z/VM
7.4

Virtual Machine Operation

IBM

SC24-6334-74

Note:

Before you use this information and the product it supports, read the information in "Notices" on page 145.

This edition applies to version 7, release 4 of IBM® z/VM® (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2024-09-18

Contents

Figures

Tables

About This Document

This document explains how to use z/VM to operate virtual machines. It is intended to help you operate z/VM, including:

- Logging on to a virtual machine and load (IPL) one of several operating systems in that virtual machine
- Managing the storage, processor, and I/O resources of a virtual machine
- Using z/VM facilities to test programs running in a virtual machine.

Intended Audience

This document is intended for anyone who is using z/VM to run an operating system in a virtual machine. Although most of the CP commands that this document describes are class G (general use) commands, it describes and notes commands of other classes.

This document assumes that you have a general idea of what z/VM does and what a virtual machine is.

Before you read this document, you should know how to run one of the following operating systems either on a real processor or in a virtual machine:

- z/OS®
- z/VM
- VSE/ESA

For information about running the Linux® operating system in a virtual machine, see *z/VM: Getting Started with Linux on IBM Z*.

Where to Find More Information

For more information about z/VM functions, see the documents listed in the "Bibliography" on page 149.

Links to Other Documents and Websites

The PDF version of this document contains links to other documents and websites. A link from this document to another document works only when both documents are in the same directory or database, and a link to a website works only if you have access to the Internet. A document link is to a specific edition. If a new edition of a linked document has been published since the publication of this document, the linked document might not be the latest edition.

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. See How to send feedback to IBM for additional information.

Summary of Changes for z/VM: Virtual Machine Operation

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line (|) to the left of the change.

SC24-6334-74, z/VM 7.4 (September 2024)

This edition supports the general availability of z/VM 7.4. Note that the publication number suffix (-74) indicates the z/VM release to which this edition applies.

SC24-6334-73, z/VM 7.3 (December 2023)

This edition includes terminology, maintenance, and editorial changes.

SC24-6334-73, z/VM 7.3 (September 2022)

This edition supports the general availability of z/VM 7.3. Note that the publication number suffix (-73) indicates the z/VM release to which this edition applies.

SC24-6334-03, z/VM 7.2 (May 2022)

This edition includes changes to support product changes provided or announced after the general availability of z/VM 7.2.

[VM66532] z/VM Support for IBM z16

With the PTF for APAR VM66532, z/VM 7.1 and 7.2 provide support to enable guests to exploit function on IBM z16™. The following support is included:

- Breaking-event-address register (BEAR) enhancement facility, which facilitates the debug of wild branches.
- Reset DAT protection facility, which provides a more efficient way to disable DAT protection, such as during copy-on-write or page-change tracking operations.

LPSWEY and RDP are added to the assembly language instructions that can be traced by using TRACE MNEMONIC1. The following topics are updated:

- "Events That You Can Trace" on page 125
- "Using TRACE MNEMONIC1" on page 138

SC24-6334-02, z/VM 7.2 (March 2021)

This edition includes changes to support product changes provided or announced after the general availability of z/VM 7.2.

[VM66173] 4 TB Real Memory Support

z/VM APAR VM66173 delivers support for up to 4 TB of real memory, allowing z/VM systems to address a full 4 TB of first-level (real) memory, doubling the previous supported limit of 2 TB. With advanced memory management capabilities available in the z/VM product, clients now have the ability to run workloads that exceed 4 TB of virtual memory across all hosted guest systems, depending on workload characteristics. In conjunction with z/VM support for 80 processors, IBM Z® and LinuxOne servers can now host even more work in a single z/VM partition, or across multiple z/VM partitions on one system.

APAR VM66173 also delivers various system command updates, such as automatic STANDBY memory for guests and an enhancement to the CP DEFINE STORAGE command.

With automatic STANDBY memory for guests, a system administrator can code a more generic DEFINE STORAGE command that does not need to be updated every time the guest's directory entry storage size changes.

The following information has been updated for this support:

- "Step 2. Check Your Virtual Machine's Storage Size" on page 19

[VM66201, VM66425] z/Architecture Extended Configuration (z/XC) support

With the PTFs for APARs VM66201 (CP) and VM66425 (CMS), z/Architecture® Extended Configuration (z/XC) support is provided. CMS applications that run in z/Architecture can use multiple address spaces. A z/XC guest can use VM data spaces with z/Architecture in the same way that an ESA/XC guest can use VM data spaces with Enterprise Systems Architecture. z/Architecture CMS (z/CMS) can use VM data spaces to access Shared File System (SFS) Directory Control (DIRCONTROL) directories. Programs can use z/Architecture instructions and registers (within the limits of z/CMS support) and can use VM data spaces in the same CMS session. For more information, see *z/VM: z/Architecture Extended Configuration (z/XC) Principles of Operation*.

Information in the following topics is updated:

- "Before You Begin to Start Your Virtual Machine" on page 17
- "Step 3. Check Your Virtual Machine's Architecture" on page 19
- "Example: Dumping Current PSW" on page 110
- "Example: Dumping Old and New PSWs" on page 110
- "Example: Dumping General Registers" on page 110
- "Example: Dumping Control Registers" on page 110
- "Displaying Information from Your Virtual Machine's Storage" on page 111
- "Altering the Contents of Your Virtual Machine's Storage" on page 113

SC24-6334-01, z/VM 7.2 (September 2020)

This edition supports the general availability of z/VM 7.2.

Updates reflect the removal of KANJI language files from base z/VM components. The only currently supported languages are American English and uppercase English.

Chapter 1. Using a Virtual Machine–Introduction

This chapter describes:

- Differences between running an operating system on a real processor and running an operating system in a virtual machine
- Using your virtual machine operator's console
- Permanently defining your virtual machine to the Control Program (CP)
- Starting your virtual machine
- Operating your virtual machine
- Temporarily defining and changing virtual machine resources
- Using a virtual machine to test and debug programs.

Ideally, running an operating system in a virtual machine should be the same as running an operating system on a real processor. In many ways it is. When you use a virtual machine to run an operating system, you still communicate with that operating system using its command language. In response to your commands, the operating system performs the same functions as it would if it were running on a real processor. In fact, with few exceptions, the operating system you run manages virtual machine storage, processors, and I/O devices the same way it would manage real storage, processors, and I/O devices.

The difference between running an operating system on a real processor and running it in a virtual machine is *not* how you communicate with that operating system, but rather that, in addition to communicating with the operating system you are running in your virtual machine, you must also communicate with the z/VM Control Program (CP). The purpose of this book is to describe how and when you might need to communicate with CP.

Using Your Virtual Machine Operator's Console

Before you use a virtual machine, you must know how to use a virtual machine operator's console to communicate with CP and with the operating system you are going to run in your virtual machine. In particular, you must know how to enter two types of commands: CP commands and the commands of the operating system you run in your virtual machine. This is the subject of Chapter 2, “Using a Virtual Machine Operator's Console,” on page 3.

Permanently Defining Your Virtual Machine to CP

To CP, your virtual machine starts out as a number of descriptive statements contained in a file called the z/VM user directory. Your virtual machine's entry in the user directory describes to CP:

- Your user identification (also called a user ID or logon ID) and password for logging on to z/VM
- The CP command privilege classes that determine what CP commands you can enter
- The initial I/O configuration, processor configuration, storage size, and architecture mode of your virtual machine
- Your virtual machine's initial scheduling share (that is, the amount of processor time your virtual machine receives)
- Whether CP handles missing interruptions for your virtual machine.

Creating the user directory is part of the system installation process; you can also update it any time after system installation. To change your virtual machine permanently, you must contact the person who updates the user directory. Information on creating and updating the user directory is contained in the *z/VM: CP Planning and Administration*.

Starting Your Virtual Machine

Once your virtual machine is described to CP in the user directory, you can log on to z/VM. When you log on, CP creates your virtual machine as it is defined in the user directory. When you are ready to log on, see Chapter 3, "Starting Your Virtual Machine," on page 17.

Operating Your Virtual Machine

After you initialize a virtual machine and the operating system you want to run, you control that operating system the same way you would if it were running on a real processor. Also, operating your virtual machine may include starting and stopping it, simulating real hardware functions (such as performing a system restart), and logging it off. For information on operating your virtual machine, see Chapter 4, "Operating Virtual Machines," on page 27.

Temporarily Defining or Changing Virtual Machine Resources

Although you must change your virtual machine's user directory entry to change your virtual machine permanently, you can use CP commands to temporarily define and change virtual machine storage, I/O, and processor resources. These changes remain in effect until you enter the LOGOFF command to end your current virtual machine session.

Chapter 5, "Managing Virtual Processors," on page 43 explains how to define and use a virtual machine multiprocessor configuration.

Chapter 6, "Managing Virtual Machine I/O and Storage Devices," on page 49 explains how to manage virtual machine I/O devices.

Chapter 7, "Using Spooled Devices to Print, Punch, and Read Information," on page 75 explains how to print, punch, and read information using a type of I/O device unique to VM operating systems–the spooled unit record device.

Testing and Debugging Programs

If you are using a virtual machine to test and debug programs, you can use any testing and debugging facilities in the operating system you are running; you can also use a number of CP commands. Chapter 8, "Dumping, Displaying, Altering Virtual Machine Storage," on page 109 describes how to use the DUMP, DISPLAY, and STORE commands. Chapter 9, "Tracing Programs in Your Virtual Machine," on page 115 and Chapter 10, "Creating Trace Traps (Examples)," on page 125 describe how to use the TRACE command.

Chapter 2. Using a Virtual Machine Operator's Console

When you run an operating system in a virtual machine, you are the virtual machine operator. You operate the virtual machine and the operating system in your virtual machine. Your *virtual machine operator's console* is the display device at which you log on z/VM. (Your virtual machine operator's console can also be referred to as a virtual console, an operator's console, or just a console. All of these terms refer to the display device at which you log on z/VM.)

By entering commands from your virtual machine operator's console, you can communicate with CP and with the operating system executing in your virtual machine. To communicate with CP, enter CP commands. To communicate with the operating system running in your virtual machine, enter the commands of that operating system.

Note: Many installations have two consoles for the virtual machine operator: a virtual machine console to communicate with CP, and a system console to communicate with the operating system running in the virtual machine. When this book refers to your virtual machine operator's console, it is referring to the console you use to communicate with CP.

This topic contains descriptions about:

- Using your virtual machine operator's console to communicate with the z/VM Control Program (CP), with operating systems executing in virtual machines, and with other users
- Interpreting information formatted by CP on a 3270 display screen
- Interpreting the following or equivalent CP screen status indicators:
  - `CP READ`
  - `VM READ`
  - `RUNNING`
  - `MORE...`
  - `HOLDING`
  - `NOT ACCEPTED`
- Entering CP commands
- Using your virtual machine's program function (PF) keys
- Sending messages to the z/VM system operator and other users
- Finding out what commands you are authorized to enter
- Finding out what user class modification allows you to do
- Activating, deactivating, and querying the protected application facility.

CP and Virtual Machine Command Environments

As just mentioned, when you use a virtual machine, you need to communicate with two operating systems: CP and the operating system you run in your virtual machine. Therefore, when you use a virtual machine, you are in one of two command environments:

- **The CP environment.** In this environment, CP handles commands you enter and controls the display of information on your 3270 display screen.

- **The virtual machine environment.** In this environment, the operating system you run handles commands you enter and controls the display of information on your 3270 display screen.

Before you load (IPL) an operating system in your virtual machine, you are *always* in the CP command environment. After you load an operating system in your virtual machine, you are in the virtual machine command environment *unless* you specifically return to the CP command environment to enter CP commands.

Screen Format of a 3270 Display Device

This section describes how CP displays information at your virtual machine operator's console when you are in the CP command environment.

CP divides your display screen into three areas (Figure 1 on page 4 shows the locations of these areas):

- The *output display area* consists of all but the last two lines of the display screen. Messages and responses from both the virtual machine and CP appear in this area, as well as the redisplay of commands you enter. You cannot enter commands or data in the output display area.

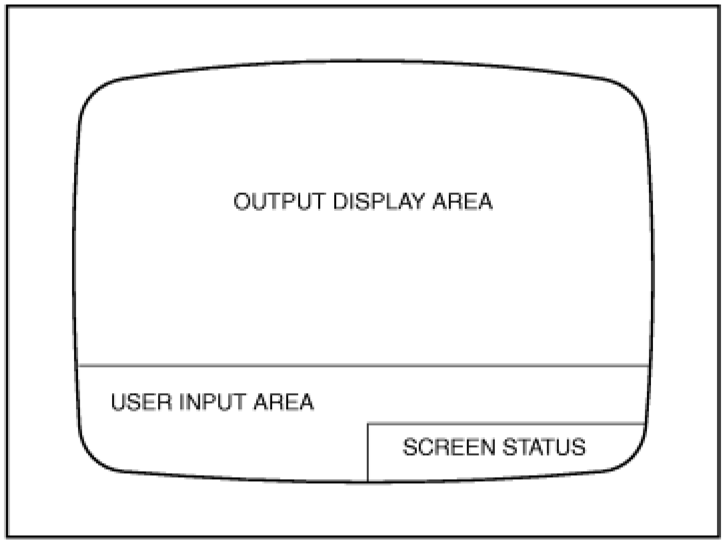


Figure 1. Screen Format of a 3270 Display

- The *user input area* consists of the last two lines of the screen, except the rightmost 21 character positions of the last line. Enter data and commands on these two lines. You can use cursor control keys, the DELETE key, the INSERT key, and logical text editing characters to alter data in this area. After you press ENTER, CP redisplays the data you enter in the output display area.
- The *screen status area* consists of the rightmost 21 character positions of the last line on your screen. Screen status indicators appear in this area. These status indicators help you determine if you are currently running in the CP or virtual machine environment. You cannot enter commands or data in this area.

In addition to these three areas, the screen on some 3270 display devices includes an operator information area, which appears below the last line of the user input area. This area provides you with system status indicators. (You cannot enter commands or data in this area.) The explanations for these status indicators are contained in a *Problem Determination Guide*.

CP Status Indicators

If you are in the CP environment, CP displays one of the following or equivalent status indicators:

CP READ
: CP issued a read request to your display and is waiting for you to enter something before it continues processing.

VM READ
A response is required from you before any more processing can be done.

RUNNING
CP is ready for you to enter your next CP command or is processing a previously entered command.

MORE...
The output display area is full and CP has more lines to display. The data currently on the screen will be displayed for 60 seconds. When you first log on to the system, CP sets the number of seconds the `MORE ...` is displayed.

To get to the next screen, press CLEAR or PA2. (PA2 clears your screen except for the input area.)

To keep the current information on the screen, press ENTER. The `HOLDING` indicator then appears in the status area. (Note that if, for some reason, someone uses the FORCE command to force you off the system, `HOLDING` does not appear. Rather, a one-minute timer is set and the `MORE ...` indicator remains.)

If you do not respond to the `MORE ...` indicator within the elapsed time, CP automatically displays the next screen. (Your display device beeps 10 seconds (default) before CP displays the next screen.)

The TERMINAL MORE command lets you change the number of seconds that elapse between the time when CP:

- issues the `MORE ...` state and sounds the terminal alarm
- sounds the alarm and clears the screen.

Note: If your terminal is attached to the system through a VTAM® service machine (VSM), the TERMINAL MORE command and the TERMINAL HOLD OFF command is supported within ACF/VTAM version 3 release 4.1. For more information about the TERMINAL MORE command, see *z/VM: CP Commands and Utilities Reference*.

HOLDING
You pressed ENTER in response to a `MORE ...` status indicator, or your display screen contains priority messages from CP. To get to the next screen, press CLEAR or PA2. (PA2 clears your screen except for the input area.)

The TERMINAL HOLD command lets you control whether CP displays the `HOLDING` status when the terminal screen is full and highlighted output appears on the screen. For more information about the TERMINAL HOLD command, see *z/VM: CP Commands and Utilities Reference*.

NOT ACCEPTED
Previous input was not processed. CP locks the keyboard for about 3 seconds while it displays `NOT ACCEPTED`, then reverts to its previous status.

CP status indicators do not usually appear after you load (IPL) an operating system in your virtual machine. The operating system you IPLed usually controls the display of screen status indicators. (If after you IPL an operating system you return to the CP environment, CP again controls the display of screen status indicators.) One of the manuals of the operating system you run in your virtual machine should describe the information, if any, that will be displayed in the screen status area.

The CP screen status indicators can be changed using the STATUS statement in the logo configuration file. Your system administrator assigns the status indicators that are displayed on the screen. For more information about changing CP status indicators, see *z/VM: CP Planning and Administration*.

CP Command General Format

You can enter CP commands and their operands in uppercase, lowercase, or a combination of both. You must, however, use one or more blanks to separate a CP command from its operands and the operands from each other. All of the following examples are valid ways to enter the QUERY VIRTUAL ALL command:

```
QUERY VIRTUAL ALL
Query Virtual All
query virtual all
```

Note: You can use abbreviations for most CP commands; for example, you can use Q for QUERY, B for BEGIN. Unless otherwise noted, this book does not use abbreviations. To find out what the abbreviations are for a particular CP command, see *z/VM: CP Commands and Utilities Reference*.

Entering Commands

When you run an operating system in a virtual machine, you need commands to communicate with both operating systems, CP and what is IPLed in the virtual machine. You will need to use:

- CP commands to communicate with CP. For example, you must use CP commands to start and stop your virtual machine, to temporarily change your virtual machine's resources, and to use CP testing and debugging facilities.
- The command language of the operating system you are running to communicate with that operating system. If you are running MVS/ESA, you need to use MVS/ESA commands. If you are running VSE, you need to use VSE commands.

Because you must enter commands to two different operating systems, you need to make sure that the correct operating system handles commands you enter. If you want to enter a CP command, make sure that CP handles the command. If you are running MVS/ESA in your virtual machine and you want to enter an MVS/ESA command, make sure that MVS/ESA handles the command.

Finding Out What Commands You Are Authorized to Use

The set of CP commands that you can use depends on the privilege class or classes assigned to you.

z/VM CP commands are divided into eight groups, each represented by a privilege class. The privilege class indicates the type of user from whom the commands in that class are accepted. The privilege class structure of CP commands, DIAGNOSE codes, and certain CP system functions can be expanded by using user class modification.

User class modification allows an installation to expand the privilege class structure of CP commands, DIAGNOSE codes, and certain CP system functions from eight classes to as many as 32 classes. By creating a more elaborate class structure, an installation gains greater control over the functions that each user can perform. The user classes can thus become more focused and specialized, increasing system integrity and security. For more information on user class modification, see *z/VM: CP Planning and Administration*.

In general, the system programmer who creates your system directory assigns you one or more privilege classes as part of your entry in the directory.

Privilege classes are denoted as:

- Class A through Z
- Class 1 through 6
- Class Any.

For more information about these classes, and the type of user who can use the commands belonging to each privilege class set, see *z/VM: CP Commands and Utilities Reference*.

To find out what CP commands you can use, enter:

```
query commands
```

To find out what operands are available to you on the QUERY and SET commands, enter:

```
query commands query
```

or

```
query commands set
```

Entering CP Commands before You IPL an Operating System

Before you IPL an operating system in your virtual machine, you are *always* in the CP command environment. This means that CP handles all commands you enter. To enter a CP command while you are in the CP command environment, type in the command and press ENTER. For example, to enter the QUERY TIME command, type:

```
query time
```

Then press ENTER.

Entering Virtual Machine Commands after You IPL an Operating System

After you IPL an operating system in your virtual machine, you are *usually* in the virtual machine command environment. The only time you may need to leave the virtual machine command environment is when you enter CP commands.

When you are in the virtual machine command environment, the operating system you are running handles all commands you enter. For example, after you IPL MVS/ESA in a virtual machine, you are usually in the MVS/ESA command environment. MVS/ESA handles all commands you enter.

Entering CP Commands after You IPL an Operating System

After you IPL an operating system in your virtual machine, you may need to return to the CP command environment to enter CP commands. How you get to the CP command environment depends on whether you are running in full-screen mode or line mode.

When the Production System Runs in Full-Screen Mode

When a production operating system runs in a virtual machine in full-screen mode, the production system formats the entire screen of your display. The screen looks as if you were running the production system directly on the hardware.

The default CP screen status indicators are: `CP READ`, `VM READ`, `RUNNING`, `MORE...`, `HOLDING`, or `NOT ACCEPTED`.

You can change the CP screen status indicators by changing the STATUS statement in the logo configuration file. To find out how to do this, see *z/VM: CP Planning and Administration*.

To enter a CP command, you must get to the CP environment by pressing either PA1 or a user-defined terminal break key on your display (see Note "2" on page 8). If your break key is defined as NONE, you cannot enter the CP environment while you are in full-screen mode.

Pressing PA1 (or the terminal break key you defined) causes the `CP READ` status screen indicator to appear in the screen status area. If a QUERY SET command indicates that RUN is OFF for your virtual machine, CP also temporarily halts your virtual machine.

If you did not enter the SET RUN ON command, you can enter as many CP commands as necessary from the CP environment. When you are ready to return to your virtual machine environment, enter the BEGIN command. Once back in your virtual machine environment, you may again enter the commands of your production system.

If you entered the SET RUN ON command, CP automatically returns you to your virtual machine environment after you enter a CP command.

