery console. Under VM, the virtual machine operator's console simulates the hardware system console functions for the guest operating system.

Priority and non-priority messages from the MVS guest operating system are displayed at the virtual machine operator's console. The operator can send commands to the MVS operating system as if they came from the hardware system console. The following commands are used to perform the functions that would be handled at the hardware processor system console if MVS were running native:

```
QUERY VMSG
QUERY PVMSG
VINPUT
VDELETE
```

For more information about these CP commands, see *z/VM: CP Commands and Utilities Reference*. Also, refer to "Using the Hardware Console Integration Facility" for information about using CP commands with the HWCI facility.

Using the Hardware Console Integration Facility

The hardware console integration facility allows message traffic that occurs either before normal operating system connections have been established or when an error occurs to be displayed at the log on console. This support simulates hardware functions that let a guest operating system use its operator's console as an IPL and error recovery console. CP intercepts messages from the guest operating system that would, at first level, go to the system hardware console and reroutes them to the guest operator's console.

Here's what happens when CP intercepts a non-priority or priority message from the guest:

- The intercepted message is sent to the log on console.
- If the virtual machine operates in full-screen mode, and if TERM BREAKIN IMMED is set, then CP breaks in to display any message that arrives. If TERM BREAKIN GUESTCTL or TERM BREAKIN MINIMAL is set, CP breaks in only to display priority messages. Non-priority messages appear only when the operator presses the break key.
- The words "Prompt text:" precede the message text if the guest operating system designated the text as "prompt text". Held messages are preceded by the letter *H*.
- When a priority message is received, a CP message precedes the priority message on the console. The CP message alerts the operator that a priority message requiring a response follows.
  - CP places the log on console in CP READ mode, ready for the response.

Note: If you are running your virtual machine with SET RUN OFF, the MVS system in your virtual machine does not run while the log on console is in CP READ mode.

– Use the CP VINPUT command so that VM routes the MVS operating system command to the MVS guest.

CP displays all messages, and retains copies of the last 16 priority messages and the last 16 non-priority messages after displaying them at the log on console.

CP also retains for recall messages designated as “Held”. They must be explicitly deleted by use of the CP VDELETE command, specifying the message number(s), or by the guest operating system.

To delete a single, non-priority message, enter:

```
vdelete vmsg
```

where *vmsg* is the message number.

To obtain the message number, use the QUERY VMSG command to recall non-priority messages retained by CP. Enter:

```
query vmsg
```

Use the QUERY PVMSG command to recall priority messages. Enter:

```
query pvmsg
```

CP Commands to Know at the MVS Operator’s Console

When you run MVS under z/VM, there are times when you must simulate real processor or hardware functions. Use the following CP commands to simulate these real operator functions. Remember, to use any CP commands, you must press the break key that you previously defined.

EXT Use the EXT command to create an external interrupt in your virtual machine. It simulates pressing the interrupt key on the real system console. Usually, you do this when you lose your virtual console to MVS because of a console switch or a console I/O error.

READY
Use the CP READY command to set a device-end interrupt pending for a specified virtual device. This simulates “popping the plug” on a real disk device. You may have to do this on the few occasions when you mount a disk in MVS and the mount message does not disappear.

RESET
Use the CP RESET command to clear all pending interrupts from a specified virtual device. You can use this command to drop a dialed terminal from the virtual machine issuing the command.

SYSTEM
Use the CP SYSTEM command to simulate the action of the RESET and RESTART buttons on the real computer console and to clear storage. The operands of this command work as follows:

RESET
Resets all pending interrupts and conditions in the virtual machine.

RESTART
Simulates the hardware system RESTART function.

CLEAR
Clears virtual storage and virtual keys to binary zeros.

STORE STATUS
Stores selected virtual machine data in low storage.

Chapter 5. Operating an MVS Virtual Machine

This chapter contains information about how to log on, initialize, operate, and analyze the performance of an MVS guest under z/VM.

Note that the terms MVS, OS/390, and z/OS are interchangeable for the purpose of this chapter.

AUTOLOG Facility

AUTOLOG is a convenient way to log on MVS production systems with many I/O devices that run under z/VM.

In general, you should log on such virtual machines immediately after IPLing z/VM. Do one of the following:

- Have the system operator enter the CP AUTOLOG command before enabling user terminals.
- Define the AUTOLOG1 virtual machine in the directory entry. The AUTOLOG1 virtual machine is automatically logged on immediately after z/VM is IPLed and can be used to log on and IPL machines that need substantial storage. The AUTOLOG1 user ID is the default. The user ID can be any valid user ID on the system. You can change the AUTOLOG1 user ID by using the STARTUP operand on the SYSTEM_USERIDS statement in the system configuration file. For more, see *z/VM: CP Planning and Administration*.

Using the CP AUTOLOG Command

Before enabling user terminals, the z/VM system operator can enter the CP AUTOLOG command for each production virtual machine that requires substantial contiguous storage. The directory entry for the user ID specified by the CP AUTOLOG command must contain an IPL statement for the desired operating system. For more information about the CP AUTOLOG command, see *z/VM: CP Commands and Utilities Reference*.

Defining AUTOLOG1 in the System Configuration File

To use AUTOLOG1 to log on several virtual machines, use the STARTUP operand on the SYSTEM_USERIDS statement in the system configuration file or define directory statements to load CMS for the AUTOLOG1 user ID. The CMS PROFILE EXEC then contains several CP AUTOLOG commands. Each AUTOLOG command starts one virtual machine containing a production operating system. Each directory entry referred to by the CP AUTOLOG command must contain an IPL statement.

The CP AUTOLOG command in the PROFILE EXEC IPLs the virtual machine. You then gain access to the virtual machine by doing one of the following:

- Logging on with the user ID specified in the CP AUTOLOG command
- Entering the CP SEND command through the secondary user's console
- Entering the CP DIAL command and specifying the guest user ID.

Multiple Systems with AUTOLOG1

In Figure 14 on page 74, AUTOLOG1 starts CMS in a virtual machine.

```
USER AUTOLOG1 PASSWORD 512K 1M ABG
  ACCOUNT ACCTNO BIN1
  IPL CMS
  CONSOLE 009 3215
  SPOOL 00C 3505 R
  SPOOL 00D 3525 P
  SPOOL 00E 1403
  LINK MAINT 190 190 RR
  LINK MAINT 19E 19E RR
  LINK MAINT 19D 19D RR
  MDISK 191 3390 1 1 UDISKA WR RPASS WPASS
```

Figure 14. Directory Entry for the AUTOLOG1 Virtual Machine

In Figure 15, the CMS PROFILE EXEC has a CP AUTOLOG command for each virtual machine to be IPLed. In this way, the production virtual machines are automatically logged on in disconnect mode by the CMS PROFILE EXEC. Each user ID identified by the CP AUTOLOG command must also have an IPL CMS statement in its directory entry. The last CP command in the PROFILE EXEC logs off AUTOLOG1.

```
/* PROFILE EXEC to AUTOLOG several MVS virtual machines */

TRACE E
ADDRESS COMMAND
'CP SPOOL CONSOLE START'
'CP SET EMSG ON'

/* The following message will inform the operator that the guest */
/* operating systems are being autologged.                       */

'CP MSG OP The guest MVS virtual machines are being autologged.'
'CP AUTOLOG MVSUSER PASSWD1'
'CP AUTOLOG MVSUSER2 PASSWD2'
'CP AUTOLOG MVSUSER3 PASSWD3'
'CP ENABLE ALL'
'CP LOGOFF'
EXIT
```

Figure 15. PROFILE EXEC to Automatically Log on Several Virtual Machines

The AUTOLOG1 directory entry and PROFILE EXEC let the MVSUSER, MVSUSER2, and MVSUSER3 virtual machines log on the system in disconnect mode. You can access these virtual machines through their secondary user's consoles, if any, or by logging on with the user ID of MVSUSER, MVSUSER2, or MVSUSER3, along with the appropriate password.

How to Initialize a Minidisk for Use by MVS

Use ICKDSF with the MIMIC(MINI) parameter. See the *ICKDSF User's Guide* for specifics.

Using CP Commands to Enhance Performance

You can use certain CP commands to aid the performance of MVS guest virtual machines. Remember that improving the performance of one machine may impair the performance of others.

Before using the following commands, see *z/VM: CP Commands and Utilities Reference* or the chapter about tuning your system in *z/VM: CP Planning and Administration*.

Command	Description
LOCK	Use the LOCK command to permanently lock in real storage selected pages of a guest's virtual storage. Those pages are excluded from future paging activity. Make sure you have enough page frames available before issuing this command, or you will severely degrade the performance of other virtual machines.
SET SHARE	Use the SET SHARE command to control the percentage of system resources a user receives. A virtual machine receives its proportion of any scarce resource (processors, real storage, or paging I/O capability) according to its SHARE setting.
SET QUICKDSP	Use the SET QUICKDSP command to designate virtual machines that do not wait in the eligible list when they have work to do. Instead, these virtual machines are assigned an eligible list class of E0 and are added to the dispatch list immediately.
SET RESERVED	Use the SET RESERVED command to let a virtual machine have a specified number of pages resident in real storage. **Note:** System performance may degrade if you specify SET RESERVED for too many users.
DEDICATE	Use the DEDICATE command to allocate a processor to a virtual machine. Similarly, use the UNDEDICATE command to remove a dedicated processor from a virtual machine.
ATTACH XSTORE	Use the ATTACH XSTORE command to give the use of Expanded Storage to a virtual machine that supports it. **Note:** With MVS, you can configure XSTORE online even if it did not take effect when MVS was IPLed. But, while running MVS as a guest of VM, you cannot configure XSTORE online if the storage was attached after you IPL the MVS guest. Your configuration is what is "installed virtually" at IPL time. The same thing is true with additional virtual processors. MVS does not bring them online if they were not defined at IPL time.

Problem Determination

System problems appear in the usual way when you run MVS under z/VM. Abends, wait states, loops, and incorrect results are typical symptoms of system problems. When you run MVS under z/VM, any of these system problems may occur with either MVS or z/VM in control of the processor. See *z/VM: Diagnosis Guide* for a comprehensive description of symptoms and how to handle them.

Error Recording and Analysis

Two sources of information for analyzing system problems are the SYS1.LOGREC data set and the CP error recording virtual machine (EREP).

See the section "Recommendations on the Configuration" in Chapter 4 for information on error recording. For more information on SET SVC76 see the *z/VM: CP Commands and Utilities Reference*.

Also, CP collects error records for its MVS guest in an error recording virtual machine. To access this data, use the CPEREPXA utility.

CP Dumps

Whenever an abend occurs in CP, the module HCPDMP produces a system abend dump. A system abend dump occurs whenever the system operator presses the PSW restart key. When you obtain a system abend dump (also called a CP dump):

- Use the CP SET DUMP command to direct an abend dump to an output device.
- The system operator can produce a system abend dump by performing a PSW RESTART.

CP Trace Table: One of the most important areas in any CP dump is the CP internal trace table. This is the key to tracing the sequence of events that preceded the system problem. z/VM allows the spooling of trace data to an output device. Detailed information about the CP trace table is available in *z/VM: Diagnosis Guide*.

Trace Table Recording Facility: CP's tracing facilities expand problem determination capability for service personnel and system programmers. The facilities, which include the SET CPTRACE, TRSOURCE and TRSAVE commands, create system trace files containing CP and guest trace data. *Use these facilities to help analyze z/VM problems that you cannot detect with a system dump.*

The CP utility, TRACERED, is included as part of the tracing facilities. TRACERED uses the reader file as input and supports output to either a spooled print file or an interactive terminal display. For additional information on using the trace facilities, refer to *z/VM: CP Commands and Utilities Reference*.

MVS Dumps

MVS provides many service aids for collecting information on system problems. For details, see *MVS Diagnosis: Tools and Service Aids*.

SVC Dumps: SVC dumps for MVS are a useful source of information about MVS problems. At times, they provide clues to what appear to be hardware problems. Remember, MVS does not know that it is running under z/VM. Certain CP instruction simulation errors may appear to the MVS guest as hardware problems. For very difficult hardware-like errors, you may have to ZAP the MVS system to load a disabled wait PSW. This will allow you to dump the MVS system using the MVS stand-alone dump.

Stand-Alone Dumps: Creating a stand-alone dump of your MVS guest is not very much different from creating a "hands-on" dump:

1. Establish a READ/WRITE link with the disk on which you have stored the stand-alone dump program.
2. Go to CP mode and enter the `SYSTEM STORE STATUS` command.
3. Without using the `CLEAR` option, issue the `IPL` command, specifying the stand-alone dump program.
4. Go to the console generated for the stand-alone dump program, and press ENTER. Then, specify the device address on which the program is to record the stand-alone dump.

MVS Trace Table: The MVS system trace table is as important as the CP trace table when examining problems in MVS guests. It specifies the sequence of events that precedes a system malfunction.

In an MVS environment, this trace table contains all subchannel start instructions and the I/O interrupts that result. If the guest has entered a LINK, ATTACH, or DEDICATE command to a CP-generated device, you must examine both the CP trace table and the MVS trace table to understand the entire I/O event.

How to Diagnose a Problem

In general, approach CP abends and disabled wait states from a CP perspective. Examine the abend or wait state code first. Abends or loops may be more difficult to diagnose. Your first question should be, “Was the control program or the guest operating system in control when the system failed or when the loop began?”

As a general rule, assume the MVS guest was in control. Gather as much information as possible from MVS. In particular, find the MVS trace table and examine the trace entries for an understanding of MVS’s role in the problem, if any. Always examine all I/O control blocks (such as UCBs and IOBs) for possible hardware errors.