Note:

1. Because you have more than one CP environment, remembering how to enter each CP environment may also be difficult. To prevent confusion, you can redefine the terminal break key you use to get to z/VM's CP environment. To find out how to do this, see "Step 9. Define Your Break Key" on page 23.

2. The terminal break key is the key you use to get to the CP environment from your production system's command environment. CP usually assumes that PA1 is the terminal break key. However, your production system may use PA1 for another function. In this case, you must redefine your terminal break key.

   MVS™, for example, uses PA1 to retrieve the last input line. Running MVS in a virtual machine, therefore, requires a special adjustment. See "Step 9. Define Your Break Key" on page 23 for information on how to define a break key other than PA1.

   If you do not redefine your terminal break key to some other key, and you press PA1 while running MVS in full-screen mode, you enter the CP command environment instead of retrieving the last MVS input line.

Example: Entering CP Commands While in Full-Screen Mode

Assume that you are running a VSE production system in a virtual machine, and you have not entered the SET RUN ON command. You have entered the QUERY TERMINAL command and found that PA1 is your terminal break key. To enter the QUERY CPLEVEL command and the INDICATE LOAD command, do the following:

1. Press PA1 to enter the CP environment.
2. Enter:

   ```
   query cplevel
   ```

   CP displays the response. Because you have not entered SET RUN ON, you are still in the CP environment.
3. Enter:

   ```
   indicate load
   ```

   CP displays the response. Because you have not entered SET RUN ON, you are still in the CP environment.
4. Enter:

   ```
   begin
   ```

   CP returns you to your VSE command environment.

When the Production System Runs in Line Mode

When the production operating system in a virtual machine does not run in full-screen mode, it is running in line mode, and the production system formats only the output display area of your screen. The screen status area at the bottom right of your display contains one of the CP screen status indicators. The default indicators are: `CP READ`, `VM READ`, `RUNNING`, `MORE...`, `HOLDING`, or `NOT ACCEPTED`.

You can change the CP screen status indicators by changing the STATUS statement in the logo configuration file. To find out how to do this, see *z/VM: CP Planning and Administration*.

To enter a CP command, you must get to the CP environment by doing one of the following:

- Press PA1
- Press a user defined Break Key (see Note "2" on page 9).
- Enter the #CP command without any other data.

Either action causes the `CP READ` status screen indicator to appear in the screen status area. If a QUERY SET command indicates that RUN is OFF for your virtual machine, CP also temporarily halts your virtual machine.

If you did not enter the SET RUN ON command, you can enter as many CP commands as necessary from the CP environment. When you are ready to return to your virtual machine environment, enter the BEGIN command. Once back in your virtual machine environment, you may again enter the commands of your production system.

If you entered the SET RUN ON command, CP automatically returns you to your virtual machine environment after you enter a CP command.

Note:

1. In order to use the #CP command, the line edit (LINEDIT) function must be on, and your line end character (LINEND) must be set to #. To find out if these functions are in effect, enter the QUERY SET and QUERY TERMINAL commands.
2. The terminal break key is the key you use to get to the CP environment from your production system's command environment. The default terminal break key is PA1. However, your production system may use PA1 for another function. In this case, you must redefine your terminal break key. See "Step 9. Define Your Break Key" on page 23 for information on how to define a break key other than PA1.

If the functions are not in effect, enter the following commands to activate them:

```
set linedit on
terminal linend #
```

Example: Entering CP Commands While in Line Mode

Assume that you are running a VSE production system in a virtual machine, and you did not enter the SET RUN ON command before IPLing the production system. When you enter the QUERY SET and QUERY TERMINAL commands, you find that the line edit function is on and the line end character is #. To enter the QUERY CPLEVEL and the INDICATE LOAD commands, do the following:

1. Enter:

   ```
   #cp
   ```

   CP puts you into the CP environment.

   ```
   CP READ
   ```

   appears in the screen status area.
2. Enter:

   ```
   query cplevel
   ```

   CP displays the response. Because you have not entered SET RUN ON, you are still in the CP environment.
3. Enter:

   ```
   indicate load
   ```

   CP displays the response. Because you have not entered SET RUN ON, you are still in the CP environment.
4. Enter:

   ```
   begin
   ```

   CP returns you to your VSE command environment.

Entering CP Commands for National Language Support (NLS)

NLS makes multiple languages available for some CP output (selected messages and responses). Facilities are provided that allow the user to specify and change the CP language for the virtual machine session.

Note: When you enter CP commands, they *are not* in the selected language, unless the selected language is mixed-case American English (specified as AMENG). To find out what CP languages are available to you, enter:

```
query cplanglist
```

When more than one language is available, CP responds as follows:

```
System default language:       AMENG
User's current language:       AMENG
Other available languages:     UCENG
```

Note: For a full list of available languages, see *z/VM: CP Planning and Administration*. When only one language is available, CP responds as follows:

```
System default language:       AMENG
User's current language:       AMENG
Other available languages:     NONE
```

To find out the current CP language, enter:

```
query cplang
```

To change the current CP language, see the sample format that follows:

```
set cplanguage langid
```

Example: Changing Your Language

Assume that your virtual machine is set to mixed-case American English and you want to change your language to uppercase English. Enter:

```
set cplanguage uceng
```

To change it back to mixed-case American English, enter:

```
set cplanguage ameng
```

Entering CP Commands to Control a TTY Virtual Console

You can use the TERMINAL command with its operands and options to control your TTY virtual console.

To find out what terminal options are in effect for your virtual console environment, enter:

```
query terminal
```

The TYPE, PROMPT, SCROLL, CNTL, and ASCIITBL options are displayed only when the QUERY TERMINAL command is entered from a TTY device.

To control the signalling of an attention interruption on keyboards and printers of line-mode ASCII terminals, enter:

```
terminal attn on
```

or

```
terminal attn off
```

Where:

ON
: is the default and specifies that an exclamation point (!) is displayed when an attention interruption occurs.

OFF
specifies that the exclamation point is not displayed and the carriage is not returned.

To set the prompting sequence on your TTY device, enter:

```
terminal prompt vm
```

or

```
terminal prompt tty
```

Where:

TTY
is the default and specifies the normal TTY prompt sequence of a period in column 1 and the cursor in column 2.

VM
specifies that the cursor will be positioned in column 1 of the input line for a read operation.

To indicate the number of lines that you want to be scrolled on your screen, enter:

```
terminal scroll cont
```

or

```
terminal scroll nnn
```

Where:

CONT
is the default and specifies continuous scrolling to the end of the output.

nnn
allows you to specify the number of lines that will be scrolled up before scrolling stops.

To define a device as a 3101 or as a TTY, enter:

```
terminal type tty
```

or

```
terminal type 3101
```

Where:

TTY
is the default and specifies that the terminal is regarded as a typewriter terminal.

3101
specifies that the terminal is regarded as a display screen and line-mode ASCII device providing the following functions:

- Allows you to use the program function (PF) keys PF1–PF24
- Causes your output to begin at column 1 of line 24 on the display
- Allows you to edit previously entered information without having to rekey the entire text.

Note: The PF COPY and PF TAB functions are not supported on TTY terminals.

Simulation Wait and Exigent CP Commands

Some instruction simulation functions may cause the virtual machine to enter a simulation wait state. While the virtual machine is in a simulation wait, CP commands cannot be executed.

However, certain simulation wait states can be aborted by the exigent effect of certain CP commands. The simulation wait is aborted when one of these CP commands is entered, not when it is executed. The exigent effect clears the way for CP command execution.

The exigent CP commands are IPL, LOGOFF, SYSTEM CLEAR, and SYSTEM RESET.

Using Program Function (PF) Keys

Another way to enter CP commands is to use program function (PF) keys. If you find that you frequently enter certain CP commands, you can set up PF keys to:

- Immediately enter a command for you
- Place a command line in the input area so you can change part of a command before you enter it.

If you plan to use a PF key to enter CP commands while you are in the virtual machine command environment, remember to prefix the command with #CP when you are in line-mode. (# is your logical line-end character. If you redefine your logical line-end character, use your new logical line-end character.)

Using PF Keys to Enter Commands Immediately

If you frequently use a command (such as the BEGIN command) where you do not have to change any operands, you can set a PF key to enter the command for you immediately. For example, to set PF4 to enter the BEGIN command immediately, enter:

```
set pf4 immed begin
```

Now whenever you are in the CP command environment and you press PF4, CP displays and runs the BEGIN command.

If you want the BEGIN command to be run without being displayed, you can specify the NODISP option. To do this, enter:

```
set pf4 nodisp begin
```

Now when you press PF4, the BEGIN command is run but is not displayed on your terminal.

Using PF Keys to Enter Commands with Variable Input

If you frequently use a command that includes a variable (such as a virtual device number), you can set a PF key that allows you to change the variable and run the command.

For example, to set PF6 to enter the QUERY VIRTUAL *vdev* command, where *vdev* is a virtual device number that changes, enter:

```
set pf6 substitu immed query virtual &1
```

You can now obtain the status of a virtual device by typing in the device number on the command line and then pressing PF6. For example, type the following on the command line and press PF6:

```
191
```

CP displays the status of device 191.

Using a PF Key to Retrieve Previous Lines of Input

Another way you can use a PF key to save time is to set a PF key as a Retrieve key. A Retrieve key lets you redisplay input lines you entered. For example, if you make a typing mistake when you enter a command, you can press the Retrieve key to redisplay the command, make the change, and then press ENTER to reenter the command.

You can set any PF key as a Retrieve key. For example, to set PF12 as a Retrieve key, enter:

```
set pf12 retrieve
```

After you enter this command, CP begins saving your input lines. CP does not save duplicate input lines or input lines you cannot see (for example, passwords).

If you want to clear the Retrieve buffer for security or other reasons, enter:

```
set pf12 retrieve clear
```

The system default setting lets you redisplay the last seven input lines you entered, most recent to oldest. To change the number of retrieve buffers available to your virtual machine, use the class G SET RETRIEVE command. To change the number of retrieve buffers available to all virtual machines, use the class C or E SET RETRIEVE command. For more information about changing the number of retrieve buffers, see *z/VM: CP Commands and Utilities Reference* or the *z/VM: CP Planning and Administration*.

To find out how many PF key retrieve buffers you have available to your virtual machine, enter:

```
query retrieve counts
```

To find out the contents of your PF key retrieve buffers, enter:

```
query retrieve buffers
```

Finding Out What Your PF Key Settings Are

To find out what all your PF keys are set to, enter:

```
query pf
```

To find out what a particular PF key is set to, see the following sample format:

```
query pfnn
```

The *nn* is the number of the PF key you want to query.

Protected Application Facility

The protected application facility protects you from an unexpected entry to a CP READ condition at the terminal. If a virtual machine action or CP error causes virtual machine execution to stop and return control to CP, the protected application facility automatically re-IPLs the guest operating system, leading to reinstatement of the application environment for you. Your terminal break key is also disabled to prevent unintentional entry into CP READ.

z/VM provides explicit control over the protected application environment through the CP commands SET CONCEAL, TERMINAL BRKKEY, and QUERY SET.

To activate the protected application environment for your virtual machine, enter:

```
set conceal on
```

To display the status of the protected application environment for your virtual machine, enter:

```
query set
```

To deactivate the protected application environment for your virtual machine, enter:

```
set conceal off
```

To reenable the break key after entering the protected application environment, enter:

```
terminal brkkey
```

Note: When you enter the SET CONCEAL ON command, the terminal break key is disabled. When you enter the SET CONCEAL OFF command, the terminal break key is reenabled. For more information on the CP commands SET CONCEAL, TERMINAL BRKKEY, and QUERY SET, see *z/VM: CP Commands and Utilities Reference*.

Sending Messages

To send a message to the system operator, enter:

```
message operator text
```

Where *text* is the message you want to send.

To send a message to any other virtual machine user, enter:

```
message userid text
```

Where:

userid
is the user ID of the person you are sending the message to.

text
is the message you are sending.

To send a special message to a virtual machine that is programmed to accept and process the message, enter:

```
smsg userid|* at sysname|* msgtext
```

Where:

userid
identifies the virtual machine to whom the special message is sent.

*****
indicates the message is sent to you.

at *sysname*
sends the special message to a user ID on a specific system, *sysname*, in the SSI cluster.

at *
sends the special message to the system you are logged onto.

msgtext
is the text of the message to be sent.

Example: Sending a Message to the Operator

To ask the operator to mount a tape, enter:

```
message operator please mount a tape
```

Example: Sending a Message to Another User

To tell the person whose user ID is SANTO that you are leaving for the day, enter:

```
message santo I'm leaving for the day
```

Example: Sending a Message to a Virtual Machine

To send a special message to virtual machine VMACH1, enter:

```
smsg vmach1 please stop sending me messages
```

For the class B user, to allow a service virtual machine to send messages without the standard MESSAGE command header, enter:

```
msgnoh userid|*|operator|all at all|sysname|* msgtext
```

Where:

***userid*|*|*operator*|all**
: identifies the user to whom the message is sent. To send a message to yourself, specify *; to send a message to the primary system operator, specify operator; to send the message to all users receiving messages, specify all.

at all
: sends the message to every system in the SSI cluster.

at *sysname*
: sends the message to a specific system, *sysname*, in the SSI cluster.

at *
: sends the message to the system you are logged onto.

msgtext
: is the text of the message to be sent.

Example: Sending a Message to All Users

To send a message to all the users logged onto the system you are using, enter:

```
msgnoh all at * there will be a staff meeting at 3:30 p.m. today
```

Note: If an external security manager is installed, you may not be authorized to enter the MESSAGE, MSGNOH, or SMSG commands. However, messages sent to or from the system operator and messages sent with the ALL option are not subject to authorization checking by the ESM. For additional information, contact your security administrator.

Chapter 3. Starting Your Virtual Machine

This topic contains descriptions about starting a virtual machine for any one of the following operating systems:

- z/OS
- z/VM
- VSE/ESA.

Before You Begin to Start Your Virtual Machine

Before you can log on, your virtual machine must have an entry in the z/VM user directory. This entry describes to CP such things as your logon identification, your password, your virtual machine's storage size, your virtual machine's I/O configuration, and the architecture (ESA/390, XC, or z/Architecture) that your virtual machine simulates. To change your virtual machine permanently, you must contact the person who updates the z/VM user directory.

Note:

1. With a few exceptions, the following virtual machine initialization procedure is the same for all the operating systems listed in the objectives. Where differences occur, they are noted.
2. If you run z/OS in your virtual machine with the resource measurement facility (RMF), the I/O Queuing Activity Report shows only the static configuration data.
3. z/VM supports VM/VS handshaking. This capability allows:
   - VSE to determine that it is running in a virtual machine and that it can use the features of that facility
   - VSE to close spool files automatically, so that you do not need to close them
   - VSE and CP to process pseudo page faults jointly, so that VSE can perform other tasks while waiting for a page
   - A nonpaging mode for VSE
   - VSE to notify CP that it has dynamically changed a channel program (for example, BTAM auto-poll).

   When you IPL VSE, it automatically enters the commands necessary to receive this support. For more information on processing pseudo page faults, see the SET PAGEX command in the *z/VM: CP Commands and Utilities Reference*.
4. If your virtual machine's directory entry is set up correctly, you may not need to perform some of the steps described in the following procedure, such as changing storage, changing the architecture mode your virtual machine simulates, and redefining your virtual machine operator's console.
5. The space CP needs to write a dump for a virtual machine is not automatically allocated during IPL processing. For information on allocating space for a dump, see *z/VM: CP Planning and Administration*.

Step 1. Log On to z/VM

When you turn on a 3270 display device, a z/VM logo and a logon prompt should appear. (If the z/VM logo does not appear, you cannot log on to z/VM at that device.)

You can log on from the logo, or clear the screen before entering the LOGON command. To log on from the logo, enter your user ID and password in the appropriate fields and press ENTER. If you clear the logo, you receive a logon prompting message that allows you to enter:

```
logon userid
```

If your user ID is accepted, you receive a password prompt message similar to the following:

```
ENTER PASSWORD (It will not appear when typed):
```

After you enter your password, CP displays the current system log message and you are ready to begin your session.

If an external security manager (ESM) is installed and security label checking is enabled, you can specify a security label on the LOGON command. For additional information, see *z/VM: CP Commands and Utilities Reference*.

Accessing z/VM from an SNA Terminal

If you are using an SNA terminal, you must first access VSCS (VM SNA Console Support), the VTAM application that acts as an intermediary between the SNA network and z/VM.

To log on to VSCS, enter:

```
logon applid(xxxxxxxx)
```

Where *xxxxxxxx* is the logical unit (LU) name of VSCS on the target z/VM system. If you do not know the LU name, contact your system administrator.

After logging on, your screen should display the z/VM logo. When using an SNA terminal, you can log on directly from the z/VM logo screen by entering your user ID and password in the appropriate fields and then pressing ENTER, or you can clear the logo screen by pressing ENTER and then enter the following:

```
logon userid password
```

Now you may begin your session.

Notes:

1. Some installations automatically log on SNA terminals to VSCS. If your terminal screen indicates that you are connected to z/VM, then enter the LOGON command described above.
2. If you have a password phrase or a mixed-case password, you cannot enter it on the logo. You must first clear the logo by pressing ENTER, then you can issue the LOGON command shown above.
3. This command may fail if the installation has issued the SET PASSWORD LOGON SEPARATE command (or includes the comparable statement in the system configuration file).
4. You can also use the CP DIAL command from some SNA terminals. The DIAL command is described in Chapter 6, "Managing Virtual Machine I/O and Storage Devices," on page 49.
5. An SNA terminal user cannot dial into or log on the VTAM service machine controlling the terminal.

Accessing z/VM from a Line Mode ASCII Terminal

If you are using a line-mode ASCII terminal, you must first enter the CP commands VARY ON and ENABLE to make your terminal ready to access the system. For more information about the VARY ON and ENABLE commands see *z/VM: CP Commands and Utilities Reference*.

When the communication line is enabled, a message is displayed, indicating that the terminal is connected to z/VM and informing you how to proceed. The message displayed depends on the type of terminal connected to the communication line.

TTY terminals display the following message:

```
z/VM ONLINE--systemid--PRESS BREAK KEY TO BEGIN SESSION
```

2741 terminals display the following message:

```
z/VM ONLINE--systemid--PRESS ATTN KEY TO BEGIN SESSION
```

The message that is displayed on 2741 terminals appears twice; once with *xxxxx xx xxxxxx* following the message, and once with *xxxxx xx xxxxxx* preceding the message.

Once the online message has been displayed, you can press the appropriate key and then enter:

```
logon userid
```

If the user ID is recognized, z/VM responds with the following prompt:

```
ENTER PASSWORD:
```

Enter your password to complete the logon procedure successfully. You are now ready to begin your session.

Step 2. Check Your Virtual Machine's Storage Size

When you log on, CP assigns your virtual machine the amount of storage specified in the z/VM user directory entry for your virtual machine. If you need to know how much storage your virtual machine has, enter:

```
query virtual storage
```

If you want to change this amount, do so *before* you IPL an operating system. (Redefining your storage causes CP to reset your virtual machine. Therefore, if you need to redefine storage after you IPL an operating system, you will need to reIPL that operating system.) The amount of storage you can define is limited to the amount specified in the z/VM user directory entry for you virtual machine. To change your current storage, enter the CP DEFINE STORAGE command.

You can use DEFINE STORAGE to configure reserved and standby storage before IPLing an operating system in the virtual machine. Note that using the INCREMENT operand of this command can impact a second-level instance of z/VM running in a virtual machine. See the Usage Notes section of the DEFINE STORAGE command in *z/VM: CP Commands and Utilities Reference* for information about the dynamic storage reconfiguration (DSR) unit size increment that might be applied as a result of using the INCREMENT operand.

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.

For information about the DEFINE STORAGE command, see *z/VM: CP Commands and Utilities Reference*.

Step 3. Check Your Virtual Machine's Architecture

Your virtual machine's user directory entry specifies the architecture of your virtual machine after a virtual system reset occurs, such as after LOGON or after an IPL command: XA, ESA, XC, or Z.

To IPL any operating system in your virtual machine, your virtual machine must simulate the appropriate architecture.

- ESA virtual machines process according to ESA/390 (31-bit) architecture. ESA virtual machines are also capable of processing according to z/Architecture (64-bit) if switched into that mode by a guest operating system.
- XA virtual machines are supported for compatibility and are functionally equivalent to ESA virtual machines. Some CMS applications may require CMS to be running in an XA virtual machine.
- XC virtual machines process according to ESA/XC or z/XC architecture. Guest operating systems can switch an ESA/XC virtual machine to a z/XC virtual machine.
- Z virtual machines start in and process entirely in z/Architecture.

So, for example, VSE/ESA can run in an ESA or XA virtual machine. z/VM can run in an XA, ESA, or Z virtual machine. If z/VM is run in an XA or ESA virtual machine, it switches into z/Architecture during IPL.

Note: 370 virtual machines are not supported. For information on executing System/370 applications, see the SET 370ACCOM command in the *z/VM: CP Commands and Utilities Reference*.

To determine the architecture of your virtual machine, enter:

```
query set
```

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

Step 4. Set the EREP Mode

You can have CP or the operating system you run in your virtual machine record EREP records. When you log on, CP automatically records EREP records. If, however, you want the operating system you run in your virtual machine to record EREP records, enter:

```
set svc76 vm
```

If, at any time, you want CP to resume error recording for your virtual machine, enter:

```
set svc76 cp
```

If, at any time, you need to know which operating system is recording EREP records, enter:

```
query set
```

Part of CP's response is `SVC76 CP` or `SVC76 VM`.

Note:

1. How you interpret EREP records depends on whether you enter SET SVC76 CP or SET SVC76 VM. When SVC76 CP is in effect, CP translates your virtual machine's storage addresses into host addresses and your virtual machine's I/O device numbers into real device numbers. When SVC76 VM is in effect, no such translation takes place.
2. Unless you are intentionally simulating hardware errors in your virtual machine (for example, in order to test error recovery procedures), you should let CP record all error records. When CP maintains the hardware error log, your installation has the most accurate information about equipment errors.

Step 5. Check the Device Type of Your Virtual Console

When you log on, CP defines your virtual machine operator's console as a 3270 console or a 3215 console, depending on what is specified in your user directory entry. To determine whether your virtual machine operator's console is currently defined as a 3215 console or a 3270 console, enter:

```
query virtual console
```

If you need to redefine the device type of your virtual machine operator's console before you IPL an operating system, you can:

1. Determine the device number of your existing console. To do this, enter:

   ```
   query virtual console
   ```

2. Detach the current virtual console. To do this, enter:

   ```
   detach vdev
   ```

   Where *vdev* is the virtual device number of the console.
3. Define a new console. To do this, enter:

   ```
   define console as vdev devtype
   ```

   Where:

vdev
: is the virtual device number of the new virtual machine operator's console.

devtype
: is the device type of the new virtual machine operator's console (3215 and 3270 are valid device types).

You can also use the TERMINAL CONMODE command to redefine your console's device type. If you enter the TERMINAL CONMODE 3270 command from a SNA/CCS terminal controlled by a VTAM service machine that does not support CONMODE 3270, the command is not executed. You also receive an error message indicating that there is an invalid device type. For more information on the TERMINAL CONMODE command, see *z/VM: CP Commands and Utilities Reference*.

Step 6. Set the RUN Mode

The SET RUN command determines whether your virtual machine remains stopped after you return to the CP command environment. When you log on, the RUN mode is automatically set to OFF. This means that if you return to the CP command environment, your virtual machine remains in `CP READ` status. To restart your virtual machine, enter the BEGIN command.

If the RUN mode is set ON and you enter a CP command, your virtual machine continues to run (unless you enter the STOP command). Some CP commands prevent the virtual machine from running until all terminal output associated with the command has been displayed. If the output is delayed, the virtual machine will not run. For example, if the screen status indicator is `MORE...` or `HOLDING`, the virtual machine will not run. The SET RUN ON command does not affect this behavior.

To set the RUN mode ON, enter:

```
set run on
```

If after you enter SET RUN ON, you want your virtual machine to remain in

```
CP READ
```

status after you enter a CP command, enter:

```
set run off
```

Note:

1. If, at any time, you need to know whether the RUN mode is ON or OFF, enter:

   ```
   query set
   ```

   Part of CP's response is `RUN OFF` or `RUN ON`
2. You may want to set the RUN mode ON if you are using your virtual machine to run a production system or if other users access your operating system. Setting RUN ON allows the users who access your operating system to continue to run while you enter CP commands.
3. If, instead of first returning to the CP command environment, you prefix a CP command with #CP, your virtual machine continues to run whether the RUN mode is ON or OFF.

   For more information about the SET RUN command, see *z/VM: CP Commands and Utilities Reference*.

Step 7. Set the Processor Identification Number

For your base processor

If you plan to run JES3 under MVS, or RSCS under VM, you may need to change your virtual machine's base processor identification number.

To find out the current identification number of your virtual machine's base processor, enter:

```
query cpuid
```

CP displays a 16-character hexadecimal number; the third through eighth characters are your virtual machine's base processor identification number.

To change your virtual machine's base processor identification number, enter:

```
set cpuid nnnnnn
```

Where *nnnnnn* is the new processor identification number in hexadecimal (you can omit leading zeros).

For an additional processor

If you defined more than one processor for your virtual machine (see Chapter 5, "Managing Virtual Processors," on page 43) and you want to change the identification number of any of those additional processors, enter:

```
cpu xx set cpuid nnnnnn
```

Where:

xx
: is the number you assigned as the address when you defined the additional processor.

nnnnnn
: is the new processor identification number in hexadecimal for the additional processor (you can omit leading zeros).

The CPU command causes the additional processor, rather than your base processor, to process CP commands.

Step 8. Check Missing Interrupt Control

CP automatically checks I/O devices for missing interrupt conditions. The default is for CP *not* to handle missing interrupts for your virtual machine. If your virtual machine is at this default setting, CP sends a message to you (the virtual machine operator) when it detects a missing interrupt. If CP *is* handling missing interrupts for your virtual machine, it does the following:

1. Cancels the active I/O on the real device on which the missing interrupt was detected.
2. If the I/O operation your virtual machine requested was active on the real device, CP reflects an interface control check to your virtual machine.
3. If the I/O operation your virtual machine requested was *not* yet active on the real device, then CP retries the I/O operation up to five times. If the I/O operation is unsuccessful after five attempts, CP reflects an interface control check to that virtual machine.

Normally, your user directory entry specifies whether CP handles missing interrupts for your virtual machine. To determine whether CP is handling missing interrupts for your virtual machine, enter:

```
query set
```

CP's response includes either `MIH ON` or `MIH OFF` commands. MIH ON means that CP *is* handling missing interrupts. If you do *not* want CP to handle missing interrupts, enter:

```
set mih off
```

CP still sends you a message when it detects a missing interrupt.

If CP's response to the QUERY SET command includes `MIH OFF`, CP is *not* handling missing interrupts. If you *do* want CP to handle missing interrupts, enter:

```
set mih on
```

Note:

1. You can use the SET MITIME command (class A, B) to control the interval at which CP checks a real I/O device for missing interrupts. For more information, see *z/VM: CP Commands and Utilities Reference*.
2. If you are running z/VM in a virtual machine, the missing interruption detector defaults to OFF at IPL, and the system running in the virtual machine does not detect missing interrupts. You can use the SET MITIME command to turn on the missing interrupt detector. (For more information about this command see *z/VM: CP Commands and Utilities Reference*.) If you are using the same terminal to communicate with CP and the operating system running in your virtual machine, MITIME should remain OFF for that terminal. Otherwise, the system running in the virtual machine can be logged off.
3. Missing interrupt detection is not provided for the Asynchronous Data Mover Facility.

Step 9. Define Your Break Key

Your virtual machine's break key allows you to return to the z/VM CP command environment after you have IPLed an operating system in your virtual machine. Unless you specify otherwise, PA1 is your break key.

Different operating systems may have different functions for PA1. For example, MVS uses PA1 to retrieve the last input line. z/VM uses PA1 to let their users return to the z/VM command environment after the user IPLs another operating system in a z/VM machine. (Remember, if you run a VM operating system in a virtual machine, you can use your virtual machine to create more virtual machines.)

If, when you run an operating system in a virtual machine, you use PA1 to perform some function, you must define a key other than PA1 as your break key. Defining a new break key lets you continue to use PA1 to perform other functions.

To define a break key other than PA1, enter:

```
terminal brkkey key
```

Where *key* is PF1, PF2...PF24, CLEAR, PA2, or NONE.

NONE specifies that no break key be defined.

If you are using a SNA/CCS terminal and specify the BRKKEY option, only PA1 and NONE are supported. If you enter the TERMINAL BRKKEY command with any other options (from a SNA/CCS terminal), you receive a message that indicates an invalid device-type error.

Note: If you change the TERMINAL BRKKEY setting while logged on to a local display, then disconnect and reconnect to a SNA/CCS terminal, your TERMINAL BRKKEY setting is affected in the following way during the reconnection processing:

- If BRKKEY was previously set to NONE or PA1, it is not changed.
- Otherwise, BRKKEY is set to PA1, and a message is issued to inform you of the change.

If, later, you want to redefine your break key as PA1, enter:

```
terminal brkkey pa1
```

If, at any time, you need to know which key is your break key, enter:

```
query terminal
```

Part of CP's response includes the BRKKEY *nnn* command, where *nnn* is your break key (for example, BRKKEY PA1 or BRKKEY PF12).

Step 10. Load (IPL) Your Operating System

How you IPL an operating system in your virtual machine depends on whether you are IPLing a named saved system (NSS). (For information on how to save your operating system as a named system, see "Managing Your Operating System as a Named Saved System" on page 30.)

To IPL a named saved system, enter:

```
ipl sysname
```

Where *sysname* is the 1- to 8-alphanumeric-character name of the named saved system you want to IPL.

Note:

1. If the NSS was created using the VMGROUP option of the DEFSYS command, your virtual machine is a member of a virtual machine group identified by *sysname*. You receive messages about members leaving or joining the group.
2. When you use the IPL command, it cancels all previous virtual machine group memberships.
3. If an ESM is installed, you may not be authorized to IPL all NSSs. If the NSS is class R, you must have a NAMESAVE statement in your user directory to IPL it. Therefore, you my not be able to IPL an NSS even if an ESM is not installed. For additional information, contact your security administrator.

To IPL an operating system that is not a named saved system, enter:

```
ipl vdev
```

Where *vdev* is the virtual device number of the virtual device on which the system residence volume of your operating system resides.

After you IPL an operating system, control that operating system the same way you would if it were running on a real processor. Also, remember that after you IPL an operating system, that operating system processes commands you enter. If you want to enter a CP command, do one of the following:

- Press break (unless you have specifically redefined your break key, press PA1) to return to the CP command environment. After you press Break, `CP READ` will display at the lower right of your virtual machine operator's console. Enter the CP command. Then, to return to the virtual machine command environment, enter:

  ```
  begin
  ```

  Note: If you previously entered the SET RUN ON command, you do not have to enter BEGIN to return to the virtual machine command environment. CP automatically returns you to the virtual machine command environment after you enter a CP command.

- If you are not in full-screen mode and you want to enter a specific CP command without returning to the CP command environment, use the #CP prefix. (# is your logical line-end character. If you have redefined your logical line-end character, use your new logical line-end character.)

  For example, if you want to enter the CP QUERY TIME command, enter:

  ```
  #cp query time
  ```

Using the LOADPARM Option

Use the LOADPARM option to pass a load parameter of up to 8 bytes of data to the operating system you are IPLing. You may use the load parameter to pass data for any purpose. It is frequently used to pass a console address to the Stand-alone Program Loader (SAPL) provided as part of z/VM or to specify an alternate MVS nucleus at IPL time. For more information about the Stand-alone Program Loader (SAPL), see *z/VM: System Operation*. Examples of both cases are given below:

To use the IPL LOADPARM option, enter:

```
ipl vdev loadparm data
```

or

```
ipl sysname loadparm data
```

Where:

vdev
is the virtual device number of the virtual device on which the system residence volume of your operating system resides.

sysname
is the 1- to 8-alphanumeric-character name of the named saved system you want to IPL.

data
is the 1- to 8-alphanumeric-characters you want to pass to the operating system you are IPLing.

For more information about the LOADPARM option on the IPL command, see *z/VM: CP Commands and Utilities Reference*.

Example: Specifying a Console Address for SAPL

The details of the following example are relevant only if you are IPLing a volume on which the Stand-alone Program Loader has been installed and you want to specify a console address to activate the fullscreen option of SAPL.

If you are IPLing the virtual device B00 on which SAPL is installed and you have your virtual console defined at virtual address 009, you can enter:

```
ipl B00 loadparm 9
```

SAPL will determine that a console address was specified with the LOADPARM option and will display a fullscreen menu on your console. You can choose what object you want SAPL to load and where in storage it is placed by modifying fields on this screen. For more information about the Stand-alone Program Loader, see the *z/VM: System Operation*.

Example: IPLing an Alternate MVS Nucleus

The details of the following example are relevant only if you are IPLing MVS or z/OS and wish to specify an alternate nucleus.

If you are IPLing from virtual device 292 and you want to use an alternate nucleus called IEANUC04 rather than the default nucleus IEANUC01, you enter:

```
ipl 292 loadparm 4
```

This passes 4 as data to the MVS Control Program that you are IPLing. MVS or z/OS takes this data and appends it to form the alternate nucleus name IEANUC04.

Chapter 4. Operating Virtual Machines

Once you have initialized your virtual machine and the operating system you want to run, you can control the operating system the same way you would if it were running on a real processor. In addition to running your operating system, you may need to use CP commands to perform the tasks listed in this topic.

This topic contains descriptions about:

- Stopping and restarting the execution of work within a virtual machine
- Keeping a console spool file
- Finding guidelines for saving your operating system as a named saved system
- Simulating real machine functions
- Disconnecting and reconnecting your virtual machine operator's console
- Logging off or ending a virtual machine session.

Stopping Your Virtual Machine

Stop your virtual machine in one of the following ways:

- Press your virtual machine's break key (PA1 is your break key unless you have previously specified otherwise). When

  ```
  CP READ
  ```

  is displayed at the lower right of your display screen, enter:

  ```
  stop
  ```

  Note:

  1. If you are running a virtual multiprocessor environment, use the STOP command to stop work on particular processors.
  2. If the RUN mode of your virtual machine is OFF, you do not need to enter the STOP command. Pressing break stops your virtual machine.

- If you are not in full-screen mode, enter:

  ```
  #cp stop
  ```

  Note: After you enter #CP STOP,

  ```
  CP READ
  ```

  is displayed in the screen status area on the lower right of your virtual machine operator's console.

Restarting Your Virtual Machine

To restart your virtual machine at the point where you previously stopped it, enter:

```
begin
```

Keeping a Console Spool File

A console spool file (also called a console log or console record) is a record of the following information:

- Commands you enter while you are logged on

- Responses you receive to these commands
- Messages you send or receive.

Note: While your virtual machine runs in full-screen mode, CP does not record information for your console spool file.

Starting Your Console Spool File

To start a console spool file, enter:

```
spool console start
```

Closing Your Console Spool File

To close a console spool file, enter:

```
close console
```

When you *close* a console spool file, you make it available to CP for processing. Depending on how the spooling options of the spool file are set, CP prints the spool file automatically or leaves it queued for your spooled printer. For more information about spooling options, see Chapter 7, "Using Spooled Devices to Print, Punch, and Read Information," on page 75.

Note: When you close a console spool file, CP automatically creates, or "opens," a new one for you.

Ending Console Spooling

If you want CP to stop console spooling and close the current spool file you are keeping, enter:

```
spool console stop close
```

Querying CP for Miscellaneous Information

CP keeps a variety of information about the system and your virtual machine. The following examples describe how to use the QUERY command to obtain some of this information. (Other commands you can use to query CP for information are described throughout the book.)

- To display the system log message, enter:

  ```
  query logmsg
  ```

- To display the time, date, and total CPU time your virtual machine has used, enter:

  ```
  query time
  ```

- To display the user IDs of other users logged on to z/VM, enter:

  ```
  query names
  ```

  Note: If your system has an active VTAM service virtual machine managing a SNA/CCS terminal, the response to QUERY NAMES includes the user ID of the VTAM service machine and the logical unit name (*luname*) of the SNA/CCS terminal.

- To find out if another user is logged on to z/VM, or to display the address of a particular user's console (including your own), enter:

  ```
  query users userid
  ```

  Where *userid* is the user ID of the person whose console address you are trying to display.

  For SNA/CCS terminals, *luname* is displayed instead of the console address.

To obtain information about the address spaces your non-XC virtual machine is authorized to access, use the QUERY SPACES command. For additional information, see *z/VM: CP Commands and Utilities Reference*.

Responding to Virtual Machine Messages during IPL or Error Recovery

During IPL or error recovery of some guest operating systems, the z/VM Control Program displays guest operating system messages intended for the system hardware console at the virtual machine operator's console. This prevents the loss of guest operating system messages when normal communication paths do not work. CP may add to or alter the guest operating system message text to indicate priority or hold status or to show that a message is a prompt.

Note: Not all operating systems can take advantage of this CP function.

To respond to guest operating system messages displayed by CP, enter:

```
vinput vmsg data
vinput pvmsg data
```

Where:

VMSG
: requests the last 16 nonpriority messages.

PVMSG
: requests the last 16 priority messages.

data
: is the command to be sent to the guest operating system.

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

CP retains copies of the last 16 priority and nonpriority messages it intercepts from the guest operating system. To display these messages enter:

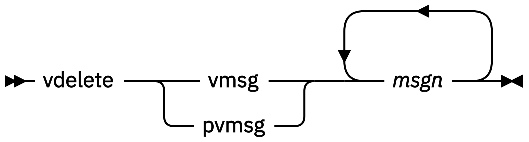
```
query vmsg
query pvmsg
```

Where:

vmsg
: requests the last 16 nonpriority messages.

pvmsg
: requests the last 16 priority messages.

CP also retains messages in the hold state until either the guest operating system or the virtual machine operator deletes them from storage. You can delete messages by entering the VDELETE command. To do this, use the following format:



Where:

vmsg
: means that a nonpriority message is to be deleted.

pvmsg
: means that a priority message is to be deleted.

msgn
is the number of the messages to be deleted. This number is displayed on the virtual machine operator's console when the QUERY VMSG or QUERY PVMSG command is entered.

Note: You should use VDELETE only for messages displayed in response to your QUERY VMSG or QUERY PVMSG command or in other, exceptional cases. Ordinarily, the guest operating system deletes messages when the situation that prompted them no longer exists.

Managing Your Operating System as a Named Saved System

A *named saved system* is an operating system that you can IPL by name rather than by device number. Saving your operating system as a named saved system (NSS) makes initializing it in your virtual machine more efficient, and can save system resources (such as real storage, paging slots, and system overhead).

Before you attempt to save your operating system as a named saved system, remember:

- To save your operating system as a named saved system, you must be able to stop I/O activity and nontimer external interrupts for your virtual machine.
- You cannot save an operating system running in a virtual machine for which you have defined multiple virtual processors.

Step 1. Defining a Named Saved System

Use the DEFSYS command to define the system you want to save. (DEFSYS is a class E CP command. You can enter the DEFSYS command before or after you load the operating system you want to save into your virtual machine.)

The format of the DEFSYS command is as follows:

```
defsys name hexpage1-hexpage2 type minsize=nnnnnnnK rstd parmregs=m vmgroup

defsys name hexpage1-hexpage2 type minsize=nnnnM rstd parmregs=none vmgroup
```

Where:

name
is a 1- to 8-alphanumeric-character name for the system you are defining. Do not assign a name that is also a valid device address (or CP will IPL the device, not the system you have named).

hexpage1-hexpage2
are the first and last hexadecimal page numbers of the pages to be saved. You can specify a single page range or multiple page ranges.

You must specify the page number as a hexadecimal value less than or equal to X'7FEFF'(2047M). The following list shows examples of how storage addresses translate into page numbers:

Hexadecimal Storage Address	Hexadecimal Page Number
00000000	0
00001000	1
00022000	22
00333000	333
04444000	4444
3E6FF000	3E6FF
7F800000	7F800

type
is the page descriptor code of a page range. It indicates the type of virtual machine access permitted to pages in the range. The *type* variable is specified as two characters:

The first character is either **E** for exclusive access, or **S** for shared access.

The second character defines the storage-protection characteristics of the page and is designated as follows:

R means that the page is read-only.

W means that users have read/write access to the page.

N stands for *no data saved,* meaning that read/write pages of zeros are loaded for these pages when a user IPLs this named saved system. The contents of *no data saved* pages are not saved by the SAVESYS command, which reduces the amount of spooling slots required to save the NSS.

C means that the NSS contains CP-writable pages that can be accessed in shared read-only access mode by the virtual machine and that no data is saved.

Valid *types* are:

Type	Explanation
EW	Exclusive read/write access
EN	Exclusive read/write access, no data saved
ER	Exclusive read-only access
SW	Shared read/write access
SN	Shared read/write access, no data saved
SR	Shared read-only access
SC	Shared read-only access by the virtual machine, no data saved, CP-writable pages.

Note: Shared read/write segments between virtual machines can cause one virtual machine to cause the other to fail. Before you specify a type, remember:

- If the segment is reentrant and unmodified, you should specify one of the shared (S) types.
- You must specify a type for each page range you specify.
- You must specify the same shared versus exclusive attribute for all page ranges within a single storage segment (megabyte).
- You must specify exclusive access (EW, ER, or EN) for any pages in segment zero. If any pages in segment zero are specified with shared access, the command will be rejected.

minsize=*nnnnnnn*K | minsize=*nnnn*M
specifies the minimum virtual storage size for a virtual machine into which a named saved system can be loaded.

rstd
is an optional parameter indicating that you want to restrict the use of the named saved system. If you do not specify RSTD, anyone can IPL the named saved system. If you do specify RSTD, then users wishing to IPL the named saved system must have the appropriate NAMESAVE statement in their user directory entries. For more information about the NAMESAVE statement and for information on user directories in general, see *z/VM: CP Planning and Administration*.

parmregs=*m* | parmregs=none
indicates that:

1. A range of general purpose registers has been specified to contain the IPL parameter string (*m*–*n*).
2. No range is specified (none) because no parameters are to be passed when the NSS receives control from CP.
3. Only one register is required, and you may specify PARMREGS=*m*.

vmgroup
is an optional parameter indicating that a user who IPLs this NSS becomes a member of a virtual machine group, identified by the name of the NSS.