Coordinate the examination of the MVS trace table with an interrogation of the CP internal trace. Determine what has occurred in CP before the system malfunction. Remember that MVS operates as a guest machine under CP. CP still controls the scheduling and dispatching of the MVS guest and may have other virtual machines to serve. Use the trace tables of both systems to help diagnose those problems that cannot be solved with a single system’s trace table.

Accessing MVS from a VM Terminal

If you use terminals to log on as a TSO user under MVS, you may want to use the IBM Pass-Through Virtual Machine Facility (PVM) or the DIAL command to access the MVS guest. When you DIAL an MVS virtual machine, you logically attach a terminal to it so that MVS can communicate with the terminal.

To DIAL your MVS virtual machine, go to a free z/VM terminal (one with a z/VM logo) and clear the screen. When you are in CP READ, enter:

```
dial userid
```

where *userid* is the MVS guest machine.

This attaches your terminal to the specified user ID. z/VM selects the first (lowest address) virtual graphics device available as previously defined in the CP directory with the SPECIAL control statements or as defined by the CP DEF GRAF command.

When dialed, the virtual machine controls the terminal.

Unsupported Devices

You can use some I/O devices that z/VM does not support. To use an unsupported device, you must attach or dedicate the device to a virtual machine. The device cannot, therefore, be shared among users. You may use these dedicated devices only under these conditions:

- No timing dependencies exist in the device or the program

- No dynamically modified channel programs exist in the access method
- No special functions need to be provided by z/VM
- No other CP restrictions are violated. See *z/VM: CP Planning and Administration*
- The device is identified by an RDEVICE statement for an unsupported device in the system configuration file.

Note: z/VM does not support every device that its many potential guests support. Nevertheless, if a guest supports a particular device, you can attach it to the guest even if z/VM does not support it. However, you cannot IPL your guest if its IPL volume resides on a DASD that z/VM does not support.

Analyzing Performance

To analyze the performance of MVS under z/VM, you can use the Performance Toolkit for VM. The Performance Toolkit for VM provides short-term study and problem-solving as well as long-term trend analysis and capacity planning. For more information, see *z/VM: Performance Toolkit Reference*.

Within the z/VM environment, there are some commands you can use to gather performance information. For example, INDICATE commands provide a broad overview of how system resources are being used.

MVS under z/VM Operating Environments

When you analyze the MVS environment, remember that you have two operating systems running in a single processor. Both z/VM and MVS are vying for the basic system resources, such as processor, I/O, storage, and paging. Each is generating its own accounting information, and each is supplying its own performance information.

Remember that MVS is unaware that it is running as a guest under z/VM. What the MVS guest thinks is real-time is actually the time-of-day clock and processor timer facility. Elapsed time as measured by the time-of-day clock is accurate. The guest's virtual processor timer (VPT) runs whenever the guest is dispatched or is in a voluntary wait state. It does not run if the guest is in a CP wait state. Thus, when z/VM dispatches another virtual machine and later dispatches the MVS guest, MVS does not realize it had stopped running.

Information about guest system performance is frequently published in Washington System Center flashes. Contact your marketing team for obtaining flashes that pertain to your particular installation.

Sharing Data Between z/VM Host and MVS Guest

There are many methods for sharing data between VM and MVS. These methods include (but are not limited to):

- Define an RSCS/NJE connection between them. This enables TSO/E TRANSMIT and VM SENDFILE to work normally and without inconvenience to you.
- Define a VTAM/VTAM link to allow cross-domain activity from terminals defined in either the host or the guest (but not in both).
- APPC programming will allow transparent data sharing from within programs.

Part 4. VM under z/VM

This part of the book shows you how to plan for and how to operate VM as a guest running under z/VM using a parm disk with a module. It assumes that you have at least a working knowledge of VM.

Chapter 6. Planning to Run a VM Guest under z/VM

Running VM as a second level (guest) system of z/VM offers opportunities for more flexible system management. In this environment you can:

- Test new application programs
- Test new releases of VM
- Test new maintenance procedures and modifications
- Train operators and system programmers.

These activities can be independent of any other tasks because they are being performed in a virtual machine.

One of the biggest advantages of running a second level VM system is the ability to generate a new system without disturbing normal production activity. A system programmer can log on his own virtual machines and go through the generation steps at his own pace while the rest of the installation uses the real z/VM system undisturbed. You can use XEDIT to create and update the files that are used during system generation. When the system is tested, it can be placed online, replacing the previous version with minimal disruption to the production activity.

Notes:

1. Although you can use a second level z/VM system to test new releases, service procedures, and modifications, do not use it to create a back-level system.
2. You cannot run VM as a guest operating system in an XC virtual machine.
3. To run VM under z/VM, the real processor must be a System z.

Performance Considerations for VM under z/VM

Several factors make it hard to predict performance characteristics when VM is running at second level. These factors can be classified into broad groups:

1. Factors affecting the configuration, including:
   - The amount of caching space
   - The location of the paging areas on DASD
   - The size of the virtual machine.
2. Operating system workload factors, including:
   - The frequency of real interrupts
   - The frequency and type of privileged instructions
   - The frequency of start subchannel instructions.

How Configuration Influences Performance

A VM system running at second level needs more real storage, DASD space, and processor speed than the average CMS user. The overhead incurred by VM's increased need for dispatching, scheduling, and paging is relatively small compared with the increase in overhead from simulating privileged instructions.

There are several hardware configuration factors that influence the performance of a second level VM system:

- The amount of real storage available
- The amount of DASD caching space available
- The speed, capacity, and number of paging devices
- The amount of channel and control unit competition and the disk arm contention affecting each paging device

- Competition among system paging devices and devices for processing a user's I/O requests.

Workload Factors Influencing Performance

Usually, two factors influence the performance of a second level virtual machine:

- The total number of active virtual machines
- The type of work each virtual machine is doing, especially the amount of I/O processing required.

By measuring and evaluating the effects of these workload factors on a specific configuration, you can anticipate their effect on performance. After measuring the performance of the VM machine, you can improve VM performance by selecting special options. These options let you redistribute system resources, either to balance them or to favor a particular virtual machine over another.

The following performance options are available to as many virtual machines as you wish:

- Favored execution with a specified percentage (the CP SET SHARE ABSOLUTE command)
- Basic favored execution without a specified percentage (the CP SET SHARE RELATIVE command)
- Favored dispatching (the CP SET QUICKDSP command)
- Locked pages.

For more, see *z/VM: CP Planning and Administration* and *z/VM: CP Commands and Utilities Reference*.

Creating a Second Level VM System

This chapter describes how to define a simple second level VM system. More complex and functional second level systems can be built. But this simple system will demonstrate the basic steps and concepts that apply to all second level systems.

This system we define can only be used as a second level system since it uses a CMS formated IPL disk.

Define User to First Level System

Create the directory entry of the user ID that will own the second level VM system. Log on to your VM system as MAINT (or a user ID that has access to the directory), edit the directory source file and use either the DIRECTXA utility, or any other directory maintenance tool available to you (for example, DirMaint™).

The following material shows the system directory entry for the second level VM system.

```
*
USER VMTEST password 64M 128M BG
   MACHINE ESA 2
   OPTION TODENABLE
   ACCOUNT DEPTNNNN BINXXX
   IPL CMS
   CONSOLE 0009 3215 T OPERATOR
   SPOOL   000C READER  A
   SPOOL   000D PUNCH   A
   SPOOL   000E PRINTER A
   LINK MAINT 0190 0190 RR
   LINK MAINT 019D 019D RR
   LINK MAINT 019E 019E RR
   MDISK  191 3390 1561  70 OT02A4 MR
   MDISK 1000 3390 1601  35 OT02A4 MR
   MDISK 12FF 3390 1631  70 OT02A4 MR
*
```

Figure 16. Directory Entry for Second Level System User ID

Description of Directory Control Statements

The following is an explanation of the user directory control statements found in the sample directory entry shown in Figure 16. For a complete list of z/VM user directory control statements available with z/VM, see *z/VM: CP Planning and Administration*.

The USER Statement: The USER statement defines a user ID and its characteristics. VMTEST is the user ID that will own the second level system.

```
USER VMTEST password 64M 128M BG
```

VMTEST
Defines the user ID as VMTEST.

password
password can be changed to the password of your choice.

64M 128M
The primary address space size at logon for VMTEST is 64M. This is enough for a small second level virtual machine. The 128M entry defines the maximum virtual machine storage size this user can define after logging on to the system.

BG This virtual machine uses two privilege classes: B and G. User class B (resource) is assigned so that the second level virtual machine user can enter CP ATTACH, GIVE, and DETACH commands. This enables the user to attach and release real tape drives and printers for the second level system. Class G (general) users control the functions associated with the execution of their virtual machines.

Attention: You can give your VMTEST user ID the user classes that meet your needs, but do not assign class A. Privilege class A users can enter the SHUTDOWN command, accidentally shutting down the first level system.

The MACHINE Statement: The MACHINE statement specifies the architecture of the virtual machine.

```
MACHINE ESA 2
```

ESA z/VM running as a second level system requires the ESA architecture.

2 VMTEST can define a maximum of 2 virtual processors.

The OPTION Statement: The OPTION statement allocates special services to VMTEST.

```
OPTION TODENABLE
```

TODENABLE
The operand allows VMTEST to change the virtual machine's Time-of-Day (TOD) clock with the SCK instruction or the SET VTOD CP command. The SCK instruction is used during CP initialization if operator elects to change the TOD clock.

The ACCOUNT Statement: The ACCOUNT control statement specifies an account number and a distribution identification:

```
ACCOUNT DEPTNNNN BINXXX
```

The ACCOUNT statement is optional. If omitted, both the account number and the distribution code are the user ID by default.

The IPL Statement: The IPL statement automatically IPLs a system either by name (for saved systems) or by device address. For example:

```
IPL CMS
```

IPLing CMS lets you run a CMS PROFILE EXEC.

The CONSOLE Statement: The CONSOLE statement specifies the console address and device type.

```
CONSOLE 009 3215 T OPERATOR
```

009 Indicates the virtual address of the console

3215 Indicates the type of virtual console

T Indicates the spool class of the output

OPERATOR
When the primary user ID is running in disconnected mode, the secondary user ID receives all CP messages. Specifying OPERATOR as the secondary user ID, gives the operator added flexibility in an environment where several virtual machines are used. The operator can control several disconnected virtual machines (with the CP SEND command) from one physical terminal.

The SPOOL Control Statement: The SPOOL statement specifies the address of a unit record device:

```
SPOOL   000C READER  A
SPOOL   000D PUNCH   A
SPOOL   000E PRINTER A
```

If the first level system configuration is used for the second level system operation, the unit record addresses and type for the first and second level VM systems should match. If the first level system configuration is not used for the second level system's operation, the spool addresses and type should be defined at the first level so they are picked up at IPL time.

You can use the CP DEFINE command to add unit record devices. For example, you can add a printer to your second level machine by entering:

```
%cp define printer vaddr
```

The printer is added at the address specified by *vaddr*.

Note: The example used "%cp" rather than "#cp". The example assumes the TERMINAL BRKKEY of the user ID on the second level system was set to a different BRKKEY than that of the first level system. Doing so allows you to use the BRKKEY defined on the second-level system to communicate with the second level CP.

The LINK Statement: The LINK statement is used to obtain access to another user's minidisk. In Figure 16 on page 83, VMTEST links to three CMS disks owned by MAINT. For example:

```
LINK MAINT 0190 0190 RR
```

MAINT
Is the target user ID (the user ID to which the VMTEST virtual machine is linking).

190 190
The first 190 indicates the virtual address of the target minidisk. The second 190 indicates the virtual address that the linking virtual machine (VMTEST in this case) uses for the minidisk.

RR Indicates an access mode of read-only.

The MDISK Statement: The MDISK statement defines minidisks for virtual machines.

```
MDISK  191 3390 1561 70 OT02A4 MR
MDISK 1000 3390 1601 35 OT02A4 MR
MDISK 12FF 3390 1631 70 OT02A4 MR
```

MDISK 191 is VMTEST's A disk. It is used for the same purpose as an ordinary user's A disk. It has 70 cylinders since VMTEST might want to store extra CP modules or dumps from the second level system on it.

MDISK 1000 will be the second level system's IPL disk. The disk is also the Parm Disk and is defined large enough to hold the CP module and system configuration files.

MDISK 12FF will be used for paging and spooling on the second level system. The disk will also contain the system directory, warmstart and checkpoint areas.

Attention: IBM strongly recommends that no minidisk or temporary minidisk begin on real cylinder 0 of CP-owned volumes, because information of critical importance to CP is stored on that cylinder.

In the directory entry example, there is:

```
MDISK 1000 3390 1601 35 OT02A4 MR
```

1000 Specifies the virtual device number of the minidisk

3390 Specifies the device type

1601 Specifies the starting cylinder number

35 Specifies the number of cylinders allocated to the minidisk

OT02A4
Specifies the label (volume serial number) of the minidisk

MR Specifies that the access mode for this minidisk read/write.

Install The New Directory

Now that you have added the VMTEST user to the first level system source directory, use the DIRECTXA utility to make the changes active to the system. For more on DIRECTXA, see *z/VM: CP Commands and Utilities Reference*.

Logon VMTEST and Initialize Its 191 Disk

Now that the VMTEST user is defined to the system, you should log on to VMTEST and use the CMS FORMAT command to initialize its 191 disk. Also, create the PROFILE EXEC and PROFILE XEDIT files according to your personal tastes and your installation's standards.

Format and Allocate Space for the Second Level System

Before the second level VM system can use the CP disks for the virtual system residence, paging, and spooling volumes, you must format and allocate space for them. Because a virtual disk is being formatted, the cylinder or block specification should reflect the size of the virtual disk being used.

The following execs format and allocate VMTEST's 1000 and 12FF minidisk. It is up to you to allocate minidisks on VM in a manner that minimizes arm contention and eliminates physical overlap. For more information about defining and allocating minidisks, see *z/VM: CP Planning and Administration*.

The following exec which formats and allocates VMTEST's 1000 minidisk assumes that the minidisk is 35 cylinders, is linked in write mode, and the CP SALIPL utility is available on an accessed disk.

```
 0 * * * Top of File * * *
 1 /* Initialize minidisk at address 1000 */
 2 /* for second level system.            */
 3 Address 'COMMAND'
 4 queue '1'
 5 queue 'IPLDSK'
 6 'FORMAT 1000 B'
 7 'FORMAT 1000 B 34 ( RECOMP'
 8
 9 /* Write SAPL to parm disk in the RECOMP area */
10 'SALIPL 1000 ( ORIGIN 2000'
11 * * * End of File * * *
```

Figure 17. Initialize Minidisk at Address 1000

The following exec which formats and allocates VMTEST's 12FF minidisk assumes that the minidisk is 70 cylinders, is linked in write mode, and the CP CPFMTXA utility is available on an accessed disk.

```
 0 * * * Top of File * * *
 1 /* Format minidisk for directory, paging, spooling, */
 2 /* and temp disk space for a second level system.   */
 3 Queue 'FORMAT'
 4 Queue '12FF'
 5 Queue '000  69'
 6 Queue 'CPPK01'
 7 Queue 'YES'
 8 Queue 'PERM   0   4  '
 9 Queue 'DRCT   5   6  '
10 Queue 'TDSK   7  10  '
11 Queue 'PERM  11  26  '
12 Queue 'PAGE  27  40  '
13 Queue 'SPOL  41  69  '
14 Queue 'END'
15 'CPFMTXA'
16 Exit
17 * * * End of File * * *
```

Figure 18. Initialize Minidisk at Address 12FF

Setting Up the System Definition Files

Creating the SYSTEM CONFIG file

The material below shows a sample system configuration file for your second level system. For specific information on the system configuration file statements, see *z/VM: CP Planning and Administration*.

```
/*************************************************/
/* System Configuration File for second level system */
/*************************************************/

System_Identifier_Default  2NDLVL

Operator_Consoles 0009 001F

Emergency_Message_Consoles 0009 001F

System_Residence,
   Checkpoint Volid CPPK01 From Cylinder 1 for 2,
   Warmstart  Volid CPPK01 From Cylinder 3 for 2

CP_Owned  Slot  1 CPPK01       /* DRCT space is here */
CP_Owned  Slot  2 reserved
CP_Owned  Slot  3 reserved

DEVICES,
   Sensed         0000-ffff,
   Online_at_ipl  0000-ffff

User_volume_Include *

/* No CP Access disk are necessary since this system */
/* does not use any custom LOGO files or CP exits.   */
/* CP_Access OPERATOR 1000 B */

Storage,
   Trace Master 10 pages

Priv_Classes,
   Operator     A ,
   IOCP_Read    CE ,
   IOCP_Write   C ,
   HW_Service   F ,
   User_Default G

Features,
   Enable,
      Set_Privclass,
   Disable,
      Auto_Warm_Ipl,
      Logmsg_From_File,
      Clear_TDisk,
   Retrieve,
      Default 7,
      Maximum 255,
   Maxusers Nolimit,
   Passwords_On_Cmds,
      Autolog yes,
      Link Yes,
      Logon yes
```

Figure 19. SYSTEM CONFIG file for the Second Level System (Part 1 of 2)

```
System_userids,
   Account1 OPERACCT noAUTOLOG ,
   Erep1    OPEREREP noAUTOLOG ,
   Symptom1 OPERSYMP noAUTOLOG ,
   Startup  AUTOLOG1 noAUTOLOG ,
   Operator OPERATOR noDISCONNECT ,
   Dump     OPERATOR               /* not OPERATNS */

Timezone_Definition EDT West 4.00.00
Timezone_Definition EST West 5.00.00

Timezone_boundary  on 1999-04-04  at 02:00:00  to EDT
Timezone_boundary  on 1999-10-31  at 02:00:00  to EST
TimeZone_Boundary  on 2000-04-02  at 02:00:00  to EDT
TimeZone_Boundary  on 2000-10-29  at 02:00:00  to EST
TimeZone_Boundary  on 2001-04-01  at 02:00:00  to EDT
TimeZone_Boundary  on 2001-10-28  at 02:00:00  to EST
TimeZone_Boundary  on 2002-04-07  at 02:00:00  to EDT
TimeZone_Boundary  on 2002-10-27  at 02:00:00  to EST
TimeZone_Boundary  on 2003-04-06  at 02:00:00  to EDT
TimeZone_Boundary  on 2003-10-26  at 02:00:00  to EST
TimeZone_Boundary  on 2004-04-04  at 02:00:00  to EDT
TimeZone_Boundary  on 2004-10-31  at 02:00:00  to EST
TimeZone_Boundary  on 2005-04-03  at 02:00:00  to EDT
TimeZone_Boundary  on 2005-10-30  at 02:00:00  to EST
```

Figure 19. SYSTEM CONFIG file for the Second Level System (Part 2 of 2)

The system name is 2NDLVL. The checkpoint and warmstart areas are on the 12FF disk. All of the virtual machine devices which can be sensed are brought online when the second level system IPLs. The OPERATOR user ID will receive any system dumps.

To learn more about planning your system configuration file see *z/VM: CP Planning and Administration*.

Creating the LOGO CONFIG File

The material below shows a sample LOGO CONFIG file for your second level system. Because the CHOOSE_LOGO statement is not used, your second level system will use logos from HCPBOX in the CP module. For specific information on the logo configuration file statements and LOGO Files, see *z/VM: CP Planning and Administration*.

```
/************************************************/
/* Logo Configuration File for second level system */
/************************************************/
 Status VM_Read       'VM READ '   ,
        CP_Read       'CP READ '   ,
        Running       'RUNNING '   ,
        More          'MORE... '   ,
        Hold          'HOLDING '   ,
        Not_Accepted  'NOT ACCEPTED'
```

Figure 20. LOGO CONFIG file for the Second Level System

Second Level System Directory

Figure 21 on page 91 shows a sample user directory that can be used for your second level system. For general information about specifying directory entries, see *z/VM: CP Planning and Administration*.

Note: VM does not check for overlapping extents in the MDISK statement. Therefore, you must ensure that minidisk extents defined in the VM directory do not overlap each other and also do not overlap the alternate track cylinders or blocks. *If overlap conditions exist, file data damage is inevitable.* You can use the DISKMAP EXEC to check for overlaps and gaps between minidisks. The DISKMAP EXEC is described in *z/VM: CP Commands and Utilities Reference*.

Modify the sample system directory entry in Figure 21 on page 91 to suit your own testing needs. The first line in the directory that is not a comment must be the address (12FF in our example) and the volume label (CPPK01 in our example) of the device on which directory is written.

```
DIRECTORY 12FF 3390 CPPK01
*-------------------------------------------------------------
* User Directory for second level system
*-------------------------------------------------------------
PROFILE COMMON
  MACHINE ESA
  SPOOL   00C RDR  A
  SPOOL   00D PUN  A
  SPOOL   00E 1403 A
  CONSOLE 01F 3215 T
  LINK OPERATOR 0190 0190 RR
  LINK OPERATOR 019D 019D RR
  LINK OPERATOR 019E 019E RR
*-------------------------------------------------------------
USER OPERATOR AB12CD  32M 64M ABCDEFG
  IPL 190 PARM AUTOCR
  MACHINE ESA 4
  ACCOUNT DEPT0001 BIN001
  SPOOL    00C RDR   A
  SPOOL    00D PUN   A
  SPOOL    00E 4248  A
  CONSOLE  01F 3270  T
  * R/W disks
  MDISK   191 3390  000  END VMT191 WR ALL
  MDISK  1000 3390  000  END IPLDSK WR ALL
  MDISK  12FF 3390  000  END CPPK01 WR ALL
  * R/O disks
  MDISK   190 3390  000  END MNT190 RR ALL
  MDISK   19D 3390  000  END MNT19D RR ALL
  MDISK   19E 3390  000  END MNT19E RR ALL
*-------------------------------------------------------------
* Show allocation cyl 0
USER $ALLOC$  NOLOG
  MDISK   001 3390  000  001 IPLDSK RR
*-------------------------------------------------------------
* Show allocation of checkpoint area
USER $SYSCKP$ NOLOG
  MDISK   001 3390  001  002 IPLDSK RR
*-------------------------------------------------------------
* Show allocation of warmstart area
USER $SYSWRM$ NOLOG
  MDISK   001 3390  003  002 IPLDSK RR
*-------------------------------------------------------------
* Show allocation of directory space
USER $DIRECT$ NOLOG
  MDISK   001 3390  005  002 IPLDSK RR
*-------------------------------------------------------------
* Show allocation of temp disk space
USER $T-DISK$ NOLOG
  MDISK   001 3390  007  004 IPLDSK RR
*-------------------------------------------------------------
* Show allocation of paging space
USER $PAGE$   NOLOG
  MDISK   001 3390  027  014 IPLDSK RR
*-------------------------------------------------------------
* Show allocation of spooling space
USER $SPOOL$  NOLOG
  MDISK   001 3390  041  029 IPLDSK RR
*-------------------------------------------------------------
```

Figure 21. Second Level System's User Directory (Part 1 of 2)

```
* Need this in order to allocate soft abend file.
USER OPERATNS AB12CD  32M 64M BCEG
  * IPL 190 PARM AUTOCR
  INCLUDE COMMON
  ACCOUNT DEPT0001 BIN001
  MDISK   191 3390  011  002 IPLDSK MR ALL
  MDISK   192 3390  013  005 IPLDSK MR ALL
*-------------------------------------------------------------
USER OPERACCT AB12CD  32M 64M BG
  * IPL 190 PARM AUTOCR
  INCLUDE COMMON
  ACCOUNT DEPT0001 BIN001
  IUCV  *ACCOUNT
  MDISK   191 3390  018  002 IPLDSK MR
*-------------------------------------------------------------
USER OPEREREP AB12CD  32M 64M BFG
  * IPL 190 PARM AUTOCR
  INCLUDE COMMON
  ACCOUNT DEPT0001 BIN001
  IUCV  *LOGREC
  MDISK   191 3390  020  002 IPLDSK MR ALL
*-------------------------------------------------------------
USER OPERSYMP AB12CD  32M 64M BCEG
  * IPL 190 PARM AUTOCR
  INCLUDE COMMON
  ACCOUNT DEPT0001 BIN001
  IUCV  *SYMPTOM MSGLIMIT 4
  MDISK   191 3390  022  002 IPLDSK MR ALL
*-------------------------------------------------------------
USER AUTOLOG1 AB12CD  32M 64M ABCDEG
  * IPL 190 PARM AUTOCR
  INCLUDE COMMON
  ACCOUNT DEPT0001 BIN001
  MDISK   191 3390  024  001 IPLDSK MR ALL
*-------------------------------------------------------------
*   THE FOLLOWING ARE FOR GENERAL USERS
*-------------------------------------------------------------
USER USER1    AB12CD  32M 64M G
  INCLUDE COMMON
  IPL 190
  MACHINE ESA 4
  ACCOUNT DEPT0001 BIN001
  LINK MAINT 0191 0191 RR
*-------------------------------------------------------------
USER USER2    AB12CD  32M 64M G
  INCLUDE COMMON
  IPL 190
  MACHINE ESA 4
  ACCOUNT DEPT0001 BIN001
  LINK MAINT 0191 0191 RR
```

Figure 21. Second Level System's User Directory (Part 2 of 2)

As shown in Figure 21 on page 91, OPERATOR defines the CP volume CPPK01 as minidisks 12FF. This makes it possible for OPERATOR to reinstall the directory using DIRECTXA while logged onto the second level system.

Also, OPERATOR defines the parm disk as address 1000. This is done so OPERATOR can access and update the system configuration file or CP module. Note that this usage is OK in our example since the system configuration file does not access the parm disk via the CP_ACCESS statement.

In the first-level directory entry for user VMTEST, address 190 is linked to a minidisk containing CMS and has a label of "MNT190". The system configuration file contains these statements.

```
DEVICES,
   Sensed         0000-ffff,
   Online_at_ipl  0000-ffff

User_volume_Include *
```

These statements allow the the second level system to recognize MNT190 and associate it with address 190 of OPERATOR. MNT190 is the first-level minidisk label and the “real” volume label of the DASD on the second level system. The 19D HELP file minidisk and the 19E program products minidisk are defined in a similar manner.

Likewise, the OPERATOR’s 191 disk on the second level system is defined with a label, device type and size that match the 191 disk of first level user VMTEST. You will need to modify this statement if the label of the VMTEST 191 minidisk is not “VMT191”.

The user directory uses the word “END” to specify the size of any minidisk that is “brought up” to the second level system. This feature is especially useful because your second level system’s user directory automatically accommodates for changes made to system minidisks on the first level system.

Change and Install Second Level System’s User Directory

Create a user directory for your second level system using the directory shown in Figure 21 on page 91 as a basis. Use the DISKMAP EXEC to check for overlaps and gaps between minidisks.

Use DIRECTXA to install your directory onto the 12FF minidisk. Expect either a return code of 4 or 5 from DIRECTXA when installing the second level system directory on CPPK01 while running DIRECTXA on the first-level user VMTEST. For either case, the directory on CPPK01 has been updated. (This assumes, of course, that there were no errors in any of the directory statements.) For more on DIRECTXA, see *z/VM: CP Commands and Utilities Reference*.