Step 2. Saving a Named Saved System

To save a named saved system, do the following:

1. Initialize your operating system in your virtual machine.
2. Make sure your operating system is stopped. Here are some reminders and guidelines:
   - Make sure there are no nontimer external interruptions pending.
   - Quiesce all I/O activity.
   - If a wait PSW is loaded when you save a named saved system, a person's virtual machine is put into a wait state when that person IPLs the named saved system.
3. Enter the following command:

   ```
   savesys name
   ```

   Where *name* is the 1- to 8-alphanumeric-character name you specified when you defined the named saved system.
4. If the system you are saving contains any initialized checkpoint files or data bases, make sure you preserve them before allowing the operating system to continue operation. When a user IPLs the NSS, these areas must be available and in the same state they were in at the time you saved the system.

IPLing a Named Saved System

After you define and save a named saved system, you (or any authorized user) can IPL the system by name rather than device number. For example, assume you have defined and saved a system named NAMSAV1. To IPL this named saved system, enter:

```
ipl NAMSAV1
```

Note: If an external security manager (ESM) is installed, you may not be authorized to IPL all NSSs. If the NSS is class R, you must have a NAMESAVE statement in your user directory to IPL it. Therefore, you may not be able to IPL an NSS even if an ESM is not installed. For additional information, contact your security administrator.

Keeping Backup Copies of Named Saved Systems

Because CP uses system spooling space to store named saved systems, and you may not be able to recover spooling space after a CP abnormal termination, you should always keep backup copies of named saved systems on tape. For more information on how to use the SPXTAPE command to keep backup copies of spool files, see *z/VM: System Operation*.

Displaying Information about Named Saved Systems

Use the QUERY NSS command to display information about existing named saved systems. (QUERY NSS is a class E command.)

Example 1: Displaying General Information

To display general information about all of the named saved systems that exist, enter:

```
query nss all
```

To display general information about a particular named saved system, for example, one that you have named NAMSAV1, enter:

```
query nss name namsav1
```

In response to these commands, CP displays information in the following format:

```
OWNERID FILE TYPE CL RECS DATE    TIME     FILENAME FILETYPE ORIGINID
*NSS    spid NSS  c  nnnn mm/dd hh:mm:ss filename NSS      originid
```

Where:

OWNERID
identifies the owner of the file. The designation *NSS indicates that the file is owned by the system. *NSS is not a user ID; it is the name of the queue on which the file resides.

FILE
identifies the spool file number.

TYPE
identifies the type of system data file. All files containing named saved systems are type NSS.

Note: Do not confuse this with the file type of the system data file.

CL
identifies the file class, which describes the current state of the file:

A
indicates that the named saved system is available for use.

P
indicates that the file is currently being used but is no longer available and that no new users may access it. CP purges class P files if:

- CP is re-IPLed or restarted, or
- The last virtual machine to use the file is re-IPLed, or
- The user enters a SYSTEM CLEAR, or
- Storage is redefined.

R
indicates that the named saved system is available for use, but has restricted access. A NAMESAVE directory statement is required in a user's directory in order for the user to gain access to a restricted NSS.

S
indicates that the file is in *skeleton* format. This means it was defined but not saved.

RECS
specifies the number of logical records in the spool file. If the number is greater than 9999, the number is shown as *nnn*K, where *nnn* is the number of lines in thousands rounded to the nearest thousand. If the number is greater than 999499, the number is shown as *nnn*M, where *nnn* is the number of lines in millions rounded to the nearest million.

DATE
identifies the month and day the file was closed. If the file is not closed, this is the date the file was opened.

TIME
identifies the time the file was closed in hours, minutes, and seconds. If the file is not closed, this is the time the file was opened.

FILENAME
identifies the file name of the system data file, which is also the name of the saved system.

FILETYPE
identifies the file type of the system data file. This is NSS for all named saved systems.

ORIGINID
identifies the user ID of the user who saved the named saved system (for a class A, R, or P file). If the file is a skeleton (class S), this is the user who defined it.

Example 2: Displaying More Detailed Information

To display more detailed information about all the named saved systems that exist, enter:

```
query nss all map
```

To display detailed information about a particular named saved system, for example, one that you have named NAMSAV1, enter:

```
query nss name namsav1 map
```

In response to these commands, CP displays information in the following format:

```
FILE FILENAME FILETYPE MINSIZE BEGPAG ENDPAG TYPE CL #USERS PARMREGS VMGROUP
spid filename filetype minsize begpag endpag type c  nnnnn  parmregs vmgroup
```

Where:

FILE
identifies the spool file number.

FILENAME
identifies the file name of the system data file, which is also the name of the saved system.

FILETYPE
identifies the file type of the system data file. This is either NSS or CPNSS (for a CP system service NSS).

MINSIZE
specifies the minimum storage size of the virtual machine into which the named saved system can be IPLed:

***nnnnnnn*K**
is the storage size in decimal kilobytes.

000*nnnn*M
is the storage size in decimal megabytes.

BEGPAG
specifies the beginning page number of a page range of the named saved system.

ENDPAG
specifies the ending page number of a page range of the named saved system.

TYPE
identifies the page descriptor code of a page range of the named saved system. It indicates the type of virtual machine access permitted to pages in the range.

Note: BEGPAG, ENDPAG, and TYPE can be repeated for several sets of page ranges.

CL
identifies the file class, which describes the current state of the file:

A
indicates that the named saved system is available for use.

P
indicates that the file is currently being used but is no longer available and that no new users may access it. CP purges class P files if:

- CP is re-IPLed or restarted, or
- The last virtual machine to use the file is re-IPLed, or
- The user enters a SYSTEM CLEAR, or
- Storage is redefined.

R
indicates that the named saved system is available for use but has restricted access. A NAMESAVE directory statement is required in a user's directory for the user to gain access to a restricted NSS.

S
indicates that the file is in *skeleton* format. This means it was defined but not saved.

#USERS
specifies the number of users attached to the named saved system.

PARMREGS
identifies the registers in which parameters are passed to the virtual machine at IPL. Possible values are:

m–n
A range of registers, where *m* and *n* are decimal integers between 00 and 15 (inclusive) and *m* is less than or equal to *n*.

NONE
PARMREGS=NONE was specified on the DEFSYS command.

OMITTED
The PARMREGS option was not specified on the DEFSYS command.

VMGROUP
specifies whether the VMGROUP attribute was specified on the DEFSYS command when the named saved system was created.

If you want to display only the number of NSS files that match the specified criteria, enter the COUNT option on the QUERY NSS command.

Note: Keep in mind that COUNT does not distinguish between named saved system files and saved segment files.

To display the user IDs that are actively using a particular named saved system enter:

```
query nss users filename
```

where *filename* is the name of the named saved system.

Purging Named Saved Systems

Use the PURGE NSS command to purge unwanted files that contain named saved systems. (PURGE NSS is a class E CP command.)

Example: Purging A Named Saved System

Assume that you have created a named saved system called NAMSAV1. To purge this named saved system, enter:

```
purge nss name namsav1
```

After you enter this command, CP purges NAMSAV1 *unless* NAMSAV1 is currently in use (by you or another user). If NAMSAV1 is currently in use, CP sends you a

```
PENDING PURGE
```

message, and then purges NAMSAV1 as soon as the file is no longer being used.

If you define and save a new named saved system with the same name as the old NSS, it is not necessary to purge the old NSS. CP purges it for you.

Simulating Real Machine Functions

Use the SYSTEM command to simulate the following real machine functions:

- Displaying the CPU timer and clock comparator
- Clearing all pending interrupts
- Clearing storage
- Using the hardware system restart facility.

Note: You must first specify the processor you want to restart. For an explanation of how to do this, see "Specifying Which Virtual Processor Handles Commands" on page 46.

Displaying the CPU Timer and Clock Comparator

To display the contents of the CPU timer and the clock comparator, use two commands. First enter:

```
system store status
```

Then, for an ESA/390 (31-bit) virtual machine, enter:

```
display d8.10
```

Or, for a z/Architecture (64-bit) virtual machine, enter:

```
display 1328.10
```

Attention: Using the SYSTEM STORE STATUS command can change values contained in your virtual machine's low storage and make it impossible for your virtual machine to continue to run.

Clearing All Pending Interrupts

To simulate the system reset button (clearing pending interrupts and active I/O operations), enter:

```
system reset
```

Clearing Storage

To clear pending interrupts and active I/O operations, and to clear your virtual machine's storage, storage keys, floating point, and general registers, enter:

```
system clear
```

Note:

1. The SYSTEM CLEAR and the SYSTEM RESET commands cause the virtual machine to be reset. If the virtual machine is a member of a virtual machine group (from an earlier IPL of a VMGROUP NSS), that group membership is cancelled.
2. SYSTEM CLEAR causes all shared storage to be released from your virtual machine.

Simulating the Hardware System Restart Facility

To simulate the hardware system restart facility (causing the RESTART NEW PSW to become the CURRENT PSW), enter:

```
system restart
```

Simulating External Interrupts

If you need to simulate an external interrupt, use the EXTERNAL command.

Simulating the External Interrupt Key

To simulate the external interrupt key, enter:

```
external key
```

CP issues an external interrupt with hexadecimal code 0040 to your virtual machine.

Simulating a Malfunction Alert

To simulate a malfunction alert, enter:

```
external malfunction n
```

Where *n* is the address (0–3F) of the virtual processor from which you enter the malfunction alert.

CP issues an external interrupt with hexadecimal code 1200 to your virtual machine.

Simulating an Interval Timer Interrupt

To simulate an interval timer interrupt, enter:

```
external interval
```

CP issues an external interrupt with hexadecimal code 0080 to your virtual machine.

Simulating an Emergency Signal

To simulate an emergency signal, enter:

```
external emergency n
```

Where *n* is the address (0–3F) of the virtual processor from which you enter the emergency signal.

CP issues an external interrupt with hexadecimal code 1201 to your virtual machine.

Simulating an External Call SIGP Signal

To simulate an external call SIGP signal, enter:

```
external call n
```

Where *n* is the address (0–3F) of the virtual processor from which you enter the external call SIGP signal.

CP issues an external interrupt with hexadecimal code 1202 to your virtual machine.

Handling Abnormal Terminations

If z/VM abnormally terminates, you must:

1. Wait until the z/VM logo reappears on your display device
2. Log on z/VM
3. Reinitialize your virtual machine and the operating system you are running. (To reinitialize your virtual machine, follow the procedure described in Chapter 3, “Starting Your Virtual Machine,” on page 17.)

Disconnecting Your Virtual Machine Operator's Console

Use the DISCONNECT command if you want to temporarily disconnect your virtual machine operator's console from your virtual machine. (Remember, if you are using two consoles to operate your virtual machine, your virtual machine operator's console is the device you use to log on to z/VM and to communicate with CP.) For example, you may want to use your display device temporarily for another purpose, or you may want to move to another display device without having to reinitialize your virtual machine.

When you use the DISCONNECT command, CP releases from your virtual machine the display device you are using as your virtual machine operator's console. The rest of your virtual machine remains intact and, if you entered the SET RUN ON command, can continue to run. When you reconnect, or reaccess your operator's console, you do *not* have to reinitialize your virtual machine. Instead, you need only log on to z/VM and, if you did not previously enter the SET RUN ON command, enter the BEGIN command.

Certain conditions cause CP to log your virtual machine off 15 minutes after you enter DISCONNECT. Entering the SET RUN ON command makes it less likely that CP will log off your virtual machine. To determine whether you entered the SET RUN ON command, enter:

```
query set
```

Part of CP's response will be `RUN ON` or `RUN OFF`. If it is `RUN OFF`, enter:

```
set run on
```

To disconnect your operator's console from your virtual machine, enter:

```
disconnect
```

If you specify the HOLD option on a non-SNA TTY display terminal connected to a 7171 Device Attachment Control Unit (DACU) or a logical terminal, it is accepted and processed. To do this, enter:

```
disconnect hold
```

If you specify the HOLD option on a SNA terminal, the command is not valid and you receive an error message. The HOLD option is accepted, but ignored on all other terminals.

Single Console Image Facility (SCIF)

By means of the single console image facility (SCIF), a user logged on to a single virtual machine can control one or more disconnected virtual machines. The controlling virtual machine is called the *secondary user.* A disconnected virtual machine being controlled is called a *primary user.*

After a primary user is under the control of a secondary user, the secondary user can enter input on behalf of the primary user by way of the CP SEND command. CP redirects the output for the primary user to the console of the secondary user, prefixing it with the primary user's user ID. Note that the single console image facility can be used only by users whose consoles operate as 3215s, as established by way of the CONSOLE directory statement or the TERMINAL CONMODE command.

A virtual machine is defined as a primary user by specifying on the CONSOLE statement in its z/VM directory entry the user ID of the secondary user by which it is to be controlled. This statement authorizes the secondary user to control the primary user by way of the single console image facility.

The secondary user receives control of the primary user when the primary user disconnects. If the secondary user is logged on, it receives control immediately. Otherwise, it receives control when it is next logged on, if the primary user is still disconnected. The primary user can regain control at its own display by using the LOGON command. When logged on, the primary user can use the SET SECUSER command to change the secondary user ID associated with its virtual machine. The primary user can also use the QUERY SECUSER command to determine the identity of its authorized secondary user.

After the primary user has disconnected, all console output not in full screen form from the primary user appears on the console of the secondary user (if logged on). Output from the primary user contains the user ID of the primary user. The prefix allows the secondary user to distinguish output from each of the primary users it controls.

The secondary user uses the CP SEND command to communicate with the primary user. The SEND command can be used to enter CP commands, virtual machine commands, responses to virtual machine or CP messages, or other virtual machine data as required on behalf of the primary user. For example, when a message indicates that the primary user has entered CP READ mode, the secondary user can use the SEND command to enter a CP command on behalf of the disconnected primary user. Similarly, when a message indicates that the primary user has entered VM READ mode, the secondary user can use the SEND command to enter the required virtual machine command or response on behalf of the disconnected primary user.

The console attributes of the secondary user are used for the display of messages. For example, if the primary user's console is spooled with the TERM option and the secondary user's console is spooled with the NOTERM option, only the messages allowed by the NOTERM option are displayed on the secondary user's console.

If the secondary user has a valid path to the message system service (*MSG) or message all system service (*MSGALL), it may run disconnected. Output on behalf of the primary user normally directed to the secondary user's console is instead directed through IUCV to one of these CP system services. For more information, see *MSG System Service and *MSGALL System Service in the *z/VM: CP Programming Services*.

The single console image facility is useful for monitoring and operating an application running in a disconnected virtual machine. It allows multiple service virtual machines (such as those in which RSCS, the VM/Pass-Through facility, and the CMS batch facility run) to be controlled from a single real display. In addition, the primary system operator can use a single real display to control both the z/VM system and one or more production or test virtual machines.

Logical Line Editing

A logical line editing facility is supported for all the data entered at the virtual operator's console. Data entered when the virtual machine is in the CP command or virtual machine command environments is edited unless all logical line editing for the virtual machine is suppressed by entering the CP command SET LINEDIT OFF.

Four logical editing functions are supported. The user calls a specific logical editing function by including the logical edit character that is associated with the function in the line of data entered at the virtual operator's console. The console input line is scanned and the editing is performed as indicated. Editing causes the logical edit character to be removed from the edited line. The edited line is then passed to the appropriate routine, such as a CP or CMS command module, for processing. The console read routine of CP that sends read commands to virtual operators' console devices performs the logical editing.

z/VM supports the following logical edit functions:

- **Logical character delete**

  The logical-character-delete character causes the character it follows to be deleted from the console input line. The system default logical-character-delete character is an "at" sign (@). Multiple successive logical-character-delete characters cause multiple character deletions. For example, if the user enters:

  ```
  began@@in
  ```

  as a console input line, the resulting edited line is

  ```
  begin
  ```

  Note that on 3270 display devices, the cursor control keys can also be used to alter incorrectly entered characters.

- **Logical line end**

  The logical-line-end character causes a new-line character to be inserted in the console input line that CP recognizes as an end-of-logical-line character. The system default logical-line-end character is a number sign (#). This function can be used to enter one physical console input line that contains multiple logical lines. For example, if the user enters:

  ```
  query names# query files# query users
  ```

  as a console input line, the edited result is the same as if the following three lines were entered:

```
query names
query files
query users
```

- **Logical line delete**

  The logical-line-delete character causes the current logical line to be deleted from the console input line. The system default logical-line-delete character is a "cent sign" (¢). For example, if the user enters:

  ```
  query names#query files¢#query users
  ```

  as a console input line, the edited result is the following:

  ```
  query names
  query users
  ```

  The ERASE INPUT key can also be used for canceling an input line on a 3270 display.

- **Logical escape**

  The logical-escape character causes the character it precedes to be ignored as a console-input line editing character. The system default logical escape character is a quotation mark ("). This function enables the system default edit characters (@, #, ¢, and ") to be interpreted literally instead of as editing characters when they are preceded by the escape character. For example, if a user enters:

  ```
  msg susan can I borrow some 11"" BY 17"" paper?
  ```

  as a console input line, the resulting edited line (the message SUSAN receives) is:

  ```
  CAN I BORROW SOME 11" BY 17" PAPER?
  ```

You can override the defaults for the logical line editing characters by using the CHARACTER_DEFAULTS statement in the system configuration files. For more information, see *z/VM: CP Planning and Administration*.

Individual logical edit functions can be suppressed, or a user-supplied logical edit character can be used instead of the system default character. The logical editing options specified in the z/VM directory entry for a virtual machine can be overridden by the TERMINAL command (CHARDEL, ESCAPE, LINEDEL, and LINEND options).

The TERMINAL command can establish the following for the virtual operator's console:

- Whether CP mode or virtual machine mode is in effect (MODE option). Virtual machine mode is the default for all users except the primary system operator.
- The logical tab character (TABCHAR option).
- Whether APL translation tables are used instead of binary-coded decimal (BCD) or correspondence code translation tables (APL option). APL ON can be specified for 3270 displays that have an APL feature installed to provide APL character support. The default is to not use APL translation tables.
- Whether text translation tables are used instead of the standard tables (TEXT option). TEXT ON can be specified for 3270 displays that have a text feature installed to provide text character support. The default is to not use these tables.
- The line length of console output (LINESIZE option). The maximum length is 255 characters.
- Whether the input command line is highlighted when CP redisplays it on a 3270 screen (HILIGHT option). The default is to not highlight.
- Whether the virtual operator's console is supported as a 3215 or a 3270 device at the START I/O or START SUBCHANNEL level (CONMODE option). The default is 3215.
- Whether CP breaks in with messages immediately or only through use of the break key while the virtual machine is running in full-screen mode (BREAKIN option). The default is to break in immediately.
- The break key, if any (BRKKEY option). The default break key is PA1.

- Whether a carriage return is performed following execution of the 3215 CCW X'01' (write without carriage return) to a graphic display operating in line mode. (AUTOCR option). The default is ON, which specifies that a subsequent line mode write-to-display will start on the line following the data just written.

The QUERY TERMINAL command can be entered to determine the TERMINAL command options that are currently in effect.

Note that when a virtual operator's console is disconnected and later reconnected, the line editing characters and attention mode (CP or virtual machine) made effective by the TERMINAL command before the disconnection are still in effect when the reconnection occurs. The other settings established by the TERMINAL command are reset to their default values.

Reconnecting Your Virtual Machine Operator's Console

When you are ready to reconnect your operator's console and reaccess your virtual machine, the first thing you must do is log on. If CP is able to reconnect you, you receive a message to that effect. If you do not receive a message advising that you are reconnected, you must reinitialize your virtual machine. (To reinitialize your virtual machine, follow the steps in Chapter 3, “Starting Your Virtual Machine,” on page 17.)

If CP is able to reconnect you, enter the BEGIN command to reaccess your virtual machine.

Note:

1. If, when you disconnected, the RUN mode was ON, you automatically reaccess your virtual machine and therefore do not need to enter the BEGIN command.
2. If you are running VSE and you are using your virtual machine operator's console as your system console in full-screen mode, do the following after you enter the BEGIN command:
    a. Check the screen status area on the bottom right of your display screen.
    b. If the screen status area indicates `MORE...`, press CLEAR or PA2. When the screen clears, you can resume work.
    c. If the screen status area does not indicate `MORE...`, VSE has temporarily *lost* your console. To resolve this situation, enter:

    ```
    #cp external
    ```

    You may need to continue entering the EXTERNAL command until you receive the `MORE...` prompt. When you receive the `MORE...` prompt, press CLEAR or PA2. When the screen clears, you can resume work.

Ending a Virtual Machine Session (Logging Off)

When you use the LOGOFF command, CP makes all of your virtual machine's resources available to other users (excluding your virtual machine's permanent DASD and minidisks). To reaccess, or recreate, your virtual machine after you log off, you must log on *and* reinitialize your virtual machine.

To log off, enter:

```
logoff
```

If you specify the HOLD option on a non-SNA TTY display terminal connected to a 7171 Device Attachment Control Unit (DACU), it is accepted and processed. To do this, enter:

```
logoff hold
```

If you specify the HOLD option on a logical device or a SNA/CCS terminal, the command is not valid and you receive an error message. The HOLD option is accepted, but ignored on all other terminals.

Chapter 5. Managing Virtual Processors

CP shares real processor resources so that all virtual machines appear to have at least one of their own processors (called the base virtual processor). This processor defaults to having an initial address of 0, unless your virtual machine's user directory entry specifies another address.

Your user directory entry also specifies the number of processors that are already defined for you when you log on, and the total number of virtual processors you can define.

This topic contains descriptions about:

- Adding, detaching, and querying virtual processors
- Starting and stopping the execution of work on virtual processors
- Specifying which virtual processors are to handle specific CP commands

Adding Virtual Processors

Use the DEFINE CPU command to add virtual processors to your virtual machine. Note that CP does not let you define more processors than the maximum contained in your virtual machine's user directory entry.

Example: Defining a Two-Way Multiprocessor Configuration

To define a two-way multiprocessor configuration for your virtual machine, enter:

```
define cpu 2
```

Where *2* is the address of the additional virtual processor.

After you enter this command, CP responds with a message similar to the following:

```
00: CPU 02 DEFINED
```

Where *00* is the address of your base processor (the processor on which you entered the DEFINE CPU command).

Example: Defining a Four-Way Multiprocessor Configuration

To define a four-way multiprocessor configuration for your virtual machine, enter:

```
define cpu 1-3
```

Where *1–3* are the addresses of the additional virtual processors.

After you enter this command, CP responds with a message similar to the following:

```
00: CPU 01 DEFINED
00: CPU 02 DEFINED
00: CPU 03 DEFINED
```

Where *00* is the address of your base processor (the processor on which you entered the DEFINE CPU command).

Detaching Virtual Processors

Before you detach a virtual processor, remember:

1. Detaching a virtual processor causes CP to reset your virtual machine. To resume work after CP resets your virtual machine, you must re-IPL the operating system you are running in your virtual machine.

2. You cannot detach your base processor regardless of its address.
3. If the virtual processor being detached was dedicated to a real processor, the real processor will now be used by all other virtual machines.

To detach a virtual processor other than your base processor, enter:

```
detach cpu n
```

Where *n* is the address (0, 1, ...) of the processor you want to detach.

To detach a range of virtual processors, enter:

```
detach cpu m-n
```

Where *m*–*n* are the addresses of the first and last processors in the range of processors you want to detach.

To detach all virtual processors, except the base processor, enter:

```
detach cpu all
```

Redefining the Address of a Processor

To redefine the address of a processor, enter:

```
define cpu m as cpu n
```

Where:

m
is the old address of the virtual processor you want to change.

n
is the new address you want to assign to your processor.

Example: Redefining the Address of Your Base Processor

To redefine the address of your base processor, assuming your base processor is at the default value of 0, enter:

```
define cpu 0 as cpu n
```

Where *n* is the new address you want to assign to your base processor.

Example: Redefining the Address of Any Processor

To redefine a virtual processor at address 2 to an address of 5, enter:

```
define cpu 2 as cpu 5
```

Stopping Virtual Processors

To stop all your virtual processors, do one of the following:

1. Press your virtual machine's break key (PA1 is your break key unless you have previously specified otherwise). When

   ```
   CP READ
   ```

   is displayed at the lower right of your display screen, enter:

   ```
   stop
   ```

Note: If the RUN mode of your virtual machine is OFF, you do not need to enter the STOP command. Pressing Break stops all of your virtual processors.

2. Enter:

   ```
   #cp stop
   ```

   After you stop your virtual processors,

   ```
   CP READ
   ```

   is displayed in the screen status area on the lower right of your virtual machine operator's console.

To stop a particular virtual processor, enter:

```
#cp stop cpu n
```

Where *n* is the address of the virtual processor you want to stop.

To stop a range of virtual processors, enter:

```
#cp stop cpu m-n
```

Where *m*–*n* are the addresses of the first and last virtual processors you want to stop.

Restarting Virtual Processors

To restart all virtual processors from the point where they were stopped, enter:

```
begin
```

To restart only the virtual processor on which you enter commands, enter:

```
begin *
```

To restart a specific virtual processor, enter:

```
begin cpu n
```

Where *n* is the address of the processor you wish to restart.

Finding Out about Your Virtual Processors

To find out how many virtual processors you have, enter:

```
query virtual cpus
```

If you have only your base virtual processor defined, CP responds to the query with a message similar to the following:

```
CPU 00  ID  FF02142730810000 (BASE)
```

If you have defined additional virtual processors, CP's response also gives information about them. For example, if you defined but not started an additional virtual processor at address 2, CP responds with a message similar to the following:

```
00: CPU 00  ID  FF02142730810000 (BASE)
00: CPU 02  ID  FF02142730810000 STOPPED
```

Once you start processor 2 (using the BEGIN CPU command described above), STOPPED no longer appears in the response.

Specifying Which Virtual Processor Handles Commands

Use the CPU command to select one virtual processor to handle all CP commands, and to enter a specific CP command on a specific virtual processor. Unless you specify otherwise, your base processor (processor 0) handles all CP commands.

To determine which processor is currently handling commands, enter any CP command (for example, QUERY TIME). The output you receive looks similar to that contained in the following example. The number on the left (in this case 00) tells you which virtual processor is currently handling commands.

For SNA/CCS terminals, the command is redisplayed exactly as you typed it (without the processor number).

```
00: QUERY TIME
00: TIME IS 13:20:25 EST FRIDAY 01/19/95
00: CONNECT= 00:12:51 VIRTCPU= 000:01.20 TOTCPU= 000:02.80
```

Example: Selecting One Processor to Handle All CP Commands

If you want virtual processor 2 to handle all CP commands, enter:

```
cpu 2
```

If, after you enter the CPU 2 command, you reenter the QUERY TIME command, CP responds with a message similar to the following:

```
02: QUERY TIME
02: TIME IS 13:21:25 EST FRIDAY 01/19/95
02: CONNECT= 00:12:51 VIRTCPU= 000:01.20 TOTCPU= 000:02.80
```

Issuing Specific CP Commands on Specific Virtual Processors

For the following examples, assume that you have defined a four-way multiprocessor configuration and that processor 00 is handling all CP commands.

If you want to enter the QUERY TIME command on virtual processor 3, enter:

```
cpu 3 cmd query time
```

CP responds with a message similar to the following:

```
00: CPU 3 CMD QUERY TIME
03: TIME IS 13:22:25 EST FRIDAY 01/19/95
03: CONNECT= 00:12:51 VIRTCPU= 000:01.20 TOTCPU= 000:02.80
```

Example: Querying Time on Virtual Processors 1 and 2

If you want to enter the QUERY TIME command on virtual processors 1 and 2, enter:

```
cpu 1-2 cmd query time
```

CP responds with a message similar to the following:

```
00: CPU 1-2 CMD QUERY TIME
01: TIME IS 13:23:25 EST FRIDAY 01/19/95
01: CONNECT= 00:12:51 VIRTCPU= 000:01.20 TOTCPU= 000:02.80
02: TIME IS 13:23:26 EST FRIDAY 01/19/95
02: CONNECT= 00:12:51 VIRTCPU= 000:01.20 TOTCPU= 000:02.80
```

Example: Querying Time on All Processors

If you want to enter the QUERY TIME command on all processors, enter:

```
cpu all cmd query time
```

CP responds with a message similar to the following:

```
00: CPU ALL CMD QUERY TIME
00: TIME IS 13:24:25 EST FRIDAY 01/19/95
00: CONNECT= 00:12:51 VIRTCPU= 000:01.20 TOTCPU= 000:02.80
01: TIME IS 13:24:25 EST FRIDAY 01/19/95
01: CONNECT= 00:12:51 VIRTCPU= 000:01.20 TOTCPU= 000:02.80
02: TIME IS 13:24:25 EST FRIDAY 01/19/95
02: CONNECT= 00:12:51 VIRTCPU= 000:01.20 TOTCPU= 000:02.80
03: TIME IS 13:24:25 EST FRIDAY 01/19/95
03: CONNECT= 00:12:51 VIRTCPU= 000:01.20 TOTCPU= 000:02.80
```

Chapter 6. Managing Virtual Machine I/O and Storage Devices

Your virtual machine's user directory entry defines to CP your virtual machine's I/O configuration. This topic explains how you can use CP commands to add, detach, and change I/O devices temporarily. A *temporary* device is one that is automatically detached when you log off. To change your I/O configuration permanently, contact the person who updates the user directory.

This topic explains how to perform the tasks necessary to manage virtual machine I/O devices, excluding spooled unit record devices. Chapter 7, "Using Spooled Devices to Print, Punch, and Read Information," on page 75, tells how to manage spooled unit record devices. For information about managing real I/O devices, see *z/VM: System Operation*.

This topic contains descriptions about:

- Adding to or removing from a virtual machine configuration any of the following devices:
  - DASD
  - Minidisks
  - Temporary disks
  - Virtual disks in storage
  - 3270 displays, printers, and communication lines
  - Tapes
  - Virtual channel-to-channel adapters
  - Dedicated unit record devices.
- Sharing DASD among virtual machines (through reserve/release) and among non-SSI VM systems (through cross-system link)
- Linking to another user's minidisk or virtual disk in storage
- Redefining the virtual device number of any virtual machine I/O device.

Some Virtual Machine I/O Terminology

If you are unfamiliar with virtual machine I/O terminology, here are some terms you should know:

- **Virtual device**—This term refers to any device in a virtual machine's I/O configuration.
- **Dedicated device**—This term refers to either of the following:
  - A virtual I/O device to which CP has exclusively allocated (dedicated) a real device
  - The real I/O device that CP allocates (dedicates) to a virtual machine

  In the context of this book, dedicated device always refers to the virtual I/O device.
- **Virtual device number**—This term refers to a 3- or 4-digit hexadecimal number (depending on the operating system you are running in your virtual machine) by which CP identifies a virtual device.

Displaying the Status of Virtual Machine I/O Devices

To display the status of all of your virtual machine's I/O devices, enter:

```
query virtual all
```

To determine the status of a particular virtual machine I/O device, enter:

```
query virtual vdev
```

or

```
query virtual vdev details
```

Where:

vdev
is the number of the device whose status you want to know.

details
is an optional parameter that will display additional information about any of your DASDs that are attached to cached control units. (See "Cache Storage" on page 52 for an explanation of cache storage.)

For more information on the response to this command, see the QUERY (Virtual Device) command in *z/VM: CP Commands and Utilities Reference*.

Redefining a Virtual Device Number for a Virtual Device

When you change a virtual device number for a virtual device, make sure you assign the virtual device a virtual device number that is not currently being used. (If you do specify a virtual device number that is in use, CP sends you a message to that effect and terminates the command).

To find out what virtual device numbers are being used, enter:

```
query virtual all
```

To redefine the virtual device number of a specific virtual device, enter:

```
redefine vdev1 as vdev2
```

Where:

vdev1
is the current virtual device number.

vdev2
is the new virtual device number.

Example: Redefining a Virtual Device's Device Number

If you want to change the virtual device number of a virtual device from 11A to 22B, enter:

```
redefine 11a as 22b
```

DASD

There are three types of DASD space your virtual machine can have:

- **Dedicated DASD**—This is a real DASD that CP dedicates to your virtual machine. For CP to permanently dedicate a DASD to your virtual machine, the DASD must be defined in your virtual machine's user directory entry. To temporarily dedicate a DASD to your virtual machine, use the ATTACH command as described in "Attaching DASD" on page 51.
- **Minidisks**—Minidisks are all or part of CP-controlled DASDs that CP provides for your virtual machine. Minidisks are defined in your virtual machine's user directory entry. They are also sometimes called permanent minidisks (to distinguish them from temporary disks) or user minidisks. The operating system you run manages minidisks as it does DASDs, although CP must perform the actual I/O operations to and from the minidisk.

A full-pack minidisk is a minidisk that occupies an entire DASD. A full-pack overlay allows for access to the whole DASD, even though it is broken up into minidisks.

- **Temporary disks (T-disks)**—This is temporary disk space that you define using a CP command.

Your virtual machine may also have access to **virtual disks in storage**, which are temporary FBA minidisks allocated from host storage instead of being mapped to a real DASD. Virtual disks in storage may operate faster than other minidisks because the I/O operations are eliminated. A virtual disk in storage is defined either by a statement in a virtual machine's directory entry or by a CP command. A virtual disk in storage defined in the directory is shareable, and is created when the first user links to it and destroyed when the last user detaches it or logs off.

Displaying the Status of Your Virtual Machine's DASD

To display the status of your virtual machine's DASD (dedicated DASD, minidisks, T-disks, and virtual disks in storage), enter:

```
query virtual dasd
```

or

```
query virtual dasd details
```

Where *details* is an optional parameter that displays additional information for any of your DASDs that are attached to cached control units.

To display the device characteristics of a dedicated DASD, full-pack minidisk, user minidisk, T-disk, or virtual disk in storage, enter:

```
query mdisk vdev
```

or

```
query mdisk vdev location
```

Where:

vdev
is the virtual device number of the device for which you want to see information.

location
tells CP you want information about the location and size of the specified device.

For more information about the QUERY VIRTUAL DASD command and the QUERY MDISK command, see *z/VM: CP Commands and Utilities Reference*.

Attaching DASD

To dedicate an offline DASD to your virtual machine, you must:

1. Be authorized to enter class B CP commands. Otherwise, you must request the primary system operator or other authorized user to attach the DASD for you.
2. Know the real device number of the DASD.
3. Enter the following two commands:

```
vary online rdev
attach rdev userid vdev cache control level
```

Where:

rdev
is the real device number of the DASD.

userid
is your user identification.

vdev
is the virtual device number you want to assign the DASD.

cache control level
is the level of subsystem control associated with the DASD. The levels of control are SYSCTL, DEVCTL, and NOCTL. (These levels of control are explained in "Cache Storage" on page 52.)

If you are attaching a DASD that is attached to a cached control unit, you can specify the level of subsystem control you want your virtual machine to have. For more information on how to do this, see *z/VM: System Operation*. When the DASD is attached, CP sends you a message to that effect. You must use the commands appropriate to the operating system you are running to format or otherwise prepare the DASD for use.

Cache Storage

Most modern DASDs are attached to DASD control units that contain a high-speed storage buffer called a cache. Caching (writing to or reading from cache storage) can improve system performance.

If your virtual machine has dedicated DASDs or full-pack minidisks attached to a cached control unit, you have certain levels of subsystem control associated with those DASDs. The levels are as follows:

- The SYSCTL level of control allows the virtual machine to successfully issue CCWs that control DASD subsystem and device resources.
- The DEVCTL level of control allows the virtual machine to successfully issue CCWs that control device resources, but not DASD subsystem resources.
- The NOCTL level of control prevents the virtual machine from successfully issuing CCWs that control either DASD subsystem or device resources.

If your virtual machine has non-full-pack minidisks attached to a cached control unit, you have access authority associated with those minidisks. The access authority is as follows:

- The CACHE access authority means that the minidisk will have access to the cache.
- The NOCACHE access authority means that CP forces read and write I/O to the minidisk to bypass the cache.

Managing Pinned Data

Data that the storage controller cannot remove from cache or NVS because of hardware failures is *pinned data*. As a guest with the appropriate level of control, you can identify the data tracks of one or more DASD that are pinned in the subsystem, by entering the QUERY PINNED command. To do this, enter:

```
query pinned device rdev
```

or

```
query pinned subsystem rdev
```

Where:

device
displays the pinned tracks for the specified device. This is the default.

subsystem
displays the total amount of subsystem cache and NVS containing pinned data.

rdev
is the real device number, a list of real device numbers, or a range of real device numbers of the devices in the subsystems being queried.

Once the pinned data is identified, you can discard that pinned data and free the cache or NVS for other data.

Note: The data is discarded, but not transferred to backing storage.

As a guest with the appropriate level of control, you can discard the pinned data by entering:

```
discard pinned all rdev
```

or

```
discard pinned cachefw rdev
```

Where:

all
: discards all pinned data for the specified device.

cachefw
: discards only pinned cache fast write data for the specified device.

rdev
: is the real device number, a list of real device numbers, or a range of real device numbers for the devices containing data to be discarded.

Delayed Responses to CP Commands

In most instances, CP commands associated with DASD subsystems complete in a short time and the command response is immediately presented. However, for cached DASD controllers, some operations do not immediately complete.

The CP commands that can result in delayed responses are:

- COMMIT
- DESTAGE
- FLASHCOPY
- SET CACHE SUBSYSTEM ON|OFF
- SET CACHE DEVICE OFF
- SET CACHFW OFF
- SET DASDFW OFF
- SET NVS ON|OFF.

The responses to these commands are delayed as follows:

- If the command completes immediately, the normal command response displays.
- If the command does not complete immediately, the following response displays:

  ```
  Command started:  command rdev.
  ```

- If the command completes successfully, the normal command response displays.
- If the command completes unsuccessfully because of errors during asynchronous processing, the following response displays:

  ```
  Command failed:  command rdev.
  ```

- If the cache storage subsystem does not notify CP when it completes a command, the normal completion responses are not generated. Command processing terminates after the following response displays:

  ```
  Command results lost:  command rdev.
  ```

- If the originator of the command is not the system operator, the following response is issued to the system operator when the command completes:

  ```
  Normal command response by userid
  ```

If you are a class B user, you can determine the status of commands entered but not completed. To do this, enter:

```
query pending commands
```

The following response displays for each pending CP command:

```
Command pending:  command rdev.
```

If you are a class B user, you can determine the status of all commands that are pending for a device. To do this, enter:

```
query pending commands allusers rdev
```

Where ALLUSERS causes the following response to display for each pending command:

```
Command pending for userid:  command rdev.
```

Status-Not-as-Required Messages

A subsystem must be in the appropriate state to process a particular command. If it is not in the appropriate state, the command fails with a command rejection with format 0, message F. If a CP subsystem management command fails, the appropriate *status not as required* message (0296E) is displayed to the issuer of the command. There are several messages that report the reason code associated with the “status not as required” message. Several examples follow:

- `Nonvolatile storage is not available`
- `FORCEOFF is invalid in the current state`
- `Recovery action for pinned data is required`
- `Unlike channel types cannot be grouped together`
- `A Diagnostic Control command failed`
- `The control unit had insufficient message buffer space`
- `Pinned tracks are not identifiable because their location and data are in failed nonvolatile storage`
- `Device is undergoing media maintenance`
- `code=nn`

For more information about these messages, see *z/VM: CP Messages and Codes*.

Cross-System Link

If your virtual machine is running on a system that is participating in cross-system link, you might need to know whether one or more volumes are under cross-system link control. To find out, you must be a class A or B user. Enter:

```
xlink check volidn
```

Where *volidn* is a 1- to 6-character entry that identifies the volume or volumes to be checked.

To display link indicators in the map records of all systems on a particular device, enter:

```
xlink display vaddr NOHeader [stack|fifo|lifo] [ cyl(nnnn) ]
```

Where:

vaddr
specifies the virtual address of the device. The device can be one of the following:

- A full-pack minidisk, defining the entire real volume

- The real volume attached to your virtual machine at address *vaddr*
- A minidisk defining the cylinder on the volume containing the CSE area

NOHeader
keeps the descriptive column header information from being put on the CMS stack.

stack
fifo
lifo
places the information in the CMS program stack rather than displaying it at the terminal. STACK is equivalent to FIFO (first in, first out).

cyl (*nnnn*)
limits the display to the information for one minidisk starting on a specific cylinder (*nnnn*).

You must have read access to the device (*vaddr*) to complete the XLINK DISPLAY command successfully.

Note: FBA DASD and virtual disks in storage are not supported for cross-system link.

The normal response to XLINK DISPLAY is:

```
HCPXXL2876I VOLUME volid WAS FORMATTED FOR CSE ON mm/dd/yy AT hh:mm:ss:
            BY USER userid AT SYSTEM sysname

            -CYL- READ/WRITE FLAGS ON volid (CYL=c TRK=t
            RECL=l RECS=r SYSTEMS=s)

            ccc      mode      sysname2     ...   ...   ...
                               ...          ...   ...   ...
                               ...          ...   ...   ...

                           -or-

            ---- NO LINK FLAGS ON THIS VOLUME
```

Where:

volid
identifies the volume of the specified device.

userid
identifies the name of the user who formatted the CSE area on volume *volid*.

sysname
identifies the system where the CSE area for this volume was formatted.

c
identifies the cylinder number where the CSE area was formatted.

t
identifies the track number of the CSE area.

l
identifies the number of cylinders made available for XLINK use. For count-key-data (CKD) devices, it is also the length of the XLINK map records.

r
identifies the number of XLINK records written on the CSE area.

s
identifies the number of systems that can share this DASD volume.

ccc
identifies the starting cylinder of an MDISK on volume *vvvvvv* that has link flags outstanding by one or more systems.

mode
identifies the link mode currently in effect for this MDISK. The mode values are:

Read on
Read-only access.

Write on
Read and write access.

SRead on
Read-only access; negates all write-access requests.

SWrite on
Read and write access; negates all write-access requests.

ERead on
Read-only access; negates all other access requests.

EWrite on
Read and write access; negates all other access requests.

sysname2
identifies the name of the system that has read or write links to the MDISK that begins on cylinder *ccc*.