Building the Second Level System’s CP Module

When preparing your second level VM system, you can create either a new CP module or make a copy of the first level system’s CP module. You must create a new CP module if you are testing new service levels of z/VM. If you have to create a new CP module, use the appropriate exec described in *z/VM: Service Guide* and *z/VM: VMSES/E Introduction and Reference*.

Also, contact your system programmer to find out about any local modifications to the z/VM service environment.

IPLing the Second Level VM System

With few exceptions, IPLing a second level system is similar to IPLing a first level VM system. You must verify (by entering a CP QUERY VIRTUAL command) that the virtual machine configuration matches or is a subset of the configuration defined in the system configuration file of the second level system. In particular, verify that:

- The console definition is compatible (address and console mode)
- The virtual machine mode is ESA or XA
- Enough virtual storage is defined.

Once this is done, you can IPL the virtual disk which contains the Stand Alone Program Loader (SAPL). In our case, this is disk 1000.

Note: If your warm start area is unnecessarily large, your IPL will not succeed. This is because during IPL the warm start area may have to be stored temporarily in virtual storage. If the virtual machine storage is too small or if there is an excessively large warm start area, then any attempt to place it in the guest's virtual storage does not succeed. Be certain, therefore, that the warm start area is of reasonable size. Four cylinders of 3390 DASD space is more than sufficient for most installations.

Other factors that affect the storage requirements of your VM guest include:

- The number of devices defined for the system on the RDEVICE statement in the system configuration file
- The trace table size per processor
- Whether the HCPLDR operand PAGEB was specified when building the CP module (This significantly increases the amount of storage that CP nucleus needs.)

As long as these factors are kept at reasonable levels, you should have no problem bringing up your second level system.

To IPL your second level system, enter the following series of commands:

```
system reset
terminal conmode 3270
set machine esa
ipl 1000 clear
```

Figure 22. Commands to IPL the Second Level System

The SYSTEM RESET command stops CMS and prevents any CMS console abends resulting from the TERMINAL command. The TERMINAL command changes the console mode from 3215 which CMS uses to 3270 which is required by CP or SAPL. The SET command defines the guest machine's mode as ESA.

Note: You can use the command **ipl 1000 clear loadparm** *consaddr* instead of the IPL shown if you wish to see the Stand Alone Program Loader screen. Pressing PF10 on the SAPL screen will continue loading the second level system.

The system's response to this series of commands is similar to the following:

```
z/VM  V5 Rn.n  SERVICE LEVEL nnnn (64-BIT)
SYSTEM NUCLEUS CREATED ON yyyy-mm-ss AT hh:mm:ss, LOADED FROM IPLDSK

************************************************************
* LICENSED MATERIALS - PROPERTY OF IBM*                    *
*                                                          *
* 5741-A05 (C) COPYRIGHT IBM CORP. 1983, nnnn. ALL RIGHTS  *
* RESERVED. US GOVERNMENT USERS RESTRICTED RIGHTS - USE,   *
* DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE *
* CONTRACT WITH IBM CORP.                                  *
*                                                          *
* * TRADEMARK OF INTERNATIONAL BUSINESS MACHINES.          *
************************************************************

Using parm disk on volume IPLDSK (device nnnn)
Parm disk resides on blocks 0 through nnnnn.
*****************************************
*  Processing System Configuration file   *
*****************************************

Start ((Warm|Force|COLD|CLEAN) (DRain) (DIsable)  (NODIRect)
      (NOAUTOlog)) or (SHUTDOWN)
```

Figure 23. Messages from CP Initialization

Enter:

```
cold drain
```

Because this is a test system, there is no data or accounting information to be recovered. Therefore, you can perform a cold start, unless for some other reason you are obliged to perform a warm start. During subsequent IPLs your system, reply “WARM DRAIN” to the start type prompt.

The system responds:

```
COLD DRAIN
NOW hh:mm:ss tmz day-of-week yyyy-mm-dd
CHANGE TOD CLOCK (YES|NO)
```

Figure 24. Time-of-Day Clock Prompt

Where:

hh:mm:ss
: specifies the current time as per the time zone used

tmz
: specifies the time zone of your installation specified on the TIMEZONE_DEFINITION statement in the system configuration file.

day-of-week
: specifies the day of the week

yyyy-mm-dd
: specifies the date

The VMTEST user ID was defined with the TODENABLE option so you can change your virtual machine’s time-of-day clock if you wish. If you do not wish to change the time-of-day clock, just press ENTER.

The system responds:

```
The directory on volume CPPK01 at address 12FF has been brought online.
HCPWRS9205A
HCPWRS9205A Checkpoint data is not valid.
HCPWRS9205A System Data file recovery data may not be valid.
HCPWRS9205A Continuation of the system IPL could result in the
HCPWRS9205A loss of system data files.
HCPWRS9205A
HCPWRS9205A To continue COLD start and attempt to
HCPWRS9205A recover system data files, enter GO.
HCPWRS9205A To stop processing, enter STOP.
```

Figure 25. COLD Start Verification Prompt

Because this is the first time that you are IPLing your second level system, these messages are normal. Reply “GO” to the prompt.

```
GO
HCPWRS2513I
HCPWRS2513I Spool files available     NONE
HCPWRS2512I Spooling initialization is complete.
DASD 12FF dump unit CP IPL pages 2974
There is no logmsg data
FILES: 0002 RDR,   NO PRT,   NO PUN
LOGON AT hh:mm:ss tmz day-of-week yyyy-mm-dd
GRAF  0009 LOGON  AS  OPERATOR USERS = 1
HCPIOP952I 0032M system storage
FILES: 0000002 RDR, 0000001 PRT,      NO PUN
HCPAAU2700I System gateway 2NDLVL identified.
DMSWSP327I The installation saved segment could not be loaded
z/VM Vn.n.n  yyyy-mm-dd hh:mm
 Ready;
```

Figure 26. The Remaining System Initialization Messages

Once you IPL the second level system, you are automatically logged on as the system operator in line mode. At this point you can enable graphic display devices to let other users dial into this system and log on the second level system.

Saving Second Level CMS

To load CMS into your virtual machine, do one of the following:

- IPL the CMS system disk
- IPL a named saved system.

To IPL the CMS system disk (in our example, MAINT’s 190 minidisk), the virtual machine must have at least 20MB of virtual storage. If a named saved system is used for CMS, it can be IPLed in 5MB of virtual storage.

Note: To minimize performance problems, keep each virtual machine in your system to a reasonable minimum size.

To define a named saved system for CMS, use the SAMPNSS exec on Maint’s 193 disk.

```
Logon Maint pw
IPL 190 cl
ACC 193 R
SAMPNSS CMS
```

The values on the DEFSYS command vary from release to release. See *z/VM: Guide for Automated Installation and Service*. The system responds:

```
HCPNSD440I The Named Saved System (NSS) CMS was successfully defined
in file ID 0001.
```

Enter:

```
cp ipl 190 parm savesys cms
```

The system responds:

```
HCPNSS440I Named Saved System (NSS) CMS was successfully saved in
fileid 0001.
z/VM Vn.n.n  yyyy-mm-dd hh:mm

Ready;
```

Once the READY message appears, a copy of the named saved system has been saved on the second level system.

Chapter 7. Operating VM under z/VM

This chapter contains information on how to operate a VM operating system running as a guest of z/VM. For more information about operating a z/VM operating system see the *z/VM: System Operation* book.

Operating the Second Level Virtual Machine

Virtual machine operation at this level can be confusing. At all times, you must be aware of which level of VM you are interacting with and what functions you are trying to perform.

Entering CP Commands to the First Level z/VM

While running VM under z/VM, use CP commands to:

- Communicate with the first level z/VM system
- Query the status of virtual machine devices or spool files
- Attach or detach devices from the virtual machine configuration

If you are using the same terminal session for both the first level and second level console, you can communicate with the first level CP by doing one of the following:

- Pressing the ATTN key
- Pressing the key that is defined as the CP TERMINAL BRKKEY on the first level user ID.

  Tip: Set the TERMINAL BRKKEY of the user ID on the second level system to a different BRKKEY than that of the first level system. Doing so allows you to use the BRKKEY defined on the second level system to communicate with the second level CP.

Enabling Terminals for a VM Guest

Most testing can be done by initializing and running tests from other than the operator's virtual machine. You can define additional graphic devices so that additional users can log on your second level system. Use the SPECIAL statement in the first level user ID's directory entry to define a graphic device (3270) when the user logs on. Use the CP DEFINE GRAF command to dynamically define a graphic device at any time. You can confirm that the virtual device is defined by issuing a CP QUERY VIRTUAL GRAF command to first level CP.

Enter the following on the first level user ID:

```
#cp query virtual graf
CONS 0009 ON LDEV L002C                          SUBCHANNEL =  0000
GRAF 0060 NOT DIALED SUBCHANNEL =  0021
```

The "NOT DIALED" status indicates that the address 60 is defined but no terminal has dialed into it.

On another terminal (on the same first level system), enter:

```
dial vmtest 060
```

Note: If you enter the CP DIAL command without a specified address, VM connects the terminal to the unused graphic device with the lowest address in the specified user ID's virtual configuration. If no lines are available or if all lines are busy, VM sends an error message and does not make the connection.

Return to the OPERATOR user ID on the second level system by entering BEGIN. You can then VARY ON and ENABLE the device from the second level system:

```
vary online 060

0060 varied online
1 device(s) specified; 1 device(s) successfully varied online

enable 060

COMMAND COMPLETE
```

You should now be able to see the logo for the second level system. This console remains connected until one of the following happens:

- The virtual machine logs off using standard log off procedure
- The virtual machine is forcibly logged off
- The terminal is turned off and then turned on
- When the CP RESET command is issued from the first level console of the user ID owning the second level system or by a user authorized with the CP RESET command.

Once disconnected, the end user is free to use the DIAL command to connect to another user ID.

Varying Devices Offline and Online

Once you IPL the second level virtual machine, the devices that are not accessible by that machine at IPL are considered offline. Any devices you specified on the DEVICES statement in the system configuration file to be offline, will be offline. However, you can attach more devices to your machine and have them placed online as required. For example, tape drives can be attached by the first level machine operator to the virtual machine configuration at the address that matches the configuration of the second level CP system. You can change these virtual addresses to conform to your second level system's device definitions by using the CP DEFINE command. The second level VM operator then enters the CP VARY command.

For example, if a real graphic device is offline and VMTEST wants the graphics device made available, notify the first level system operator with a message:

```
#cp msg operator Please attach 080 to VMTEST as 080
```

The first level system operator then enters:

```
vary online 080
```

The system response to the first level system operator is:

```
080 VARIED ONLINE
```

The first level system operator enters:

```
attach 080 to vmtest 080
```

The system response to the first level system operator is:

```
080 ATTACHED TO VMTEST 080
```

The second level system operator then must enter:

```
vary online 080
```

The system response to the second level system operator is:

```
080 VARIED ONLINE
```

The device is now ready for the second level VM system to use it. The operator informs VMTEST that the graphics device is now attached and ready for use.

Note: If the graphics device is a VM-supported terminal, the CP ENABLE command must be entered before the end user can log on to the second level system that is using the device.

Spooling Options When Running VM under z/VM

When running VM under z/VM, spooling occurs in the first level VM system and may occur in the second level VM system.

First Level VM System Spooling

If your virtual machine produces a large volume of unit record output and keeps a unit record device constantly busy, then you may want to dedicate a unit record device to that virtual machine. To eliminate double spooling of printer output, include a DEDICATE statement in the first level system's directory entry, such as:

```
DEDICATE 00E 002
```

This statement causes all output from the second level system virtual printer 00E to go directly to the real printer at address 002.

You can also have the system operator dynamically dedicate a unit record device to your virtual machine. For example, if your first level system has a 4245 printer, you can have the operator enter a CP ATTACH command before the second level system is IPLed. Send the operator a message requesting that *cuu* be attached to VMTEST as 00F, where 00F is the address of the 4245 printer on the real system.

Enter:

```
#cp msg operator Please attach real printer 00F to me as 00F
```

If the real printer at 00F is not in use by the system or any other virtual machine, the operator enters the ATTACH command.

```
attach 00F to vmtest as 00F
```

When the device is attached, VM sends a confirmation message to VMTEST:

```
PRT 00F ATTACHED
```

Second Level VM System Spooling

If the virtual machine performs any spooling operations, second level CP is also spooling (unless it has dedicated unit record devices). This double spooling operation does not create a problem. Second level CP detects that it is running in a virtual machine and at the end of each spooled output file issues a CP CLOSE command to first level CP. This produces real spooled output for virtual spool files.

Notice that double separators occur. For instance, the separator page on virtual printed output includes four pages - two pages for the second level VM system and two more pages for the separator of the first level virtual machine on which the virtual CP system is running. The extra set of separator pages can be avoided by using the START command with the NOSEP option on the second level system.

Problem Determination for the Second Level VM System

This section discusses error recording and dumps in the second level VM system.

Error Recording and Analysis

When running VM under z/VM, all hardware errors are recorded in the error recording area of the first level system. To access the recorded data, use the CPEREPXA utility.

Dump Procedure

To place a dump of the second level system in its virtual reader, you must:

1. Specify a user ID that is defined on the second level system for the DUMP operand on the SYSTEM_USERIDS statement in the system configuration file.
2. IPL the second level system. The system issues the equivalent of CP SET DUMP CP during initialization.
3. Issue the SNAPDUMP command on the OPERATOR of the second level system or SYSTEM RESTART on the first level user ID.

For more debugging information, see *z/VM: Diagnosis Guide* or *z/VM: Dump Viewing Facility*.

Backup and Restore Procedure for the VM Guest

This section contains information about backup and restore procedures for the second level VM system.

Creating a DASD Dump Restore (DDR) Utility Tape

VM supplies a utility known as VM DASD Dump Restore (DDR). This utility lets the user dump, restore, copy, or print VM user minidisks and system volumes. The DDR utility:

- Dumps part or all of the data from a direct access storage device (DASD) to tape
- Transfers data from tapes created by the DDR dump function to a DASD, which must be the same DASD type as the one that originally contained the data
- Copies data from one device to another of the same type
- Prints selected parts of DASD and tape records in hexadecimal and EBCDIC on the virtual printer
- Displays selected parts of DASD and tape records in hexadecimal and EBCDIC at the terminal.

The first file of any DASD backup tape should contain a copy of the stand-alone DDR program. This ensures that the system can be restored with the same level of the DDR utility used to make the duplicate. Also, another tape should contain all the VM utilities, such as DDR, format/allocate, and the Device Support Facilities. This tape should be used in emergency situations, such as when encountering a bad copy of the DDR utility on a backup tape.

For now, concentrate on putting the stand-alone DDR utility on tape. Mount a tape in read/write mode. Log on as VMTEST and attach the tape drive as 181 to VMTEST. To actually put the DDR utility on the tape, enter the following series of commands:

```
ipl cms
cp rewind 181
filedef input disk ipl ddr s
filedef output tap1
movefile input output
cp rewind 181
```

The FILEDEF commands define the input as coming from the file IPL DDR on the CMS system disk, and the output as going to the tape at virtual address 181. The MOVEFILE command then takes a copy of the IPL DDR program and puts it on the tape. By rewinding the tape after the MOVEFILE is completed, you can enter IPL 181 to verify the results of the MOVEFILE. You now have a stand-alone DDR utility tape.

DASD/DUMP RESTORE and the Second Level System

Once you have created the stand-alone DDR utility tape, you can create a backup copy of the IPLDSK and CPPK01 disks. IPLDSK contains the parm disk. CPPK01 contains important system-related areas such as the cylinders for the checkpoint, directory and warm start areas, and several minidisks belonging to such users as OPERATNS, OPERACCT, OPEREREP, OPERSYMP and AUTOLOG1. To maintain the current level of your system, simply back up this entire volume. You then are able to recover the second level system completely if the need arises.

The process of creating a backup copy of the IPLDSK volume is simple. The copy is made on the same tape as the DDR utility. Make sure the tape containing the DDR utility created under “Creating a DASD Dump Restore (DDR) Utility Tape” on page 102 is still on the drive and attached to the VMTEST user ID.

Make sure you start the tape at the beginning. Enter:

```
cp rewind 181
```

When the tape is positioned at the beginning, load the DDR utility into storage. Enter:

```
cp ipl 181 clear
```

The system responds:

```
z/VM DASD DUMP/RESTORE PROGRAM
ENTER CARD READER ADDRESS OR CONTROL STATEMENTS
ENTER:
```

Request that the message be printed at the terminal. Enter:

```
sysprint cons
```

You want the input to come from the system volume IPLDSK at virtual address 1000. Enter:

```
input 1000 3390 ipldsk
```

Specify your output device type. Enter:

```
output 181 tape
```

The entire volume is saved. Enter:

```
dump 000 034
```

The system responds:

```
ENTER NEXT EXTENT OR NULL LINE
```

To enter a null line, just press ENTER. The null line is entered because all the extents have already been supplied.

The system responds:

```
DUMPING IPLDSK
```

Following this message, the system informs you which cylinders or blocks are being recorded on tape. When the message END OF DUMP appears at the terminal, enter a null line to end the DDR program.

When all the information has been recorded on tape, the tape must be rewound and removed from the tape drive. You should note the date and time of the DDR backup on the tape and also keep a record of which tape was used for the backup. You may need to restore your system later, and by keeping records of when backups were made and what tapes were used, you can save time.

Now that you have successfully saved a copy of the IPLDSK volume, you can do the same for the CPPK01 volume. You can detach the tape at address 181 and put a second tape on that drive. Perform the same steps for creating a stand-alone DDR utility but supply this information instead.

```
input 12ff 3390 cppk01
dump 000 069
```

The other DDR statements are identical except for the dump cylinder range and when the message END OF DUMP appears at the terminal the CPPK01 volume has been saved.

Restoring Your Second Level System

Restoring the second level system from the DDR tape is similar to creating the backup copy of the IPLDSK volume. When you created the backup copy, the INPUT statement specified virtual address 1000, and the OUTPUT statement specified virtual address 181. These addresses are reversed during the restoring procedure. (VMTEST owns the minidisk at virtual address 1000, which contains the full system volume IPLDSK.) Attach the tape containing the IPLDSK volume and make sure the tape is rewound.

Attention: Be certain to write-protect the tape to prevent accidental damage.

Enter:

```
cp rewind 181
```

When the tape is positioned at the beginning, load the DDR utility into storage. Enter:

```
cp ipl 181 clear
```

The system responds:

```
z/VM DASD DUMP/RESTORE PROGRAM
ENTER CARD READER ADDRESS OR CONTROL STATEMENTS
ENTER:
```

Request that the information be printed at the terminal. Enter:

```
sysprint cons
```

Specify your input device type. Enter:

```
input 181 tape
```

Output is to the system volume IPLDSK at virtual address 123. Enter:

```
output 1000 3390 ipldsk
```

Restore everything from tape. Enter:

```
restore all
```

Following the RESTORE command, the message, `RESTORING IPLDSK` appears at the terminal, followed by information about which cylinders or blocks are being written to DASD. When the message `END OF RESTORE` is displayed on the screen, enter a null line to end the DDR program.

When all the information on the tape has been restored to the IPLDSK volume, the tape is rewound and unloaded from the drive. The old system is restored and ready for operation.

The steps for restoring the CPPK01 volume are the same as those to restore the IPLDSK volume. Detach the tape containing the IPLDSK backup and attach the tape containing the CPPK01 backup. After DDR has been loaded from the CPPK01 backup, enter the same set of input statements to the DDR program as you did for the IPLDSK restoration with these exceptions:

```
output 12ff 3390 cppk01
```

When the `END OF RESTORE` message appears, the CPPK01 volume has been restored, and your second level system contains exactly what it did when you saved it earlier.

Part 5. Guest Coupling Simulation Support

This part describes how to plan and setup a simulated Parallel Sysplex® within a z/VM system.

Chapter 8. Setting Up and Running Guest Coupling Simulation Support

z/VM is able to simulate advanced CF (Coupling Facility) and MF (Message Facility) function, namely, the ability of a set of CFs to be directly connected to one another. z/VM Guest Coupling Simulation provides for the simulation of one or more complete parallel sysplexes within a single z/VM system image. The intent is to provide a pre-production testing platform for a coupled-system installation. The z/VM simulated environment is not intended for production use since its single points of failure negate the intent of the parallel sysplex environment. Other than the processors required, there is no special hardware needed: no coupling links and no external coupling facilities. Neither is such hardware supported. z/VM Guest Coupling Simulation does not run on Integrated Facility for Linux® (IFL) processors. All guest operating systems coupled within a simulated sysplex can be coupled (through simulated coupling links) only to coupling facilities also running as guests of the same z/VM system. Up to 32 virtual machines can be coupled within a simulated sysplex, with each such virtual machine coupled to up to 8 coupling facility virtual machines. In addition, when the processor and z/VM are so capable, each simulated Coupling Facility can connect to up to 7 peer simulated Coupling Facilities. All coupled guests and simulated Coupling Facilities run within a single instance of the Control Program (CP).

Understanding a Parallel Sysplex Configuration

In order to understand the value of z/VM Guest Coupling Simulation Support, a brief description of a Parallel Sysplex environment is needed. For simplicity, this chapter will refer to MVS, OS/390, and z/OS operating systems simply as MVS.

The purpose of a sysplex configuration is to provide a processing environment that could be available twenty-four hours a day, seven days a week and is totally scalable. This is accomplished by connecting multiple MVS images running on various processors or logical partitions in parallel. Although each MVS image is independent, to the end user a single system image is reflected. Processing is balanced across all the systems within the sysplex. If any processor within the sysplex requires service or maintenance, the processor can be removed from the sysplex without affecting system availability. The other MVS images can maintain the current workload. If more processing power is ever required, you can add new processors and additional MVS images to the sysplex at any time.

For the multiple MVS images to run as a single processing complex, there has to be an efficient way for all the images to share information and serialize their processing. The Coupling Facility and Message Facility were the mechanisms developed to perform just that function. The Coupling Facility is a logical facility with its own processor(s) which basically contains structured storage that is accessible to all the MVS images. The Coupling Facility is attached to the processors by special fiber-optic channels (the Message Facility) that are used to transfer commands, data, and responses between the attached processors and the Coupling Facility. The Coupling Facility will allow MVS to maintain such things as data, common scheduling queues, status and locks that are shared by all the MVS images. In addition to the shared storage, the Coupling Facility also provides a command-set to perform operations required to control a parallel system environment.

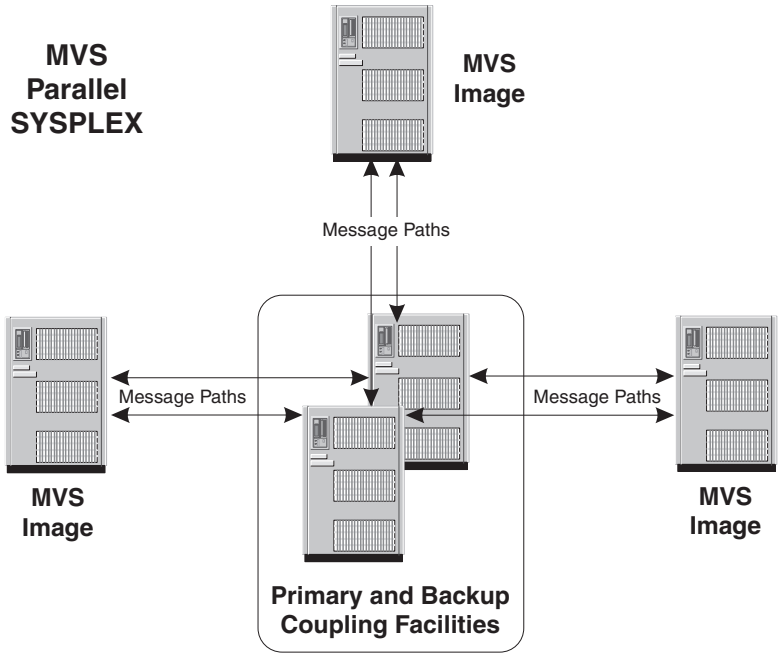

Figure 27. Parallel Sysplex Example. This figure illustrates an MVS sysplex environment using multiple Coupling Facilities.

z/VM Simulation of a Parallel Sysplex Configuration

z/VM Guest Coupling Simulation Support is the software that simulates the hardware and software required to run an MVS sysplex environment as second level guests under z/VM.

Components of the z/VM Guest Coupling Simulation Support

The z/VM Guest Coupling Simulation Support consists of three components: Coupling Facility Service Machines, Coupled Guests, and simulated Message Facility. Figure 28 on page 111 illustrates the simulated Coupling Facility environment in z/VM.

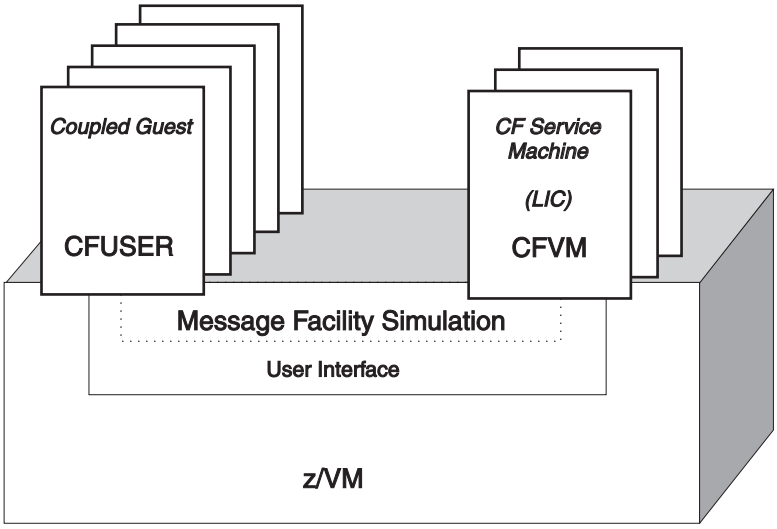

Figure 28. z/VM Coupling Facility Support

The CF Service Machine

This is a special type of z/VM service machine that runs the Coupling Facility Control Code (CFCC) (Licensed Internal Code). This code is not part of z/VM and is loaded directly from the processor. Since z/VM is executing the actual code that runs in a real Coupling Facility, there is basically no difference in function between a real Coupling Facility and z/VM's implementation.

The system administrator must define a user ID for each CF Service Machine that will run under z/VM. To define each user ID as a CF Service Machine, the CFVM operand must be specified on the OPTION directory control statement of each CF Service Machine. The CFVM operand defines the virtual machine as a CF Service Machine and allows connections between itself and coupled guests. Refer to the *z/VM: CP Planning and Administration* for more information about the OPTION and SPECIAL directory control statements.

Any number of CF Service Machines can be defined. Up to 32 coupled guests can be connected to each CF Service Machine. When the processor and z/VM are so capable, each CF service machine can couple to up to 7 other CF service machines.