Sharing DASD

Reserve/release is a function that allows your virtual machine to share a DASD with another virtual machine or with a system running on another processor. For more information about reserve/release, see *z/VM: System Operation* and *z/VM: CP Planning and Administration*.

For cross-system link, use the XLINK FORMAT utility to prepare a DASD volume for sharing. XLINK FORMAT initializes the CSE area on the volume:

```
xlink format vaddr volid
```

Where:

vaddr
specifies the virtual address of the device. The device can be one of the following:

- A full-pack minidisk, defining the entire real volume
- The real volume attached to you at address *vaddr*
- A minidisk defining the cylinder on the volume containing the CSE area

volid
specifies the volume identifier that must match the volume identifier on the device.

Note:

1. Virtual disks in storage are not supported for cross-system link.
2. The volume label on the disk is checked and must match the *volid*.
3. You must have write access to device *vaddr*.
4. It is recommended that you allocate the CSE area as PERM. If the volume is CP-owned, allocate it as PERM space with the CPFORMAT command.
5. The volume does not need to be reformatted if it has a CSE area with the ECKD format. ECKD format exists if the volume is ECKD capable and was last formatted from a z/VM release 1.1 or later system.
6. Before using the stable and exclusive link modes to link to a minidisk that is controlled by cross-system link, the CSE area on the volume where the minidisk is located might need to be reformatted with the XLINK FORMAT command.

The following are responses from XLINK FORMAT:

```
HCPXXL2868I XLINK FORMAT volid - CYL=cccc TRK=tt RECL=llll
            RECS=rr SYSTEMS=ss
```

Where:

volid
identifies the volume of the specified device.

cccc
: identifies the cylinder number where the CSE area was formatted.

tt
: identifies the track number of the CSE area.

llll
: identifies the number of cylinders made available for XLINK use. For CKD devices it is also the length of the XLINK map records.

rr
: identifies the number of XLINK records written on the CSE area.

ss
: identifies the number of systems that can share this DASD volume.

```
HCPXXL2860I WARNING!  IMPROPER USE OF THE XLINK FORMAT COMMAND WILL CAUSE
            LOSS OF MINIDISK DATA.  BE SURE THAT VOLSER volid
            IS NOT ON-LINE TO ANY ASSOCIATED SYSTEMS.

            CONTINUE?  ENTER YES OR NO.
```

If you enter YES, the following response is issued:

```
HCPXXL2869I CSE XLINK FORMAT COMPLETE
```

The following response occurs when you enter the XLINK FORMAT command on a system that is not participating in cross-system link:

```
HCPXXL2879I CANNOT VERIFY CSE LINK TABLE (HCPSYSTB) WITH CP
```

Linking to Other Users' Minidisks

To link to another user's minidisk, use the LINK command.

Note: You can link to another user's virtual disk in storage in the same manner, if the virtual disk in storage is defined in the directory and not by the DEFINE command.

Before you can link to a minidisk, you must know:

1. The read, write, or multiple passwords (if any) of the minidisk.
2. The virtual device number of the minidisk to which you want to link (as defined in the user directory entry of the minidisk owner).
3. The virtual device number you want to assign to the minidisk. You can assign the minidisk any virtual device number as long as the number you assign is not the same as the virtual device number of any of your other I/O devices. (If you specify a duplicate virtual device number, CP sends you a message to that effect and terminates the command). To find out the virtual device numbers you are currently using, enter:

   ```
   query virtual all
   ```

To link to another user's minidisk, enter:

```
link to userid vdev1 as vdev2 mode
```

Where:

userid
: is the logon ID of the virtual machine user who owns the minidisk. If you own the minidisk, specify your own logon ID or an asterisk (*).

vdev1
: is the virtual device number of the minidisk that you want to access, as defined in *userid*'s directory entry.

vdev2
is the virtual device number you want to assign to the minidisk.

mode
determines how you want to access the minidisk. Valid modes include:

R
This gives you read-only access to the minidisk, unless another user has write mode (W, M), exclusive mode (ER, EW), or stable write mode (SW or SM) access to the disk.

RR
This gives you read-only access to the minidisk, unless another user has exclusive mode (ER, EW) access to the disk.

W
This gives you write access to the minidisk, unless another user has existing access to the disk.

WR
This gives you write access if no other user is linked to the minidisk. If another user is linked to the minidisk and the user does not have exclusive mode (ER, EW) access to the disk, you get read-only access.

M
This gives you multiple-write access, unless another user has write mode (W, M), stable mode (SR, SW, SM), or exclusive mode (ER,EW) access to the disk.

MR
This gives you write access, unless another user has write mode (W, M), stable mode (SR, SW, SM), or exclusive mode (ER, EW) access to the disk. If write mode (W, M) or stable mode (SR, SW, SM) access is established, you get read-only access. If exclusive mode (ER, EW) access is established, you are denied access.

MW
This gives you multiple-write access, unless another user has stable mode (SR, SW, SM) or exclusive mode (ER, EW) access to the disk; if so, access to the disk is denied.

SR
This gives you stable read-only access, unless another user has write mode (W, M), stable write mode (SW, SM), or exclusive mode (ER, EW) access to the disk. All requests for write access to the disk with an existing SR mode access are denied.

SW
This gives you stable write access, unless another user has an existing access to the disk. All requests for write access to the disk with an existing SW mode access are denied.

SM
This gives you stable multiple access, unless another user has write mode (W, M), stable mode (SR, SW, SM), or exclusive mode (ER, EW) access to the disk. All requests for write access to the disk with an existing SM mode access are denied.

ER
This gives you exclusive read-only access, unless another user has access to the disk. All requests for access to the disk with existing exclusive mode (ER, EW) access are denied.

EW
This gives you exclusive write access, unless another user has access to the disk. All requests for access to the disk with existing exclusive mode (ER, EW) access are denied.

Note: For information on other access modes you can specify with the LINK command, see *z/VM: CP Commands and Utilities Reference*.

Access-mode checking ignores conflict between a full-pack minidisk and any smaller minidisk on that same volume. A read/write full-pack minidisk and a read/write user's A-disk on the same DASD volume can be held as exclusive (each linked in EW mode). Although the first user links exclusive, that user cannot prevent the other from establishing a simultaneous write access. This situation is only possible if one defines a full-pack minidisk that overlaps other smaller minidisks on the same DASD volume.

In order to prevent conflicting access and to provide full data integrity, you must take the following precautions, do not:

- Define overlapping minidisks.
- Grant DEVMAINT authority to allow full-pack overlays.
- Allow the primary system operator to use DEFINE MDISK.

Note: If an external security manager (ESM) is installed, you may not be authorized to enter the LINK command for all minidisks and all access modes. The ESM may downgrade certain requests for write access to read access. For additional information, contact your security administrator.

Example: Getting Read-Only Access to Another User's Minidisk

If you want read-only access to user ID JENKINS's 222 disk as your 333 disk, enter:

```
link to jenkins 222 as 333 rr
```

After you enter the LINK command, CP prompts you for the read password (if any) of JENKINS's 222 disk.

Note: If another user is already linked exclusively to that disk, you are not given access.

Example: Getting Write Access to Another User's Minidisk

If you want write access to user ID JENKINS's 222 disk as your 333 disk, enter:

```
link to jenkins 222 as 333 wr
```

After you enter the LINK command, CP prompts you for the write password (if any) of JENKINS's 222 disk.

Note: If another user is already linked exclusively to that minidisk, you are not given access. If another user is already linked in any other mode to that disk, you are given read access.

Example: Getting Stable Write Access to Another User's Minidisk

If you want stable write access to user ID JENKINS's 218 disk as your 334 disk, enter:

```
link to jenkins 218 as 334 sw
```

After you enter the LINK command, CP prompts you for the write password (if any) of JENKINS's 218 disk.

Note: If another user is already linked (in any mode) to that minidisk, you are not given access.

Example: Getting Exclusive Write Access to Another User's Minidisk

If you want exclusive write access to user ID JENKINS's 298 disk as your 287 disk, enter:

```
link to jenkins 298 as 287 ew
```

After you enter the LINK command, CP prompts you for the write password (if any) of JENKINS's 298 disk.

Note: If another user is already linked (in any mode) to that minidisk, you are not given access.

Adding Temporary Disk Space

To add temporary disk space to your virtual machine you must (1) define a temporary disk (T-disk), and (2) use the commands of the operating system you are running to format it. The temporary disk you define remains attached to your virtual machine until you log off or until you detach the disk using the DETACH command.

To define a temporary disk, enter:

```
define txxxx as vdev cyl ncyl
or
define txxxx as vdev blk nblk
```

Where:

txxxx
: is the type of disk space you want (3390, 3380, 3995, etc. (CKD) or valid FB-512 (FBA) types.

vdev
: is the virtual device number you assign to the temporary disk. (Specify any virtual device number you did not already use. If you specify a duplicate virtual device number or a virtual device number above X'FFFF', CP sends you a message to that effect and terminates the command.)

ncyl
: is the size of this tdisk in cylinders (valid only for CKD and ECKD tdisks).

nblk
: is the size of this tdisk in blocks (valid only for FBA tdisks).

Note: The type of disk space you specify must already be defined to CP as temporary disk space. (Defining temporary disk space to CP is part of system installation.) If the type of space you define is not available, CP sends you an error message.

After you define the temporary disk, you must format it using the command appropriate for the operating system you are running.

Example: Defining Temporary Disk Space

Assume that you want to define 5 cylinders of 3380 temporary disk space or 12 blocks of FBA temporary disk space and that you want to assign this temporary disk virtual device number 888. To do this, enter:

```
define t3380 as 888 cyl 5
define tfb-512 as 888 blk 12
```

FBA temporary disks are rounded up to pages (8 512-byte blocks per page). Therefore, the FBA temporary disk actually created in this case is 16 blocks.

Defining a Virtual Disk in Storage

You can use the DEFINE command to add a virtual disk in storage to your virtual machine. Because a virtual disk in storage is allocated in host storage and not mapped to a real DASD, it is temporary and fast. Unlike a virtual disk in storage defined in the directory, which is shareable, a virtual disk in storage created with the DEFINE command is private. The virtual disk in storage remains attached to your virtual machine until you log off or detach it using the DETACH command.

To define a virtual disk in storage, enter:

```
define vfb-512 as vdev blk nnnnnnn
```

Where:

vfb-512
: indicates that the virtual disk in storage simulates an FBA minidisk. Having a real FBA DASD in your system configuration is not required.

vdev
: is the virtual device number you assign to the virtual disk in storage. (Specify any virtual device number you did not already use. If you specify a duplicate virtual device number or a virtual device number above X'FFFF', CP sends you a message to that effect and terminates the command.)

nnnnnnn
: is the size of the virtual disk in storage in 512-byte blocks. The maximum size is 4194296 blocks.

After you define the virtual disk in storage, you must format it using the command appropriate for the operating system you are running.

Example: Defining a Virtual Disk in Storage

Assume that you want to define a 165-block virtual disk in storage and assign it virtual device number 799. To do this, enter:

```
define vfb-512 as 799 blk 165
```

Because space for a virtual disk in storage is allocated in 8-block pages, the size of the virtual disk in storage that is created is rounded up to the nearest page (168 blocks). You can use all of the 168 blocks.

Example: Creating a Temporary Disk

Assume that you want to create a five-cylinder temporary disk. You have virtual device address 888 and file mode F available, so you decide to use these for your temporary disk. To create the temporary disk, your session might look like this:

```
Ready;
define t3380 as 888 cyl 5
DASD 0888 DEFINED
Ready;
format 888 f (blksize 4k
DMSFOR603R FORMAT will erase all files on disk F(888). Do you wish to continue?
 Enter 1 (YES) or 0 (NO).
1
DMSFOR605R Enter disk label:
tmpdsk
DMSFOR733I Formatting disk F
DMSFOR732I 5 cylinders formatted on F(888)
Ready;
```

Detaching DASD

To detach a dedicated DASD, minidisk, T-disk, or virtual disk in storage from your virtual machine, enter:

```
detach vdev
```

Where *vdev* is the virtual device number of the device you want detached.

Note: Keep in mind that the LOGOFF command also detaches and destroys a T-disk or a private virtual disk in storage (created with the DEFINE command). If the virtual disk in storage is shareable (defined in the directory), detaching it or logging off destroys the the virtual disk in storage only if you are the last user.

Querying Fenced Components

Fencing is the isolation (by partitioning) of failing components of a DASD subsystem by hardware, microcode, or system software on a cached storage subsystem. Once the component is fenced, it is no longer available for any I/O activity until it is unfenced. As the operator of a guest with the appropriate level of control, you can display the fenced components, if any exist, by entering:

```
query fences rdev
```

This tells you if the subsystem is running in a degraded mode that requires service. To unfence the fenced components use ICKDSF.

Working with Virtual Communication Lines

The following section explains how to display information about, add, and dial your virtual communication lines.

Displaying Information about Virtual Communication Lines

To display information about your virtual communication lines, enter:

```
query virtual lines
```

CP responds with a message similar to the following:

```
LINE vdev ON LINE rdev
```

Where:

vdev
is the virtual device number of the communication line.

ON LINE rdev
shows the real device that is dialed into or attached to this virtual machine.

or

```
LINE vdev ENABLED
          DISABLED
```

Where:

vdev
is the virtual device number of the communication line.

ENABLED
shows that the virtual line is enabled.

DISABLED
shows that the virtual line is disabled.

If your virtual console is conmode 3215, refer to the *z/VM: CP Commands and Utilities Reference* for more information.

Adding and Dialing a Virtual Communication Line

If you want to add a virtual communication line to your virtual machine configuration, enter:

```
define line as vdev ibm1
```

or

```
define line as vdev tele2
```

Where:

vdev
is the virtual device number of the line you are defining.

ibm1
is the virtual device type of the line you are defining. This is a 2741, 3767, or equivalent device. It is the default.

tele2
is the virtual device type of the line you are defining. This is a 3101, 3151, 3161, 3162, 3163, or equivalent device.

CP responds with a message similar to the following:

```
LINE vdev DEFINED
```

Once the line is defined, you must enter the CP DIAL command from a real device in order to logically connect the line to your virtual machine. The device from which you enter the DIAL command must be supported by the operating system that is running in your virtual machine.

To invoke the DIAL command, you must first press the appropriate key (see the displayed messages given for the logon procedure in "Step 1. Log On to z/VM" on page 17) and then enter:

```
dial userid vdev
```

Where:

userid
is the identification of the virtual machine in which the multiple-access operating system is running.

vdev
is the device number of the local virtual display to which the connection is to be made.

Note: The DIAL command may be disabled if an ESM is installed.

If the user ID is recognized, logged on, or disconnected, and has a virtual communication line available at the specified address, the system responds with a message similar to the following:

```
DIALED TO userid vdev
```

From this time until the terminal is dropped from the virtual machine, the terminal is entirely under the control of that virtual machine.

Dropping a Dialed Connection to a Virtual Communication Line

When you want to drop a dialed connection to a local virtual display, enter:

```
reset vdev
```

Where:

vdev
identifies the virtual device number of the device to be reset.

Working with 3270 Display Devices

The following sections describe how to display information about, add, and detach your virtual machine's 3270 displays and 3270 printers.

Displaying Information about 3270 Display Devices

To display information about your 3270 displays and 3270 printers, enter:

```
query virtual graf
```

To display the extended color and extended highlighting values in effect for your virtual machine console, enter:

```
query screen
```

CP responds with a message similar to the following:

```
CPOUT     color   exthilight        VMOUT    color   exthilight
INREDISP  color   exthilight        INAREA   color   exthilight
STATAREA  color   exthilight
```

Where:

color
is the color value currently in effect for your virtual console.

exthilight
is the extended highlight value currently in effect for your virtual console.

The extended color and highlighting values are set either by the SCREEN statement in your directory entry or when you enter the SCREEN command.

If you do not have a SCREEN directory statement in your directory entry, and you have not entered the SCREEN command during the current session, the response to the QUERY SCREEN command is

```
DEFAULT
```

for the color value and

```
NONE
```

for the extended highlight value.

For more information on the SCREEN command, see *z/VM: CP Commands and Utilities Reference*.

Extended Color and Highlighting Support

z/VM supports extended color (7-color) mode, extended highlighting, and programmed symbol sets for 3270 models for which these features are available. Extended color and extended highlighting values can be set for the three screen areas defined by CP (the user input area, the output display area, and the screen status area) and for the following types of output: CP output, virtual machine output, and input redisplay. The colors supported are blue, red, green, yellow, turquoise, pink, and white. Supported extended highlighting values are blinking, underline, and reverse video.

The color and highlighting values to be used can be set to initial values by way of a SCREEN statement in the virtual machine's VM directory entry or can be set at any time during the terminal session by way of the SCREEN command. The default colors for input and output are green (not highlighted) and white (highlighted). The default extended highlighting setting is NONE. The QUERY SCREEN command can be used to display the current extended color and extended highlighting settings.

Note that CP accepts extended color and highlighting settings even from a virtual operator's console that does not support one or both of these features. If the user then disconnects and logs on to a virtual operator's console that does support these features, the extended color and highlighting settings take effect.

APL Character Support

CP supports the use of APL characters on 3270 displays and 3270 printers for which an APL feature is available. When the APL feature is present on a 3270 display, it enables a user to enter and display the full APL character set, which includes the standard upper and lower case EBCDIC characters. The standard dual-case 3270 character set is combined with characters unique to APL, including compound and overstruck symbols. An underscored uppercase alphabet can also be used.

To cause CP to use the APL character translation tables instead of the EBCDIC tables, the user must enter the CP command TERMINAL APL ON. Thereafter, when the APL ON/OFF key is pressed, APL characters may be entered. While the TERMINAL APL ON setting is in effect, APL characters are always displayed, regardless of the setting of the APL ON/OFF key.

A local or remote 3270 display with the APL feature can be used to interact with the VS APL licensed program in a CMS virtual machine. Hard copy support for a local or remote 3270 with the feature is also supported when the feature is installed on a 3270 printer.

When VS APL is to be used, CMS must be IPLed in a logged-on virtual machine. The APL-CMS interface program must be called by running the APL EXEC. The interface program prompts the user to press the APL ON/OFF key so that APL characters can be entered. The interface program sends the TERMINAL APL ON command to activate the APL character translation tables included in CP and sends DIAGNOSE code X'54' to cause an external interrupt to be reflected to VS APL when the PA2 key is pressed. The VS APL program is then called by the interface program and a ready message is sent.

To return control to CMS, the APL command OFF must be sent. This command calls the APL-CMS interface program, which issues the TERMINAL APL OFF command, prompts the user to press the APL ON/OFF key to turn off the capability of entering APL characters, and returns to CMS.

Text Character Support

CP supports the use of text characters on 3270 displays and 3270 printers for which a text feature is available. When the text feature is present on a 3270 display, it enables a user to enter and display all of the special text characters present in the TN print train in addition to the standard 3270 characters. When the feature is present on one of the supported 3270 printers, hard copy of a 3270 display containing the special text characters can be printed.

To activate the feature, the TEXT ON/OFF feature must be pressed and the CP command TERMINAL TEXT ON must be entered. The TERMINAL TEXT ON command automatically forces a TERMINAL APL OFF command to take effect and causes CP to use the translation tables required by the text feature instead of the usual translation tables.

The TERMINAL TEXT OFF command deactivates the text feature. When a TERMINAL APL ON command is entered, a TERMINAL TEXT OFF command is automatically forced.

Double-Byte Character Set (DBCS)

Virtual machines can support DBCS output data by sending an I/O request to a channel program. CP will ensure that a console is DBCS capable and will format screen data accordingly. DBCS and SBCS data will be properly displayed on the screen. The user is not responsible for inserting new line (NL) characters into the data.

Virtual machines can also use DIAGNOSE code X'58' operation code X'49' in conjunction with DBCS support. Double-byte character set (DBCS) data is supported for output in line mode while CP manages the display session. If this method is used, the user is responsible for inserting NL characters into the data to ensure proper formatting. The display must be defined as a 3215.

z/VM has input support for DBCS. If your terminal has the capability of using national languages with double-byte character sets (DBCS), such as Kanji, you will have this new feature. With this new support a terminal user can input mixed DBCS data on the input line of a line mode terminal screen. CP will pass the data to a guest running in the virtual machine (for example, CMS). If there is no guest running in the virtual machine, CP will handle the input data. Also, CMS can pass valid DBCS output strings to CP.

Some CP commands have been changed to accept DBCS data in the text portion of the command. CMS commands that allowed DBCS data and could be previously entered in fullscreen CMS can now be entered in line mode.

If the DBCS data is contained in a CP command which does not support DBCS, CP will return an error message indicating an incorrect command or command option.

Before logging on, you can use DBCS text in the text portion of the MESSAGE command after clearing the logo screen. This is all you can do with DBCS before logging on.

DBCS support in REXX lets your application specify REXX identifiers, such as variable names, labels, symbols and compound symbols in DBCS.

Adding and Dialing 3270 Displays

By temporarily defining one or more 3270 displays for your virtual machine, you can allow other users to access the operating system you are running. To temporarily add a display to your virtual machine I/O configuration, enter:

```
define graf as vdev
```

Where *vdev* is the virtual device number you want to assign to the display.

After you define this display, use the commands of the operating system you are running to ready the display for use. Authorized users can then use any 3270 display to access the operating system you are running. This is accomplished by either of the following:

- Entering the DIAL command from the COMMAND field of the logo
- Clearing the logo screen first (press ENTER) and then entering the command on the input line.