The Coupled Guest

This is a uni-processor (UP) or multi-processor (MP) virtual machine that has a coupled connection to a CF Service Machine using the software simulation of the message facility. This virtual machine is running an operating system that supports the Coupling Facility, such as MVS.

The system administrator must define a user ID for each coupled guest that will run under z/VM. To define each user ID as a coupled guest, the CFUSER operand must be specified on the OPTION directory control statement for each virtual machine. The CFUSER operand defines the virtual machine as a coupled guest and allows it to connect to a CF Service Machine.

Any number of coupled guests can be defined, and each coupled guest can establish a coupled connection with up to eight different CF Service Machines concurrently.

The Message Facility Simulation

The Message Facility, which is an optional part of the channel subsystem, is the hardware that connects a processor complex to the Coupling Facility. CP simulates the message facility for the benefit of CF service machines and coupled guests. The simulated Message Facility allows a coupled guest to exchange information with a CF service machine, and when the hardware and z/VM are so capable, it allows CF service machines to exchange information with each other. From the coupled guest's point of view, the Message Facility appears as a set of virtual message devices. The CF service machine's view of the Message Facility is proprietary.

A Sample Guest Coupling Facility Setup

This sample illustrates creating one configuration of a z/VM sysplex environment. This sample is based on the configuration show in Figure 27 on page 110. Other configurations can be defined and implemented. Additional information about the tasks involved in setting up a Guest Coupling environment can be found in the section "Sysplex Setup Notes" on page 121.

To set up a Guest Coupling environment under z/VM, you need to perform the following tasks:

1. Plan the Sysplex Configuration
2. Create the CF Service Machines
3. Create the Coupled Guest Virtual Machines
4. Create the CF Service Machine Console User ID
5. Update the System Directory
6. Start the CF Service Machines on z/VM
7. Establish a Coupled Connection with the CF Service Machines
   a. Establish a Coupled Connection Manually
   b. Establish a Coupled Connection Automatically at Logon
8. Verify the Sysplex Configuration
9. IPL All of the MVS Images
10. Define the z/VM CF Service Machines to MVS
11. Establish Data Sharing Among MVS Virtual Machines
12. Modify the MVS Clock Definitions

Plan the Sysplex Configuration

Planning the resources for a sysplex configuration to run under z/VM is similar to planning for hardware and software requirements for a native production sysplex environment. However, the resources provided under z/VM are virtual resources (that is, software equivalents of the hardware and software required in a native sysplex environment).

The following steps will define a virtual sysplex environment on z/VM of the native parallel sysplex illustrated in Figure 27 on page 110. This sysplex consists of the following resources:

Two Coupling Facilities, one primary and one backup (optional)

For each Coupling Facility being simulated, there will be a CF Service Machine running under z/VM.

Three Coupled Guests

Each coupled guest will be simulated by a virtual machine running an independent MVS image under z/VM.

One CF Service Machine Console User ID

On a real Coupling Facility, a hardware console is available so that commands can be issued to manage the coupled environment. z/VM Guest Coupling Simulation support provides this same support by defining a user ID as a secondary console for each CF Service Machine. In our example, a single user ID will serve as the console for all of the CF Service Machines. However, you have the option of defining a separate user ID to serve as the console for each CF Service Machine.

Message Facility Simulation

This is equivalent to the fiber-optic channels that transfer information between the Coupling Facility and the processors running the MVS images in a physical sysplex environment. z/VM's simulated Message Facility transfers data between the CF service machines and the coupled guest virtual machines.

Create the CF Service Machines

Define two CF Service Machine user IDs by adding the following statements to the system directory:

```
*************************************************************************
* Primary Coupling Facility for Sysplex 1                               *
*************************************************************************
USER CFCC1    CFCC1  128M 256M G
  XAUTOLOG CFCONSOL CGUEST1 CGUEST2 CGUEST3
  OPTION CFVM TODENABLE
  ACCOUNT 14 CFVMACNT
  MACH ESA
  CONSOLE 0009 3215 T CFCONSOL

*************************************************************************
* Backup Coupling Facility for Sysplex 1   (Optional)                   *
*************************************************************************
USER CFCC2    CFCC2  128M 256M G
  XAUTOLOG CFCONSOL CGUEST1 CGUEST2 CGUEST3
  OPTION CFVM TODENABLE
  ACCOUNT 14 CFVMACNT
  MACH ESA
  CONSOLE 0009 3215 T CFCONSOL
```

Figure 29. Primary and Optional Backup CF Service Machine Sample Directory Entries

Statement	Explanation
USER	The actual host storage requirement for the CF Service Machine is determined much the same way it is for a real Coupling Facility. The only exception is that the CF Service Machine will not use XSTORE that is attached to the virtual machine. The 128M specified here is for a small sysplex.
XAUTOLOG	If you want to give privilege class G user IDs the ability to start the

CF Service Machine, specify the list of authorized class G user IDs on the XAUTOLOG or AUTOLOG statement.

OPTION The CFVM operand must be specified to inform CP that this virtual machine is a CF Service Machine. If CFVM is omitted, then no coupled guests will be able to establish a connection with this CF Service Machine. See the OPTION directory control statement in the *z/VM: CP Planning and Administration* for information about specifying the CFVM operand.

MACH The CF Service Machine must run in a ESA virtual machine.

CONSOLE If you need to issue CFCC commands to the CF service machine, you will need to set up secondary console support. Note that secondary console support will let you issue only CFCC commands themselves; z/VM will not let you use secondary console support to issue CP commands to the CF service machine. See "z/VM Guest Coupling Simulation Support Commands" on page 122 for an explanation of the CFCC commands.

Create the Coupled Guest Virtual Machines

Define three coupled guests (CGUEST1, CGUEST2, CGUEST3) in the system directory for the MVS images that will couple to the sysplex. Set up the three guest user IDs (CGUEST1, CGUEST2, CGUEST3) in the system directory so that they can IPL MVS. Use the sample template illustrated in Figure 30 as a guide, substituting the values 1, 2, and 3 for *x*.

```
*********************************************************************
* CGUEST1, CGUEST2, and CGUEST3, System Directory Template           *
*********************************************************************
USER CGUESTx password 64M 128M G
  OPTION CFUSER TODENABLE
  ACCOUNT 14 CFCGACNT
  MACH ESA
  CONSOLE 0009 3215 T
  SPOOL   000C 2540 READER *
  SPOOL   000D 2540 PUNCH  A
  SPOOL   000E 1403 A
*
* ADD THE REQUIRED RESOURCES FOR YOUR MVS SYSTEM HERE.
*
```

Figure 30. Template for CGUEST1, CGUEST2 and CGUEST3 Directory Entries

Statement	Explanation
OPTION	CFUSER must be specified to inform CP that this virtual machine is a coupled guest. If CFUSER is omitted, then this user will not be able to create a virtual message facility environment.
MACH	The coupled guest must run in a ESA virtual machine to use z/VM Guest Coupling Simulation Support.

Create the CF Service Machine Console User ID

Define a user ID in the system directory that will be used as the CF Service Machine console. Use the sample in Figure 31 on page 115 as a guide.

```
*************************************************************************
* Coupling Facility Hardware Console                                    *
*************************************************************************
USER CFCONSOL CFCONSOL 2M 2M G
  ACCOUNT 14 CFCGACNT
  MACH ESA
  CONSOLE 0009 3215 T
```

Figure 31. CF Service Machine Console User ID Sample Directory Entry

Note: The name of the console user ID in this example is CFCONSOL. It could be any name as long as it matches the name specified on the CONSOLE statement of the CF Service Machines (CFCC1 and CFCC2). See the CONSOLE statement in Figure 29 on page 113.

Update the System Directory

Verify that the changes that you have made to the directory file are correct. Then replace the old directory with the updated directory by entering the following:

```
directxa filename
```

Where *filename* is the file name of the directory file.

Note: The virtual machine that enters the DIRECTXA command must have write access to the volume that is to contain the new directory. If you create a directory that is to be written on the active z/VM system residence volume, your virtual machine's current directory entry must have write access to the volume that contains the current directory.

After the directory is updated, directory changes for a virtual machine currently logged on to the system do not take effect until the user logs off the system and then logs back on.

Start the CF Service Machines on z/VM

The first step in starting the sysplex environment on z/VM is to start the CF Service Machines.

1. Log on to the user ID that is defined as the console user ID for the CF Service Machine (that is, `LOGON CFCONSOL`). This allows you to see any errors that may occur during the startup of the CF Service Machines.
2. From the CFCONSOL user ID (or any user ID that has CF Service Machine XAUTOLOG authority), issue the following commands to start the CF Service Machines:

   ```
   XAUTOLOG CFCC1
   XAUTOLOG CFCC2
   ```

Note: CP will not allow you to LOGON to a user ID that is a CF Service Machine. The only way to start a CF Service Machine is with the CP XAUTOLOG or AUTOLOG commands. Since you cannot log on to a CF Service Machine, you must use the z/VM secondary console support facility (SCIF) to display CFCC initialization messages.

The following is a sample of messages that are displayed on the CF Service Machine console when CF Service Machine CFCC1 is started. The CF Service Machine is not ready to use until it has finished IPLing.

```
xautolog cfcc1
Command accepted
CFCC1   : HCPMFT2816I Loading message processor CFCC1 from the processor controller.
```

```
HCPQCS150A User CFCC1 has issued a CP read
CFCC1   : HCPMFT2817I Load completed from the processor controller.
HCPMFT2817I Now starting message processor CFCC1.
CFCC1   : 08:27:13 CF0009I Licensed Internal Code - Property of IBM
CFCC1   :                  Coupling facility control code
CFCC1   :                  (C) Copyright IBM Corp 1993,1994,1995,1996,1997
CFCC1   :                  All rights reserved.
CFCC1   :                  US Government Users Restricted Rights -
CFCC1   :                  Use, duplication or disclosure restricted
CFCC1   :                  by GSA ADP Schedule Contract with IBM Corp.
CFCC1   : 08:27:13 CF0280I CFCC Release 10.00, Service Level 00.57
CFCC1   :                  Built on 06/12/2000 at 10:19:00
CFCC1   :                  Code Load Features:
CFCC1   :                     Facility Operational Level: 10
CFCC1   : 08:27:13 CF0010I Coupling Facility is active with:
CFCC1   :                     1 CP
CFCC1   :                     0 CF Receiver Channels
CFCC1   :                     0 CF Sender Channels
CFCC1   :                     51.75 MB of allocatable central storage
CFCC1   :                     0 MB of allocatable expanded storage
CFCC1   : 08:27:13 CF0102I MODE is POWER SAVE.  Current status is VOLATILE.
CFCC1   :                  Power-Save feature is not installed.
```

Establish a Coupled Connection with the CF Service Machines

Establishing a coupled connection creates a message facility environment that allows the coupled guests and CF Service Machines to communicate with each other. There are two methods for establishing a coupled connection with a coupling facility:

A. Manually: Each coupled guest issues the DEFINE MSGPROC command

B. Automatically: The directory entry of each coupled guest contains a SPECIAL MSGPROC statement and the connection is established when the coupled guest logs on.

Both methods require that the CF Service Machine has already been successfully started.

A. Establishing a Coupled Connection Manually

1. Log on to each of the coupled guests (CGUEST1, CGUEST2, CGUEST3) from an available terminal.
2. (Optional) If you wish to run your sysplex with a date and time that is different from the z/VM system date and time, Refer “Change and Synchronize the Time-of-Day Clocks” on page 122.
3. Issue the following DEFINE MSGPROC commands from each of the coupled guest user IDs (CGUEST1, CGUEST2, CGUEST3) to establish a coupled connection:

   ```
   def msgp cfcc1 vdev 400

   def msgp cfcc2 vdev 500
   ```

   Note: In the sample DEFINE MSGPROC commands, 0400 and 0500 are the first of four consecutive device numbers that are not currently being used in the coupled guest’s I/O configuration.

   You will receive one of the following responses for each DEFINE MSGPROC command that successfully completes:

   ```
   HCPMFC2804I Message devices 0400–0403 defined and coupled to CFCC1

   HCPMFC2804I Message devices 0500–0503 defined and coupled to CFCC2
   ```

4. To verify that the coupled connections are established, issue the following command from each of the coupled guest user IDs (CGUEST1, CGUEST2, CGUEST3):

```
q msgdev
```

The response should be as follows:

```
MSGD 0400 MESSAGE PROCESSOR = CFCC1    SUBCHANNEL = 0000
MSGD 0401 MESSAGE PROCESSOR = CFCC1    SUBCHANNEL = 0001
MSGD 0402 MESSAGE PROCESSOR = CFCC1    SUBCHANNEL = 0002
MSGD 0403 MESSAGE PROCESSOR = CFCC1    SUBCHANNEL = 0003
MSGD 0500 MESSAGE PROCESSOR = CFCC2    SUBCHANNEL = 0010
MSGD 0501 MESSAGE PROCESSOR = CFCC2    SUBCHANNEL = 0011
MSGD 0502 MESSAGE PROCESSOR = CFCC2    SUBCHANNEL = 0012
MSGD 0503 MESSAGE PROCESSOR = CFCC2    SUBCHANNEL = 0013
```

The above responses verify that this coupled guest has eight message devices in two sets defined in its I/O configuration (0400 through 0403 and 0500 through 0503) and that they have a coupled connection with the CF Service Machines (CFCC1 and CFCC2).

Note: The values displayed in the `SUBCHANNEL = xxxx` fields are dependent on the virtual machine's current I/O configuration and might appear different than the values shown above.

B. Establishing a Coupled Connection Automatically at Logon

To establish a coupled connection automatically at logon, the system administrator must add SPECIAL MSGPROC statements to the system directory entry for each coupled guest (CGUEST1, CGUEST2 and CGUEST3).

```
SPECIAL 0400 MSGP CFCC1
SPECIAL 0500 MSGP CFCC2
```

Note: In the sample SPECIAL statements, 0400 and 0500 are the first of four consecutive device numbers that are not currently being used in the couple guest's I/O configuration.

- Refer to the *z/VM: CP Planning and Administration* for more information about the SPECIAL directory control statement.

Controlling Access to CF Service Machines

Defining a message processor by the SPECIAL MSGPROC directory statement allows the system administrator to restrict the CF Service Machines that a user is allowed to define with the DEFINE MSGPROC command. Once a SPECIAL MSGPROC directory statement is specified in the directory entry of a user, the user may only define message processors specified by SPECIAL MSGPROC statements.

Establish Connections Between the CF Service Machines

If the processor and z/VM are so capable, you can establish simulated Message Facility connections between the CF service machines themselves. You can establish these connections manually, or you can have CP establish them automatically as the CF service machines log on.

A. Establishing CF Connections Manually

1. Issue the following DEFINE CFLINK commands from any one of the coupled guest user IDs (CGUEST1, CGUEST2, or CGUEST3) or from a class A or class B user ID (such as OPERATOR):

```
DEFINE CFLINK CFCC1 VDEV 400 CFCC2
DEFINE CFLINK CFCC2 VDEV 400 CFCC1
```

Notes:

a. These commands set up two links: one from CFCC1 to CFCC2, and one in the other direction. Each link uses two consecutive device numbers in the sender's I/O configuration.

b. To use DEFINE CFLINK, the issuing user ID must be coupled to both involved CF service machines or must be class A or B.

The DEFINE CFLINK command simply produces the response "Command complete" when it completes successfully.

2. To verify that the link from CFCC1 to CFCC2 is established, enter the following command from any one of the coupled guest user IDs (CGUEST1, CGUEST2, CGUEST3) or from a class A or class B user ID (such as OPERATOR):

```
QUERY CFLINK NAME CFCC1
```

The response should be as follows:

```
MESSAGE PROCESSOR=00    USERID=CFCC1     PEERS=01
TYPE=SIMDEV  MODEL=001  MFG=IBM  PLANT=EN  SEQ=0000000CFCC1
DEVICES=0401 0400
CFS=E0 (PA=80) CFR=F0 (PA=80) S_RLINK=CFCC2
```

The above response verifies that CFCC1 has a set of message devices leading to CFCC2.

3. To verify that the link from CFCC2 to CFCC1 is established, enter QUERY CFLINK NAME CFCC2 and proceed similarly.

B. Establishing CF Connections Automatically

To establish a coupled connection automatically at logon, the system administrator must add SPECIAL MSGPROC statements to the system directory entry for each CF service machine (CFCC1 and CFCC2).

To CFCC1's directory, the system administrator would add this:

```
SPECIAL 0400 MSGP CFCC2
```

To CFCC2's directory, the system administrator would add this:

```
SPECIAL 0400 MSGP CFCC1
```

Note: In the sample SPECIAL statements, 0400 is the first of two consecutive device numbers that are not currently being used in the CF service machine's I/O configuration.

CF service machines log on one at a time (serially). When the first CF service machine logs on, its partner CF service machine will not yet be logged on and thus CP will not be able to create the link to the partner. When this happens, the CF secondary console machine will see this message:

```
CFCC1   : HCPMFZ2821I Some SPECIAL MSGP processing was deferred.
```

CP defers the creation of the link, creating the link when the partner finally does log on.

Controlling CF Access to Other CFs

If any SPECIAL MSGPROC statements appear in a CF service machine's directory entry, CP permits the CF service machine to couple to only the CF service machines named on those SPECIAL MSGPROC statements. On the other hand, if

no SPECIAL MSGPROC statements appear in a CF service machine's CP directory entry, the CF service machine may couple to any other CF service machine.

Verify the Sysplex Configuration

Verify that the coupled guests user IDs (CGUEST1, CGUEST2, and CGUEST3) are all coupled to the CF Service Machines (CFCC1 and CFCC2). Issue the following command from any of the couple guests:

```
query msgproc
```

This sample should produce the following response:

```
MESSAGE PROCESSOR=01    USERID=CFCC1     USERS=03
TYPE=SIMDEV  MODEL=001  MFG=IBM  PLANT=EN  SEQ=0000000CFCC1
DEVICES=0403 0402 0401 0400
CGS=C0 80 (PA=C0) CFR=80 C0 (PA=C0)
CG=CGUEST1    CGUEST2    CGUEST3

MESSAGE PROCESSOR=02    USERID=CFCC2     USERS=03
TYPE=SIMDEV  MODEL=001  MFG=IBM  PLANT=EN  SEQ=0000000CFCC2
DEVICES=0503 0502 0501 0500
CGS=C1 81 (PA=C0) CFR=80 C0 (PA=C0)
CG=CGUEST1    CGUEST2    CGUEST3
```

To ensure the sysplex configuration is correct, verify the following:

- The number of users connected to the CF Service Machine. The value for this sample should be:

  ```
  USERS=03
  ```

- The user IDs of the coupled guests that are currently coupled to the CF Service Machine (CFCC1 or CFCC2). The values for this sample should be:

  ```
  CG=CGUEST1  CGUEST2  CGUEST3
  ```

IPL All of the MVS Images

From each of the coupled guest user IDs (CGUEST1, CGUEST2, and CGUEST3), IPL your MVS system.

Define the z/VM CF Service Machines to MVS

Before MVS can use the z/VM Guest Coupling Simulation support, the administrative policy data in the Coupling Facility Resource Manager (CFRM) must be updated to define the CF Service Machines to MVS. The following is the JCL required to define our sysplex. The sysplex will use two z/VM CF Service Machines. The first CF Service Machine user ID is CFCC1 and the second one is CFCC2.

```
//IXCCFRMP JOB
//*****************************************************************
//*                                                               *
//* EXAMPLE JCL TO UPDATE THE ADMINISTRATIVE POLICY DATA IN THE    *
//* COUPLE DATA SET FOR CFRM (COUPLING FACILITY RESOURCE MANAGER)  *
//*                                                               *
//*****************************************************************
//STEP20   EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//SYSIN    DD   *

     DATA TYPE(CFRM) REPORT(YES)

     DEFINE POLICY NAME(POLICY1) REPLACE(YES)

        CF NAME(CFCC1)
           TYPE(SIMDEV)
```

```
        MFG(IBM)
        PLANT(EN)
        SEQUENCE(0000000CFCC1)
        PARTITION(0)
        CPCID(00)
        DUMPSPACE(2000)

CF NAME(CFCC2)
        TYPE(SIMDEV)
        MFG(IBM)
        PLANT(EN)
        SEQUENCE(0000000CFCC2)
        PARTITION(0)
        CPCID(00)
        DUMPSPACE(2000)

STRUCTURE NAME(LIST_01)
        SIZE(10000)
        INITSIZE(1000)
        PREFLIST(CFCC1,CFCC2)
        EXCLLIST(CACHE_01)

STRUCTURE NAME(CACHE_01)
        SIZE(1000)
        REBUILDPERCENT(25)
        PREFLIST(CFCC2,CFCC1)
        EXCLLIST(LIST_01)
```

The information for TYPE(), MFG(), PLANT(), and SEQUENCE() can be obtained by using the CP QUERY MSGPROC command (see “Verify the Sysplex Configuration” on page 119). The TYPE(). MFG(), and PLANT() information will always be the same when using z/VM Guest Coupling Simulation support. The SEQUENCE() value must be unique and is based on the user ID of the CF Service Machine. For more information about z/VM’s fabricated Node Descriptor, Refer to “Node Descriptor (ND) Support for the Message Processor” on page 124.

Establish Data Sharing Among MVS Virtual Machines

Working allegiance simulation must be used when running two or more MVS guests as part of a sysplex configuration using z/VM Guest Coupling Simulation Support. The WRKALLEG operand on the MINIOPT directory statement or the CP SET WRKALLEG command must be used for any minidisk containing CFRM datasets to maintain cross-system lock integrity (and thereby, data integrity) within the sysplex. Refer to the MINIOPT directory statement in the *z/VM: CP Planning and Administration* book and the SET WRKALLEG command in the the *z/VM: CP Commands and Utilities Reference*.

Also refer to the chapter entitled “DASD Sharing” in the *z/VM: CP Planning and Administration* book. That chapter contains information on sharing data on z/VM, such as sharing data among two or more MVS virtual machines.

Modify the MVS Clock Definitions

z/VM Guest Coupling Simulation Support does not support or simulate sysplex timers (ETR). When running in a z/VM environment, all virtual machine Time-of-Day (TOD) clocks are synchronized with the system TOD clock. This implicit synchronization allows all MVS images running in a sysplex on z/VM to have identically set TOD clocks without the use of an ETR. To inform MVS that clock synchronization will be done by z/VM, you must either modify the CLOCK00 member in SYS1.PARMLIB, or create a new CLOCK*xx* prefixed member for running

under z/VM. Otherwise, you will not be able to run in sysplex mode with multiple MVS images. Only monoplex mode will be supported.

Figure 32 shows a sample CLOCK member required to run a sysplex under z/VM.

```
TIMEZONE W.04.00.00
ETRMODE YES
ETRDELTA 10
ETRZONE YES
SIMETRID 00
```

Figure 32. SYS1.PARMLIB(CLOCK00) Sample Definition

Sysplex Setup Notes

This section provides additional information related to the tasks involved in setting up a z/VM sysplex environment. In addition, it describes the CP commands that are available to help you manage the simulated sysplex environment.

Notes on CF Service Machine Console Support

A real Coupling Facility has a hardware console that allows you to issue a limited set of regular mode CFCC commands to retrieve information about the coupling environment. These commands are listed below:

- CONFIGURE
- CP
- HELP
- MODE
- RIDEOUT
- SHUTDOWN
- TIMEZONE
- DISPLAY MODE
- DISPLAY CHPIDS
- DISPLAY RESOURCES
- DISPLAY LEVEL
- DISPLAY RIDEOUT
- DISPLAY TIMEZONE

z/VM Guest Coupling Simulation Support provides console support for the CF Service Machine through z/VM's existing secondary console support. The system administrator has the option to define a user ID and specify that user ID on the CONSOLE statement in the CF Service Machine's system directory entry. Doing so will direct all console output from the disconnected CF Service Machine to the console of the user ID that is specified.

The system administrator can use the PROP facility (see the *z/VM: CP Planning and Administration*) or a similar function to allow unattended console processing of the CF Service Machine by the Secondary userid.

Note: After SHUTDOWN is issued for a CF Service Machine which has been successfully started, coupled guests will be able to establish a connection, but the Coupling Facility will not be operational until a RESTART MSGPROC command is issued.

Change and Synchronize the Time-of-Day Clocks

z/VM allows you to simulate running your sysplex environment with a date and time that is different from the system date and time with the CP SET VTOD command.

In a z/VM environment, all virtual machine Time-of-Day (TOD) clocks are synchronized with the system TOD clock when they log on. If you want to run the virtual machines in a sysplex with a TOD setting that is different than the current system TOD setting, you must change and synchronize all the virtual machine TOD clocks within the sysplex.

To change the TOD clocks for the coupled guests, prior to starting your sysplex issue the following sequence of commands:

1. From the CGUEST1 user ID, issue the DEFINE MSGPROC commands to add CFCC1 and CFCC2 to your I/O configuration. Because CGUEST1 has only Class G privileges, the SET VTOD command in the next step requires that the CF Service Machine whose date and time that you want to set be in the I/O configuration of the coupled guest that is issuing the SET VTOD command.

   ```
   def msgp cfcc1 vdev 400

   def msgp cfcc2 vdev 500
   ```

2. From the CGUEST1 user ID, issue the following commands, substituting the appropriate values for the date and time.

   Set the TOD clock:

   ```
   set vtod date mm/dd/yyyy time hh:mm:ss
   ```

   Set the TOD clocks for the CF Service Machines:

   ```
   set vtod msgp cfcc1

   set vtod msgp cfcc2
   ```

3. Issue the following command from the CGUEST2 user ID to synchronize the CGUEST2 TOD clock with the sysplex:

   ```
   set vtod fromuser cfcc1
   ```

4. Issue the following command from the CGUEST3 user ID to synchronize the CGUEST3 TOD clock with the sysplex:

   ```
   set vtod fromuser cfcc1
   ```

z/VM Guest Coupling Simulation Support Commands

The following CP commands help you manage the simulated sysplex environment:

DEFINE MSGPROC
Creates a message facility environment for the coupled guest and establishes a connection from the coupled guest to a specific CF Service Machine. This specification is equivalent to defining message devices, control devices, and message paths on a processor that supports the message facility.

DETACH MSGPROC
Selectively detach a coupled guest from a CF Service Machine without eliminating the entire virtual message facility.

SET MSGFACIL OFF
Eliminates the entire virtual message facility for a coupled guest.

QUERY VIRTUAL MSGDEVICE
Displays a list and status of all message devices currently defined in a coupled guest virtual machine.

DEFINE CFLINK
Creates a simulated Message Facility link from one CF service machine to another.

SET CFLINK
Logically enables or disables CF-to-CF links without physically removing the CHPIDS from the virtual machine's I/O configuration.

QUERY CFLINKS
Displays status information about the simulated Message Facility in a CF service machine.

QUERY VIRTUAL ALL
Displays the size of a virtual machine's storage, status of all virtual devices and processors. The information displayed by this command includes the information displayed by the QUERY VIRTUAL MSGDEVICE command.