The DIAL command is entered as follows:

```
dial userid
```

Where *userid* is your logon identification.

After the user enters this command, the logo for your operating system appears on the display. The user has access to your operating system until you log off, re-IPL the operating system you are running, or enter the RESET command to drop the dialed connection to your temporary display.

Note:

1. A SNA/CCS terminal user cannot dial into or log on to the VTAM service machine controlling the terminal.
2. The DIAL command can only be used on certain SNA/CCS terminals. The terminal must be controlled by VSCS, as VCNA does not support the DIAL command.
3. The DIAL command may be disabled if an ESM is installed.

Dropping a Dialed Connection to a Display

If you (the virtual machine operator) need to drop the dialed connection to the temporary display (for example, if the user no longer needs the display), enter:

```
reset vdev
```

Where *vdev* is the virtual device number of the display.

If the user no longer needs access to your operating system and wants to drop the dialed display, the user may do so by resetting the TEST/NORMAL switch on the terminal. Press the switch to the TEST position and then back to the NORMAL position. Users who have terminals without a TEST/NORMAL switch must use the POWER OFF/POWER ON switch. Press the switch to the POWER OFF position and then back to the POWER ON position. If the operating system you are running is z/VM and the dialed display is not logged on, the UNDIAL command can be issued from the dialed display to drop the connection. For more information on the UNDIAL command, see *z/VM: CP Commands and Utilities Reference*.

When the dialed connection is dropped, a message is sent to the user. Following the DROP response, the user is prompted with one of the following messages, depending on the type of terminal being used:

Type of Terminal
    Message

3270

```
Press ENTER or CLEAR key to continue
```

SNA/CCS 3270

```
Press ENTER key to continue
```

When the appropriate key is pressed, the logo appears on the screen.

Note:

1. For terminals not listed above, control automatically returns to CP, and the z/VM online message is displayed.
2. For more information on the DROP response, see the DIAL command in *z/VM: CP Commands and Utilities Reference*.

Attaching a 3270 Printer

1. To attach a 3270 printer to your virtual machine, you must:
   a. Be authorized to enter class B CP commands. Otherwise, you must request the primary system operator or other authorized user to attach the 3270 printer for you.
   b. Know the real device number of the 3270 printer.
   c. Determine the current status of the device. To do this, enter:

      ```
      query rdev
      ```

      Where *rdev* is the real device number of the 3270 printer.

      CP's response is that the device is (1) attached to another user, (2) online and enabled, (3) online but disabled, or (4) offline.
2. What you do next depends on CP's response:
   a. If the device is attached to another user, you must contact that user and ask that the device be detached. The device can only be attached to one user at a time.
   b. If the device is online and enabled, enter the following commands:

      ```
      disable rdev
      attach rdev to userid as vdev
      ```

      Where:

      rdev
      is the real device number of the 3270 printer.

      userid
      is your user identification.

      vdev
      is the virtual device number you want to assign the 3270 printer.
3. If the device is online but already disabled, enter the following command:

   ```
   attach rdev to userid as vdev
   ```

   Where:

   rdev
   is the real device number of the 3270 printer.

   userid
   is your user identification.

   vdev
   is the virtual device number you want to assign the 3270 printer.
4. If the device is offline, enter the following commands:

   ```
   vary online rdev
   attach rdev to userid as vdev
   ```

   Where:

   rdev
   is the real device number of the 3270 printer.

   userid
   is your user identification.

   vdev
   is the virtual device number you want to assign the 3270 printer.

After the 3270 printer is attached, use the commands of the operating system you are running to control it.

Detaching a 3270 Printer

To detach a 3270 printer from your virtual machine, enter:

```
detach vdev
```

Where *vdev* is the virtual device number of the 3270 printer you want to detach.

Working with Tape Drives

To check the status of any tape drive attached to your virtual machine, enter:

```
query virtual tape
```

Attaching a Tape Drive

To attach an offline tape drive to your virtual machine, you must:

1. Be authorized to enter class B CP commands. Otherwise, you must request the primary system operator (or other authorized user) to attach the tape drive for you.
2. Know the real device number of the tape drive.
3. Enter the following two commands:

   ```
   vary online rdev
   attach rdev to userid as vdev
   ```

   Where:

   rdev
   is the real device number of the tape.

   userid
   is your user identification.

   vdev
   is the virtual device number you want to assign the tape.

CP sends you a message when the tape drive is attached. Once the tape drive is attached, manage it using the commands of the operating system you are running.

Mounting a Tape

Once you have attached your tape drive, you can mount a tape. You do not need to use CP commands to mount a tape.

Rewinding a Tape

To rewind a tape on a tape drive that is dedicated to your virtual machine, enter:

```
rewind vdev
```

Where *vdev* is the device number of the tape drive on which you want the tape rewound.

Detaching a Tape Drive

To detach a tape drive from your virtual machine, enter:

```
detach vdev
```

Where *vdev* is the device number of the tape drive you want detached.

Transferring Control of a Tape Drive

As a Class B user, you can transfer ownership of dedicated tape drives among users running on the same system by entering the GIVE command. You can transfer the tape drive to another user or you can transfer it from one user to a third user. As a privileged user, you obtain dedicated use of a tape drive by entering the ATTACH command.

To reclaim ownership of the tape drive once the receiver is finished with it, you specify the RETURN option for the GIVE command. If you give the tape drive with the RETURN option, also specify whether the tape should be rewound and unloaded, or left in its current position. You do this with the UNLOAD and LEAVE options, respectively. You can also specify whether the receiver is to have read/write or read/only access to the given tape drive.

The receiver logically owns the tape drive until it is detached through the DETACH command or the receiver logs off the system. If the virtual machine that was given the tape drive has class B authorization and the tape drive was given without the RETURN option, the receiver can give the tape drive to a third party. If RETURN option was specified, the second GIVE is rejected and a message is issued.

Note: In an installation with more than one tape server virtual machine, one of the servers may omit the RETURN option. In this situation, one server machine can give a tape drive to another without the RETURN option; the second server machine can then become the owner of the tape drive and can give it to a virtual machine requesting a tape drive.

To cancel a return request for the tape drive, you (as the giver) can enter the DETACH command, specifying the reserved virtual device number, or you can log off your virtual machine. Even if you log off and log on again before the receiver detaches the tape drive, the device is not returned to you. When the receiver enters the DETACH command, the tape drive is returned to either the giver or the system, depending on the current RETURN status of the tape drive.

When the DETACH command is entered (by a class B user) to remove a tape drive, the LEAVE or UNLOAD option specified on DETACH overrides any option specified on the GIVE command.

Example: Transferring a Tape Drive with Read/Write Access

The receiver wants to receive control of a tape drive with read/write access. You want the tape returned in its current position. To transfer the tape you may do the following:

1. You verify that all tape drives are dedicated to the service virtual machine.
2. The receiver logs on and requests a tape.
3. You mount the tape, and read and verify the label.
4. You attach the tape to the receiver by entering:

   ```
   give vdev1 userid vdev2 return leave
   ```

   Where:

   vdev1
   is the virtual address of the tape to be transferred.

   userid
   is the ID of the virtual machine that receives the tape drive.

   vdev2
   is the virtual address where the tape is to be attached to the target virtual machine.

   return
   specifies that the invoker wishes to regain control of the transferred tape drive when detached by the receiver.

   leave
   specifies that the tape is to be left in its position when it is detached and returned to the original owner.

The tape will be returned to you and left in the current position. By specifying RETURN and LEAVE, you verify the integrity of the label.

5. The receiver accesses the tape (and reads and writes).
6. When finished with the tape, the receiver detaches the tape from the virtual machine by entering:

   ```
   detach vdev
   ```

   Where *vdev* is a virtual device number, a list of virtual device numbers, or a range of virtual device numbers to be detached from the virtual machine.
7. Control of the tape drive returns to you with the tape left in its current position.
8. You rewind the tape and read the label to make certain the receiver did not write over the label.
9. Unload the tape.

Example: Transferring a tape drive with Read/Only Access

The receiver wants control of a tape drive. For the particular application, read/only access is necessary. You want the tape returned and rewound. To transfer the tape you may do the following:

1. You verify that all tape drives are dedicated to the service virtual machine.
2. The receiver logs on and requests a tape.
3. You mount the tape, and read and verify the label.
4. You attach the tape to the receiver by entering:

   ```
   give vdev1 userid vdev2 return unload r/o
   ```

   Where:

   vdev1
   : is the virtual address of the tape to be transferred.

   userid
   : is the ID of the virtual machine receiving the tape drive.

   vdev2
   : is the virtual address where the tape is to be attached to the target virtual machine.

   r/o
   : attaches the tape to the receiver in read-only mode. Attempts to write to the tape drive result in an error.

   The tape will be returned to you, unloaded, and rewound.
5. The receiver accesses the tape (and reads from it).
6. When the receiver is finished with the tape, he detaches the tape from the virtual machine by entering:

   ```
   detach vdev
   ```

   Where *vdev* is a virtual device number, a list of virtual device numbers, or a range of virtual device numbers to be detached from the virtual machine.
7. Control of the tape drive returns to you with the tape unloaded and rewound.

Working with Virtual Channel-to-Channel Adapters

A virtual channel-to-channel adapter (CTCA) provides a communications link between two virtual machines, regardless of whether your installation has defined a real channel-to-channel adapter. To check the status of any virtual channel-to-channel adapters in your virtual machine configuration, enter:

```
query virtual ctca
```

Note: z/VM supports CTCAs designed after 1981. For more information on what commands are supported by the z/VM VCTCA, see the *IBM Channel-to-Channel Adapter Manual*.

Coupling Two Virtual Machines

To define a virtual channel-to-channel adapter between your virtual machine and another user's virtual machine, you and the other virtual machine user must:

1. Be logged on.
2. DEFINE CTCA or DEFINE 3088 commands to define virtual channel-to-channel adapters.
3. Enter the COUPLE command to connect these two channel-to-channel adapters.

To define a virtual channel-to-channel adapter, enter:

```
define ctca as vdev user userid
```

or

```
define 3088 as vdev user userid
```

Where:

ctca
: indicates that you are defining a channel-to-channel adapter.

3088
: indicates that you are defining a single logical channel-to-channel adapter of a 3088 Multisystem Channel Communication Unit.

vdev
: is the virtual device number of the CTCA or 3088.

userid
: is the logon identification of the user who wants to connect to the CTCA or 3088. You may use an asterisk (*) to specify your own user ID.

To define a CTCA or 3088 without specifying a user ID, enter:

```
define ctca as vdev
```

or

```
define 3088 as vdev
```

Where:

ctca
: indicates that you are defining a channel-to-channel adapter.

3088
: indicates that you are defining a single logical channel-to-channel adapter of a 3088 Multisystem Channel Communication Unit.

vdev
: is the virtual device number of the CTCA or 3088.

Since no user ID is specified, any user ID is able to connect to this CTCA or 3088.

To connect two virtual channel-to-channel adapters, enter:

```
couple vdev1 to userid vdev2
```

Where:

vdev1
: is the virtual device number of the issuer's channel-to-channel adapter.

userid
can be either *your* logon identification or *another* user's logon identification.

vdev2
is the virtual device number of the other user's channel-to-channel adapter.

Note: If an ESM is installed, you may not be authorized to enter the COUPLE command. For additional information, contact your security administrator.

Example: Coupling Two Virtual Machines

Assume that the virtual machine users with user IDs BECKERT and YOUNG want to define a virtual channel-to-channel adapter between their virtual machines. BECKERT wants to assign his channel-to-channel adapter device number 100. YOUNG wants to assign her channel-to-channel adapter device number 200.

BECKERT must enter:

```
define ctca as 100 user young
```

YOUNG must enter:

```
define ctca as 200 user beckert
```

To connect the two channel-to-channel adapters, BECKERT must enter:

```
couple 100 to young 200
```

Or, YOUNG must enter:

```
couple 200 to beckert 100
```

Detaching a Virtual Channel-to-Channel Adapter

To detach a virtual channel-to-channel adapter, enter:

```
detach vdev
```

Where *vdev* is the virtual device number of the channel-to-channel adapter.

Working with Virtual Machine Unit Record Devices

There are two ways for CP to provide your virtual machine with unit record devices (printers, punches, and card readers):

- CP can dedicate real unit record devices to your virtual machine. In this case, your virtual machine has dedicated unit record devices.
- CP can use a method called spooling to perform unit record device I/O operations for your virtual machine without having to dedicate real unit record devices to your virtual machine. In this case, your virtual machine has spooled unit record devices.

How you manage a particular unit record device depends on whether that device is spooled or dedicated. If it is spooled, you must use CP commands to manage the device. See Chapter 7, “Using Spooled Devices to Print, Punch, and Read Information,” on page 75 for information on how to use spooled devices.

If a unit record device is dedicated, use the commands appropriate to the operating system you are running to manage the device.

Determining If Unit Record Devices Are Dedicated

To find out which, if any, of your virtual machine unit record devices are dedicated, enter:

```
query virtual ur
```

The response to this command shows you all your spooled unit record devices and all your dedicated unit record device.

Example: Finding Out about Your Unit Record Devices

Assume you have a spooled reader at virtual device number 000C, a spooled punch at virtual device number 000D, a spooled printer at virtual device number 000E, and a dedicated printer at virtual device number 000F. CP responds to the QUERY VIRTUAL UR command with the following:

```
RDR  000C CL * NOCONT NOHOLD   EOF       READY
     000C 2540         CLOSED    NOKEEP NORESCAN
PUN  000D CL A NOCONT NOHOLD COPY  001    READY FORM STANDARD
     000D TO RSCS     RDR DIST 45D-0253  DEST FLOOR3
     000D FLASH      000 CHAR      MDFY      0 FCB
     000D 2540 NOEOF CLOSED    NOKEEP NOMSG NONAME
     000D SUBCHANNEL=0006
PRT  000E CL A NOCONT   HOLD COPY  001    READY FORM STANDARD
     000E TO NINA     PRT DIST 45D-0253   FLASHC 000 DEST FLOOR3
     000E FLASH          CHAR      MDFY      0 FCB     LPP 060
     000E 1403 NOEOF CLOSED    NOKEEP NOMSG NONAME
     000E SUBCHANNEL=0007
PRT  000F ON DEV   0003
```

The first three unit record devices are spooled devices; therefore, CP's response shows their spooling options. See "Displaying the Spooling Options of Your Reader" on page 98 and "Displaying the Spooling Options of Your Printer or Punch" on page 100 for descriptions of these options.

The last unit record device is a dedicated device. Dedicated devices do not have spooling options, so CP's response simply identifies the real address and the virtual address of the dedicated device. The response

```
PRT  000F ON DEV   0003
```

means that a real printer whose real address is 0003 is dedicated to your virtual machine at virtual device number 000F.

Adding a Dedicated Unit Record Device

To temporarily dedicate a unit record device to your virtual machine, you must first contact the primary system operator. The message you send to the operator should specify the virtual device number you want the operator to assign the device. For example:

```
message operator please attach a 1403 printer to my
        userid at device number 000F
```

Note: If an ESM is installed, you may not be authorized to enter the MESSAGE command. However, messages sent to or from the system operator and messages sent with the ALL option are not subject to authorization checking by the ESM. For additional information, contact your security administrator.

Detaching Unit Record Devices

To detach a dedicated unit record device from your virtual machine, enter:

```
detach vdev
```

Where *vdev* is the virtual device number of the device you want to detach.

Chapter 7. Using Spooled Devices to Print, Punch, and Read Information

Spooling is a method that allows CP to print, punch, and read information for your virtual machine without having to dedicate a real printer, punch, or card reader to your virtual machine. For CP to use spooling, your virtual machine's unit record devices must be spooled devices. Your virtual machine's spooled devices can be permanently defined to CP in your virtual machine's directory entry; you can also use CP commands to define spooled devices temporarily. (If you need to define a spooled unit record device for your virtual machine temporarily, see "Adding Spooled Devices" on page 104.)

This topic contains descriptions about:

- Determining whether a unit record device is spooled
- Differences between using dedicated unit record devices and spooled unit record devices to print, punch, and read information
- Setting the spooling options on a spool file
- Sending spool files to your (or another user's) spooled printer, card reader, or punch
- Changing the processing order of spool files at your spooled printer, punch, or reader
- Deleting spool files that are queued for your spooled printer, punch, or reader
- Saving spool files and system trace files on tape and restoring them
- Setting the spooling options for a spooled unit record device
- Adding and detaching spooled unit record devices.

Determining If Unit Record Devices Are Spooled

To find out which, if any, of your existing unit record devices are spooled, enter:

```
query virtual ur
```

If a unit record device is spooled, CP's response will include two or three lines of spooling options. If a unit record device is dedicated, CP's response will be a single line, as shown in the following example.

Example: Identifying Spooled and Dedicated Devices

Assume you have a spooled reader at virtual device number 000C, a spooled punch at virtual device number 000D, a spooled printer at virtual device number 000E, and a dedicated printer at virtual device number 000F. CP responds to the QUERY VIRTUAL UR command with the following:

```
RDR  000C CL * NOCONT NOHOLD   EOF       READY
     000C 2540         CLOSED    NOKEEP NORESCAN
PUN  000D CL A NOCONT NOHOLD COPY  001    READY FORM STANDARD
     000D TO RSCS     RDR DIST 45D-0253  DEST FLOOR3
     000D FLASH      000 CHAR      MDFY      0 FCB
     000D 2540 NOEOF CLOSED    NOKEEP NOMSG NONAME
     000D SUBCHANNEL=0006
PRT  000E CL A NOCONT   HOLD COPY  001    READY FORM STANDARD
     000E TO NINA     PRT DIST 45D-0253   FLACHC 000 DEST FLOOR3
     000E FLASH          CHAR      MDFY      0 FCB     LPP 060
     000E 1403 NOEOF CLOSED    NOKEEP NOMSG NONAME
     000D SUBCHANNEL=0007
PRT  000F ON DEV   0003
```

The first three unit record devices are spooled devices; therefore, CP's response for them shows their spooling options. See "Displaying the Spooling Options of Your Reader" on page 98 and "Displaying the Spooling Options of Your Printer or Punch" on page 100 for descriptions of these options.

The last unit record device is a dedicated device. Dedicated devices do not have spooling options, so CP's response simply identifies the real address and the virtual address of the dedicated device. The response

```
PRT  000F ON DEV   0003
```

means that a printer whose real address is 0003 is dedicated to your virtual machine at virtual device number 000F.

Spooling—An Overview

To learn how to use spooling to print, punch, and read information, it helps to understand the basic difference between spooled unit record devices and dedicated unit record devices.

Using Dedicated Unit Record Devices

Assume that your virtual machine has dedicated unit record devices. This means that your virtual machine has exclusive use of a real printer, punch, and card reader. You do not need CP commands to manage a dedicated unit record device *unless* you want to attach or detach a dedicated unit record device.

Using Spooled Unit Record Devices

On the other hand, assume that instead of dedicated unit record devices, your virtual machine has spooled unit record devices. This means that your virtual machine does not have exclusive use of a real printer, punch, and card reader. As a result:

- When you print or punch information, the information does not go directly to a real printer or punch. Rather, CP places the information into a *spool file,* which is on a DASD. When the spool file is closed (that is, CP is notified that the spool file contains all the information it is going to contain), and a real printer or punch is available, CP can print or punch the spool file.
- When you read information queued for your spooled reader, what you are actually doing is reading a spool file that CP has placed on a DASD.

If your virtual machine has spooled unit record devices, you may need CP commands *and* the commands of the operating system you are running to manage the device. You need the commands of the operating system you are running to send information to your spooled printer or punch and to read information queued for your spooled reader. You need CP commands to manage your spooled unit record devices and spool files queued for your spooled unit record devices.

Associating Descriptive Information with Spool Files or Devices

You can associate descriptive information with system spool files or spooled devices by using the TAG command.

To display the tag information associated with a particular virtual spool file or spooled device, enter:

```
tag query file spoolid
```

or

```
tag query dev devtype
```

To add the descriptive information for a spool file, enter:

```
tag file spoolid tagtext
```

or, to add descriptive information for a spool device, enter:

```
tag dev devtype tagtext
```

Where:

query
requests that tag information associated with a particular spool file or spooled device be displayed.

dev
is the device to be tagged.

devtype
is one of the following:

PRINTER|PRT
PUNCH|PCH
CONSOLE
vdev

It specifies the spooling device whose output is assigned the tag information. If PRT, PCH, or CONSOLE is entered, all current virtual devices of that type are affected.

FILE *spoolid*
is a closed spool file queued on your reader, printer, or punch queue to which you want to assign tag information. The operand *spoolid* is the spool file identification that CP assigned when the spool file was closed.

tagtext
is the information you want to associate with the specified spool device or spool file.

Example: Display Tag Information for all Current Virtual Printers

You want to know what tag information is associated with all current virtual printers. Enter:

```
tag query prt
```

Example: Add Tag Information for a Spool File

You want to add tag information on commands A through C to the spool file identified by 1594. Enter:

```
tag file 1594 A-C commands
```

Note: If an ESM is installed, you may not be authorized to enter the TAG command. For additional information, contact your security administrator.