QUERY VIRTUAL MSGPROC
Displays a list and the status of all message processors currently defined in a coupled guest's I/O configuration. The information displayed consists of the following:

- Message processor number
- CF Service Machine user ID
- Total count of coupled guests
- z/VM's fabricated Node Descriptor (see "Node Descriptor (ND) Support for the Message Processor" on page 124)
- User's message devices associated with the message processor
- Sender CHPIDS used by the guest to communicate with the CF service machine.
- CF Service Machine Receiver CHPIDS
- User ID's of all coupled guests

RESTART MSGPROC
Reload and start a specific CF Service Machine. All structures within the coupling facility are lost when this command is issued. Caution should be exercised when issuing this command since it will affect all coupled guests connected to the message processor.

SET VTOD
Sets the issuing virtual machine's Time-of-Day (TOD) clock. This command can also be used to synchronize a CF Service Machine's TOD clock with the issuing user's TOD clock.

QUERY VTOD
Queries the issuing virtual machine's Time-of-Day (TOD) clock. This command can also list the set of virtual machines whose TOD clock is synchronized with the issuer's.

Refer to the *z/VM: CP Commands and Utilities Reference* for the a complete description, syntax, usage notes, and responses for these commands.

Node Descriptor (ND) Support for the Message Processor

A node descriptor (ND) is a thirty two byte field that provides the description of a physical device attached to a serial channel. Every physical processor that can attach to a serial channel has it's own worldwide unique ID. This allows software and hardware to distinguish one 3390 DASD from another 3390.

Since z/VM Guest Coupling Simulation Support simulates a real Coupling Facility, the software must create a unique ND for every CF Service Machine to be used. z/VM dynamically creates an architecturally correct unique ND when the message facility environment is created in the CF Service Machine. The method used to create the ND is documented so the user can generate the administration policy for the CFRM in MVS. This will allow MVS's CFRM policy generator to be used and there is no need for the user to specify an ND. This is exactly how it works with the real hardware.

The following is the information that is contained in the ND that z/VM creates:

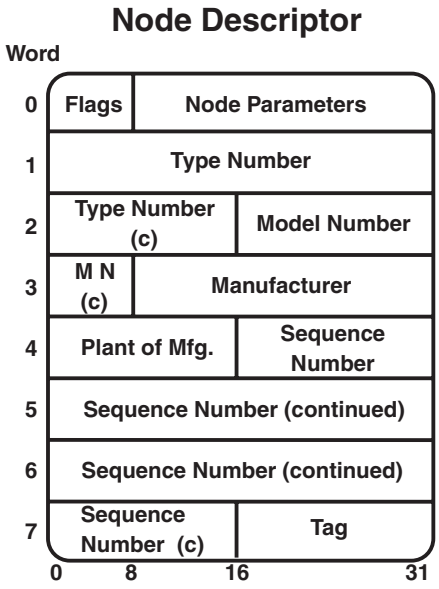

Figure 33. Node Descriptor Format

Field	z/VM Simulated ND Data
Flags	Byte 0 of Word 0 describes the manner in which selected fields of the ND are to be interpreted.

Bits	Description (hex '18')
0-2	(000) Node is valid.
3	(1) Indicates that this is a CPC type node.
4	(1) Indicates that this interface is not accessed by means of a link. It is always on for ND objects.
5	(0) Indicates that the interface has an associated CHPID.
6-7	(00) Reserved

Node PARMS Bytes 1-3 of word 0 contain information about the node.

Byte	Description (hex '000000')
1	(00) Specifies the side and partitioning information for the specified interface. The following information is reflected: • Node is on machine side 0 • CPC can't be physically partitioned into more than one machine side.
2	(00) Specifies that the interface is an unspecified class.
3	(00) CPC Image ID of the Coupling Facility.

Type Number Word 1 and bytes 0-1 of word 2 is the six character device type in the worldwide unique ID.

SIMDEV Specifies this is a z/VM simulated message device type.

Model Number

Bytes 2-3 of word 2 and byte 0 of word 3 will contain the three character model number in the worldwide unique ID.

001 Three bytes of EBCDIC characters for the simulated model number.

Manufacturer Bytes 1-3 of word 3 contains a three byte code that identifies the manufacturer in the worldwide unique ID.

IBM Indicate IBM as the manufacturer.

Plant Bytes 0-1 of word 4 contain a two character EBCDIC plant code that identifies the plant of manufacturer in the worldwide unique ID.

EN Indicate Endicott as the manufacturing plant.

Sequence # Bytes 2-3 of word 4, words 5-6, and bytes 0-1 of word 7 contain the right justified serial number in EBCDIC.

This field will make the ND of the CF Service Machine unique in z/VM's implementation. The 12 byte field will contain the CF Service Machine user ID right justified. All leading unused characters will be padded with EBCDIC zeros. The following sequence number will be created if a CF Service Machine of CFCC1 is specified:

0000000CFCC1
Dynamically created sequence number.

The only valid characters for the sequence number are EBCDIC 0-9 or uppercase A-Z. Any character in the CF Service Machine user ID that is not EBCDIC 0-9 or uppercase A-Z is replaced with an EBCDIC zero. For example, the following sequence number will be created if a CF Service Machine user ID of CFCC-1 is specified:

000000CFCC01
Dynamically created sequence number with a user ID that contain characters not valid for a sequence number field (@ # $ _ -).

Note: It is recommended that the CF Service Machine user ID not contain any special characters.

Tag The last two bytes of the ND contains an interface identifier that uniquely identifies the CPC in which the Coupling Facility is located. The following CPC ID will be reflected:

0000 Interface ID

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 all 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 grant you any license to these patents. You can send license inquiries, in writing, to:

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

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.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, New York 12601-5400
U.S.A.
Attention: Information Request

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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 may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

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

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

Java is a trademark of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Glossary

For a list of z/VM terms and their definitions, see *z/VM: Glossary*.

The z/VM glossary is also available through the online z/VM HELP Facility. For example, to display the definition of the term “dedicated device”, issue the following HELP command:

```
help glossary dedicated device
```

While you are in the glossary help file, you can do additional searches:

- To display the definition of a new term, type a new HELP command on the command line:

  ```
  help glossary newterm
  ```

  This command opens a new help file inside the previous help file. You can repeat this process many times. The status area in the lower right corner of the screen shows how many help files you have open. To close the current file, press the Quit key (PF3/F3). To exit from the HELP Facility, press the Return key (PF4/F4).
- To search for a word, phrase, or character string, type it on the command line and press the Clocate key (PF5/F5). To find other occurrences, press the key multiple times.

  The Clocate function searches from the current location to the end of the file. It does not wrap. To search the whole file, press the Top key (PF2/F2) to go to the top of the file before using Clocate.

Bibliography

See the following publications for additional information about z/VM. For abstracts of the z/VM publications, see *z/VM: General Information*.

Where to Get z/VM Information

z/VM product information is available from the following sources:

- z/VM Information Center at publib.boulder.ibm.com/infocenter/zvm/v6r1/index.jsp
- z/VM Internet Library at www.ibm.com/eserver/zseries/zvm/library/
- IBM Publications Center at www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss
- *IBM Online Library: z/VM Collection on DVD*, SK5T-7054

z/VM Base Library

Overview

- *z/VM: General Information*, GC24-6193
- *z/VM: Glossary*, GC24-6195
- *z/VM: License Information*, GC24-6200

Installation, Migration, and Service

- *z/VM: Guide for Automated Installation and Service*, GC24-6197
- *z/VM: Migration Guide*, GC24-6201
- *z/VM: Service Guide*, GC24-6232
- *z/VM: VMSES/E Introduction and Reference*, GC24-6243

Planning and Administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6167
- *z/VM: CMS Planning and Administration*, SC24-6171
- *z/VM: Connectivity*, SC24-6174
- *z/VM: CP Planning and Administration*, SC24-6178
- *z/VM: Getting Started with Linux on System z*, SC24-6194
- *z/VM: Group Control System*, SC24-6196
- *z/VM: I/O Configuration*, SC24-6198
- *z/VM: Running Guest Operating Systems*, SC24-6228
- *z/VM: Saved Segments Planning and Administration*, SC24-6229
- *z/VM: Secure Configuration Guide*, SC24-6230
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6236
- *z/VM: TCP/IP Planning and Customization*, SC24-6238
- *z/OS and z/VM: Hardware Configuration Manager User's Guide*, SC33-7989

Customization and Tuning

- *z/VM: CP Exit Customization*, SC24-6176
- *z/VM: Performance*, SC24-6208

Operation and Use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6166
- *z/VM: CMS Pipelines Reference*, SC24-6169
- *z/VM: CMS Pipelines User's Guide*, SC24-6170
- *z/VM: CMS Primer*, SC24-6172
- *z/VM: CMS User's Guide*, SC24-6173
- *z/VM: CP Commands and Utilities Reference*, SC24-6175
- *z/VM: System Operation*, SC24-6233
- *z/VM: TCP/IP User's Guide*, SC24-6240
- *z/VM: Virtual Machine Operation*, SC24-6241
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6244
- *z/VM: XEDIT User's Guide*, SC24-6245
- *CMS/TSO Pipelines: Author's Edition*, SL26-0018

Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6162
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6163
- *z/VM: CMS Application Multitasking*, SC24-6164
- *z/VM: CMS Callable Services Reference*, SC24-6165
- *z/VM: CMS Macros and Functions Reference*, SC24-6168

- *z/VM: CP Programming Services*, SC24-6179
- *z/VM: CPI Communications User's Guide*, SC24-6180
- *z/VM: Enterprise Systems Architecture/ Extended Configuration Principles of Operation*, SC24-6192
- *z/VM: Language Environment User's Guide*, SC24-6199
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6202
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6203
- *z/VM: OpenExtensions Commands Reference*, SC24-6204
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6205
- *z/VM: OpenExtensions User's Guide*, SC24-6206
- *z/VM: Program Management Binder for CMS*, SC24-6211
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6220
- *z/VM: REXX/VM Reference*, SC24-6221
- *z/VM: REXX/VM User's Guide*, SC24-6222
- *z/VM: Systems Management Application Programming*, SC24-6234
- *z/VM: TCP/IP Programmer's Reference*, SC24-6239
- *Common Programming Interface Communications Reference*, SC26-4399
- *Common Programming Interface Resource Recovery Reference*, SC31-6821
- *z/OS: IBM Tivoli Directory Server Plug-in Reference for z/OS*, SA76-0148
- *z/OS: Language Environment Concepts Guide*, SA22-7567
- *z/OS: Language Environment Debugging Guide*, GA22-7560
- *z/OS: Language Environment Programming Guide*, SA22-7561
- *z/OS: Language Environment Programming Reference*, SA22-7562
- *z/OS: Language Environment Run-Time Messages*, SA22-7566
- *z/OS: Language Environment Writing ILC Applications*, SA22-7563
- *z/OS MVS Program Management: Advanced Facilities*, SA22-7644
- *z/OS MVS Program Management: User's Guide and Reference*, SA22-7643

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6161
- *z/VM: CP Messages and Codes*, GC24-6177
- *z/VM: Diagnosis Guide*, GC24-6187
- *z/VM: Dump Viewing Facility*, GC24-6191
- *z/VM: Other Components Messages and Codes*, GC24-6207
- *z/VM: TCP/IP Diagnosis Guide*, GC24-6235
- *z/VM: TCP/IP Messages and Codes*, GC24-6237
- *z/VM: VM Dump Tool*, GC24-6242
- *z/OS and z/VM: Hardware Configuration Definition Messages*, SC33-7986

z/VM Facilities and Features

Data Facility Storage Management Subsystem for VM

- *z/VM: DFSMS/VM Customization*, SC24-6181
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6182
- *z/VM: DFSMS/VM Messages and Codes*, GC24-6183
- *z/VM: DFSMS/VM Planning Guide*, SC24-6184
- *z/VM: DFSMS/VM Removable Media Services*, SC24-6185
- *z/VM: DFSMS/VM Storage Administration*, SC24-6186

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6188
- *z/VM: Directory Maintenance Facility Messages*, GC24-6189
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6190

Open Systems Adapter/Support Facility

- *System z10, System z9 and eServer zSeries: Open Systems Adapter-Express Customer's Guide and Reference*, SA22-7935
- *System z9 and eServer zSeries 890 and 990: Open Systems Adapter-Express Integrated Console Controller User's Guide*, SA22-7990

- *System z: Open Systems Adapter-Express Integrated Console Controller 3215 Support*, SA23-2247

Performance Toolkit for VM™

- *z/VM: Performance Toolkit Guide*, SC24-6209
- *z/VM: Performance Toolkit Reference*, SC24-6210

RACF® Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6212
- *z/VM: RACF Security Server Command Language Reference*, SC24-6213
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6214
- *z/VM: RACF Security Server General User's Guide*, SC24-6215
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6216
- *z/VM: RACF Security Server Messages and Codes*, GC24-6217
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6218
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6219
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6231

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6223
- *z/VM: RSCS Networking Exit Customization*, SC24-6224
- *z/VM: RSCS Networking Messages and Codes*, GC24-6225
- *z/VM: RSCS Networking Operation and Use*, SC24-6226
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6227
- *Network Job Entry: Formats and Protocols*, SA22-7539

Prerequisite Products

Device Support Facilities

- *Device Support Facilities: User's Guide and Reference*, GC35-0033

Environmental Record Editing and Printing Program

- *Environmental Record Editing and Printing Program (EREP): Reference*, GC35-0152
- *Environmental Record Editing and Printing Program (EREP): User's Guide*, GC35-0151

Index

Special characters

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

X

Z

Program Number: 5741-A07

Printed in USA

SC24-6228-00