Managing Spool Files

The following sections describe how to use CP commands if you need to:

- Determine the spool file limit for the system or a user
- Close spool files
- Display information about your spool files (general, expanded, and print server)
- Print your spool files
- Change the spooling options on a spool file
- Send spool files to other users or other spooled devices
- Change the order of spool files queued for a spooled device
- Purge spool files that are queued for a spooled device
- Save spool files or system trace files on tape
- Restore spool files or system trace files from tape

Determining the Spool File Limit for the System or a User

Use the QUERY MAXSPOOL command to display the spool file limit for the system or for any individual user. To find out your spool file limit, enter:

```
query maxspool *
```

Closing Spool Files

When you print or punch information, CP places it in an open spool file. When you read information, CP reads it from an open spool file. As long as the spool file is open, CP can place information in it or read information from it. When a spool file is closed, CP can process it on a real device or send it to another spooled device (either yours or another user's).

How do you close a spool file? Some operating systems (for example, VSE) can close spool files automatically. If your operating system cannot automatically close spool files, enter:

```
close vdev
```

Where *vdev* is the virtual device number of the device that is processing the spool file.

If you have one or more open reader, printer, punch, or console files, and you want to close all of them, you can enter the corresponding command:

```
close printer
close punch
close reader
close console
```

If you need to determine whether a spool file is closed, use the QUERY PRINTER ALL, QUERY PUNCH ALL, or QUERY READER ALL commands (described in "Displaying Information about Your Spool Files" on page 78). If a spool file is closed, the information under DATE and TIME is the date and time the spool file was closed. If a spool file is open, the information under DATE and TIME indicates that the spool file is open.

Displaying Information about Your Spool Files

The command you use to display information about your spool files depends on the type of information you want to display. To display specific information about your spool file's forms control options and 3800 characteristics, see "Displaying Forms Control Options and 3800 Attributes" on page 80.

To display general information about spool files queued for your spooled reader, printer, or punch, enter the corresponding command:

```
query reader all
query printer all
query punch all
```

In response to these commands, CP displays information similar to the following:

```
ORIGINID FILE CLASS RECORDS   CPY HOLD DATE  TIME      NAME     TYPE     DIST
userid   spid c typ nnnnnnnn  nnn NONE mm/dd hh:mm:ss  filename filetype distcode
                             *nnn USER
                                  SYS
                                  USYS
```

To display information about the security labels of spool files queued for your spooled reader, printer, or punch, enter the corresponding command:

```
query reader seclabel
query printer seclabel
query punch seclabel
```

In response to these commands, CP displays information similar to the following:

```
ORIGINID FILE CLASS RECORDS  CPY HOLD FORM     DEST     SECLABEL
userid   spid a typ nnnnnnnn nnn NONE formname dest     ssssssss
                            *nnn USER
userid   spid a typ ******** *** SYS  ******** ****     ********
                            **** USYS
```

Note: When an external security manager (ESM) is installed and security label checking is enabled, the ESM is called to verify the user's access to each file. If the user is denied access to a specific file, some of the information about that file is hidden from the user. The only fields of the response that will contain information are ORIGINID, FILE, CLASS, HOLD, DATE and TIME. All other fields contain asterisks. If a security label is not assigned to the file, the SECLABEL field contains the word NONE.

Where:

ORIGINID
is the logon identification (user ID) of the file owner or originator. OWNERID display when *userid* or the XFER operand is used, or when a class D user is defaulting to, or specifying, SYSTEM.

FILE
is the spool file's identification number.

CLASS
tells you the class of the spool file and the type of device on which the spool file was created.

RECORDS
is the number of logical records in the spool file.

CPY
is one of the following:

nnn
is the number of copies requested for the spool file. This is shown as a 3-digit number.

****nnn***
an asterisk (*) preceding *nnn* indicates that the 3800 printer should print each page from the spooled output files *nnn* times before going on to the next page.

HOLD
is the status of the file, where:

NONE
means the file is not in hold status.

USER
means the file is in user-hold status.

SYS
means the file is in system-hold status.

USYS
means the file is in both user- and system-hold status.

Note: If the file is in system-hold or user-hold status, you cannot print it.

DATE
is indicated as follows:

mm/dd
is the month and day that the file was created.

TIME
is indicated as follows:

hh:mm:ss
is the time when the file was created in hours:minutes:seconds.

NAME
is the file name of the spool file.

TYPE
is the file type of the spool file.

DIST
is an installation-dependent code (1 to 8 alphanumeric characters) that your installation can use as it wants (for example, to specify where output is sent after CP processes it on a real printer or punch).

SECLABEL
: identifies the security label associated with the file; *ssssssss* is a 1- to 8-character alphanumeric value. The SECLABEL information is displayed only if an ESM is installed, security label checking is enabled, and the SECLABEL option is specified on the QUERY READER command.

Displaying Forms Control Options and 3800 Attributes

To display specific information about forms control options and 3800 attributes of spool files, use the expanded information (EXP) option. Enter one of the following commands:

```
query reader exp
query printer exp
query punch exp
```

In response to these commands, CP displays information similar to the following:

```
ORIGINID FILE CLASS RECORDS  FLASH FCB MDFY  FLSHC LOAD CHARS               SIZE
userid   spid c typ nnnnnnnn ovly  fcb mod n ccc   NO   name name name name size
                                                   BEG
                                                   ANY
```

Where:

ORIGINID
: is the logon identification (user ID) of the file owner or originator. OWNERID display when *userid* or the XFER operand is used, or when a class D user is defaulting to, or specifying, SYSTEM.

FILE
: is the spool file's identification number.

CLASS
: tells you the class of the spool file and the type of device on which the spool file was created.

RECORDS
: is the number of logical records in the spool file.

FLASH
: is the name of the forms overlay frame that is superimposed on the output.

FCB
: is the name of the forms control buffer (FCB) in use.

MDFY
: is the name of the copy modification module and the number (0, 1, 2, or 3) of the copy modification character arrangement table you are using to alter output text.

FLSHC
: is the number of copies printed while the forms overlay frame is in place.

LOAD
: is where the 3800 LOAD CCWs are positioned within the spool file. There are three possible responses:

  NO
  : means that there are no LOAD CCWs in the spool file.

  BEG
  : means that LOAD CCWs are in the beginning of the spool file.

  ANY
  : means that LOAD CCWs are located throughout the spool file.

CHARS
: is the name of the character sets that determine the size and style of the characters on your printed output. You may specify up to four names.

SIZE
: is the number of 4KB DASD blocks allocated for data for the file.

Displaying the Print Server Facility (PSF) Attributes

The PSF option allows you to display the additional information fields of DESTINATION, CONVERSION, SPECIAL, and PURGE. You can display this information by entering one of the following commands:

```
query reader psf
query printer psf
query punch psf
```

In response to these commands, CP displays information similar to the following:

```
ORIGINID FILE CLASS RECORDS  FORM     DEST    CONVERSION SPECIAL PURGE
userid   spid c typ nnnnnnnn formname dest    convstat   YES     YES
                                                         NO      NO
```

Where:

ORIGINID
is the logon identification (user ID) of the file owner or originator. OWNERID is displayed when *userid* or the XFER operand is used, or when a class D user is defaulting to, or specifying, SYSTEM.

FILE
is the spool file's identification number.

CLASS
tells you the class of the spool file and the type of device on which the spool file was created.

RECORDS
is the number of logical records in the spool file.

FORM
is the name of the paper type as understood by the user. (This is displayed for a class G user.)

DEST
is the destination value assigned to the spool file. The destination value can be 1 to 8 characters.

CONVERSION
is the conversion status (*convstat*) of the file, which can be any of the following:

COMPLETE
means that the print server has read the spool file, has marked it as being converted, and has created a corresponding CMS file containing the actual data and print controls used to print the file.

ACTIVE
means that the file is currently being converted to a corresponding CMS file.

NOTCONV
means that the file was converted to a corresponding CMS file.

SPECIAL
indicates whether the file contains CCWs for advanced printer data streams, where:

YES
means that the file does contain CCWs.

NO
means that the file does not contain CCWs.

PURGE
indicates whether the file was purged from CP and is waiting for the print server to recognize this status, where:

YES
means that the file was purged.

NO
means that the file was not purged.

Printing Your Spool Files

You can have locally attached printers process your spool files by using the SPOOL and CHANGE commands. Your installation can tell you how the routing is accomplished—generally by using a combination of CLASS, FORM, and DEST values to differentiate among output class, paper type, and location of printer. You will then have to match these options to those of your installation.

Note: Advanced function printers (AFP*) are controlled by a print server rather than CP. For more information about AFPs, see the *Print Services Facilities/VM System Programmer's Guide*, S544-3467.

In the two examples below, assume that your installation has specified the settings for local printing on your system to be the following:

CLASS	FORM	DEST
U (Unclassified)	MEMO (8-1/2 by 11 inch paper)	FLOOR1 (First-floor printer and distribution)
C (Classified)	LISTING (11 by 14 inch paper)	FLOOR3 (Third-floor printer and distribution)

Example: Using the CHANGE Command to Process Spool Files

To print a classified listing of a held spool file using the third-floor printer, enter:

```
spool vdev hold
print filename filetype
change prt spid class c form listing dest floor3 nohold
```

Where *vdev* is the number of a virtual printer, print *filename filetype* will create a printer file in hold status, and *spid* is the spool file identifier by which you may select a particular file.

Example: Using the SPOOL Command to Process Spool Files

To print an unclassified CMS file on 8-1/2 by 11 inch paper using the first-floor printer, enter:

```
spool vdev class u form memo dest floor1 nohold nocont
print filename filetype
```

Where *vdev* is the virtual device number of the virtual spooling device whose options you want to modify.

Note:

1. The PRINT command is a CMS command.
2. If an external security manager (ESM) is installed, you may not be authorized to issue the CHANGE and SPOOL commands. Further, the ESM may put the spool file you are trying to print in hold status and issue messages to the system operator. This occurs if you are not authorized to print files on the printer you selected. For additional information, contact your security administrator.

Changing the Spooling Options for a Spool File

This section describes some ways you can use the CHANGE command to change spooling options for a specific closed spool file. (For more information on using the CHANGE command, or using the CLOSE command to change a spool file's spooling options, see *z/VM: CP Commands and Utilities Reference*.)

The general format of the CHANGE command is:

```
change devtype spoolid options
```

Where:

devtype
is the type of spooled device (PRINTER, PUNCH, or READER) for which the spool file you are changing is queued.

spoolid
is the identification number of the spool file. (To determine the spool file identification number, use QUERY PRINTER ALL, QUERY READER ALL, or QUERY PUNCH ALL; the number listed under FILE is the spool file's identification number.)

options
are the new spooling options you want to assign to the spool file.

Note: If an ESM is installed, you may not be authorized to enter the CHANGE command. For additional information, contact your security administrator.

Example: Preventing CP from Processing a Spool File (HOLD and NOHOLD)

You can keep CP from processing a specific spool file by placing the spool file in user-hold status. For example, if you want to prevent CP from printing spool file 0001 on a real printer, enter:

```
change printer 0001 hold
```

Where:

printer
tells CP that the spool file you want to change is queued for your printer.

0001 hold
tells CP to place spool file 0001 in user-hold status.

To determine whether a file is in user-hold status, enter the QUERY PRINTER ALL command. CP responds with information similar to the following:

```
ORIGINID  FILE CLASS RECORDS  CPY HOLD DATE  TIME      NAME  TYPE  DIST
YOURID    0001 A PRT 00000150 001 USER 10/22 22:05:01              BIN-9999
YOURID    0002 A PRT 00000150 001 USER 10/23 04:05:25              BIN-9999
```

If a file is in user-hold status, USER is displayed in the HOLD field.

If you want to remove spool file 0001 from user-hold status, enter:

```
change printer 0001 nohold
```

Example: Keeping a Copy of a Spool File After It Is Processed

If you want a copy of spool file 0001 to remain queued for your spooled printer after CP prints it on a real printer, enter:

```
change printer 0001 keep
```

To determine whether a file is in keep status, enter the QUERY PRINTER command. The CP response should be similar to the following:

```
ORIGINID  FILE CLASS RECORDS  CPY  HOLD FORM     DEST    KEEP MSG
YOURID    0001 A PRT 00000150 001  USER LISTING  FLOOR3  ON   ON
YOURID    0002 A PRT 00000150 001  USER LISTING  FLOOR3  OFF  OFF
```

Where:

ORIGINID
is the logon identification (user ID) of the file owner or originator. OWNERID is displayed when *userid* or the XFER operand is used, or when a class D user is defaulting to, or specifying, SYSTEM.

FILE
is the spool file's identification number.

CLASS
tells you the class of the spool file and the type of device on which the spool file was created. The class can be a letter (A–Z) or a number (0–9). The origin may be one of the following:

PRT
is the virtual printer.

RDR
is the real reader.

PUN
is the virtual punch.

CON
is the virtual console.

RECORDS
is the number of logical records in the spool file.

CPY
is one of the following:

nnn
is the number of copies requested for the spool file. This is shown as a 3-digit number.

****nnn***
an asterisk (*) preceding *nnn* indicates that the 3800 printer should print each page from the spooled output files *nnn* times before going on to the next page.

HOLD
is the status of the file, where:

USER
means the file is in user-hold status.

SYS
means the file is in system-hold status.

USYS
means the file is in both user- and system-hold status.

NONE
means the file is not in hold status.

Note: If the file is in system-hold or user-hold status, you cannot print it.

FORM
is the name of the form associated with the file as understood by the user.

DEST
is the destination value assigned to the spool file. The destination value can be 1 to 8 characters.

KEEP
is the keep status of the file, where:

ON
means that the file is in keep status.

OFF
means that the file is not in keep status.

Note: If the file is in keep status, you may print the file, but the system also keeps a copy in user hold.

MSG
is the message status of the file, where:

ON
means that the file is in message status.

OFF
means that the file is not in message status.

Note: If the file is in message status, CP sends a message to the file's owner when you print the file.

If you want to remove spool file 0001 from keep status, enter:

```
change printer 0001 nokeep
```

Example: Changing the Destination Value of a Spool File

If you want to change the destination of spool file 0001 from FLOOR3 to FLOOR2, enter:

```
change printer 0001 dest floor2
```

To determine the destination of a spool file, enter the QUERY PRINTER command. The CP response should be similar to the following:

```
ORIGINID  FILE CLASS RECORDS  CPY  HOLD FORM     DEST    KEEP MSG
YOURID    0001 A PRT 00000150 001  USER LISTING  FLOOR2  ON   ON
YOURID    0002 A PRT 00000150 001  USER LISTING  FLOOR3  OFF  OFF
```

The destination is listed under DEST.

Example: Changing the Number of Copies You Print

If you want to change to five the number of copies CP prints of spool file 0002, enter:

```
change printer 0002 copy 5
```

To determine the number of copies of a spool file that CP processes, enter the QUERY PRINTER ALL command. The CP response should be similar to the following:

```
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME      NAME  TYPE   DIST
YOURID   0001 A PRT 00000150 001 NONE 10/22 22:05:01               BIN-9999
YOURID   0002 A PRT 00000150 005 USER 10/23 04:05:25               BIN-9999
```

The number of copies CP processes is listed under CPY.

Example: Changing a Spool File's Class

If you want to change spool file 0002 to class C, enter:

```
change printer 0002 class c
```

To determine the class of a spool file, enter the QUERY PRINTER ALL command. The CP response should be similar to the following:

```
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME      NAME  TYPE   DIST
YOURID   0001 A PRT 00000150 001 NONE 10/22 22:05:01               BIN-9999
YOURID   0002 C PRT 00000150 001 USER 10/23 04:05:25               BIN-9999
```

The class is listed under CLASS.

Class D users can change the value of the security label associated with any file in the print queue if the following conditions are met:

- An external security manager (ESM) is installed
- Security label checking is enabled, and:
  - The user is exempt from ESM authorization checking, or
  - The system has been placed in a tranquil state by the security administrator.

For additional information, contact your security administrator.

The format of the CHANGE command with a security label is:

```
change devtype spoolid SECLABEL ssssssss
```

Where:

devtype
: is the type of spooled device (PRINTER, PUNCH, or READER) for which the spool file you are changing is queued.

spoolid
: is the identification number of the spool file.

SECLABEL *ssssssss*
: is the security label to be assigned to the file. The *ssssssss* variable is a 1- to 8-character alphanumeric value.

Example: Adding a Security Label to a Printer Spool File

You need to add a security label to spool file 1392. To do this, enter:

```
change prt 1392 seclabel prior999
```

Sending Spool Files to Other Users and Devices

Use the CHANGE command to send spool files to other spooled devices (either yours or another user's). First, you must find out the identification number of the spool file you want to send (use QUERY PRINTER, QUERY PUNCH, or QUERY READER; the number listed under FILE is the spool file's identification number). Once you know the identification number of the spool file, enter:

```
change device1 spoolid to userid device2
```

Where:

device1
: specifies the type of device (PRINTER, PUNCH, READER) from which you are moving the file.

spoolid
: is the identification number of the spool file you are moving.

userid
: is the user ID of the person who owns the virtual device to which you are moving the file. (If you own the device, use your user ID or *.)

device2
: specifies the type of device you are moving the file to (PRINTER, PUNCH, or READER).

Note: If an ESM is installed, you may not be authorized to enter the CHANGE command. For additional information, contact your security administrator.

Example: Moving a File from Your Reader to Your Printer

To move a file with identification number 0111 from your spooled reader to your spooled printer, enter:

```
change reader 0111 to * printer
```

Example: Moving a File from Your Printer to Another User's Printer

To move a file with identification number 0222 from your spooled printer to the spooled printer of userid BANKS, enter:

```
change printer 0222 to banks printer
```

Note: If an ESM is installed, you may not be authorized to enter the TRANSFER command. For additional information, contact your security administrator.

To transfer spool files, enter:

```
transfer devtype spoolid|all to|from *|userid2|altid devtype2
```

Where:

devtype
is the type of spooled device (PRINTER, PUNCH, or READER) containing the spool file you want to transfer.

spoolid
is the identification number of the spool file.

all
specifies the transfer of all files in the queue.

to|from *
indicates files are transferred to or from your own user ID.

to|from *userid2*
identifies the user to or from whom the files are transferred.

to|from *altid*
indicates the files are transferred to or from your alternate ID.

devtype2
is the type of spooled device (PRINTER, PUNCH, or READER) where the spool file is to be transferred.

Example: Transferring a Spool File from Your Reader to a User

You want to transfer a spool file with spoolid 9876 from the reader queue to the reader queue of WARRENT. To transfer the spool file, enter:

```
transfer rdr 9876 to warrent
```

Example: Transferring Spool Files from the Printer to Your Reader

Spool files on your printer queue need to be transferred to your reader queue. To transfer the spool files, enter:

```
transfer prt all to * rdr
```

Changing the Processing Order of Your Spool Files

Use the ORDER command to change the processing order of spool files queued for your spooled printer, punch, or reader. (To find out the current order, use QUERY PRINTER ALL, QUERY PUNCH ALL, or QUERY READER ALL.)

Example: Ordering Files by Spool File Identification Number

Assume that when you enter the QUERY READER ALL command, CP responds with the following:

```
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME     NAME    TYPE  DIST
YOURID   0001 C PUN 00000152 001 NONE 10/22 22:03:01 COLD    FILE  BIN-9999
YOURID   0003 C PUN 00000064 001 USER 10/23 04:09:25 COOL    FILE  BIN-9999
YOURID   0010 C PUN 00000176 001 SYS  10/23 06:05:09 BOILING FILE  BIN-9999
YOURID   0015 C PUN 00000088 001 USYS 10/23 05:08:17 HOT     FILE  BIN-9999
```

Now assume that you want to process the files named BOILING and HOT first. To do this, enter:

```
order reader 0010 0015
```

where *0010 0015* are the identification numbers of the files in the order you want to process them.

When you reenter the QUERY READER ALL command, CP lists your reader spool files in their updated order as follows:

```
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME     NAME    TYPE  DIST
YOURID   0010 C PUN 00000176 001 SYS  10/23 06:05:01 BOILING FILE  BIN-9999
YOURID   0015 C PUN 00000088 001 USYS 10/23 05:08:25 HOT     FILE  BIN-9999
YOURID   0001 C PUN 00000152 001 NONE 10/22 22:03:09 COLD    FILE  BIN-9999
YOURID   0003 C PUN 00000064 001 USER 10/23 04:09:17 COOL    FILE  BIN-9999
```

Example: Ordering Files by Class

Assume that when you enter the QUERY PUNCH ALL command, CP responds with the following:

```
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME     NAME    TYPE  DIST
YOURID   0210 X PUN 00000011 001 NONE 11/08 20:05:01               BIN-9999
YOURID   0315 A PUN 00000022 001 USER 03/09 04:05:25               BIN-9999
YOURID   0501 Z PUN 00000133 001 SYS  07/31 05:01:09               BIN-9999
YOURID   0003 A PUN 00000034 001 USYS 09/18 04:31:17               BIN-9999
```

Now assume that you want CP to punch all class A files before any other files. To do this, enter:

```
order punch class a
```

where *class a* is the file class you want to process first.

When you reenter the QUERY PUNCH ALL command, CP lists your spool files in their updated order as follows:

```
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME     NAME    TYPE  DIST
YOURID   0315 A PUN 00000022 001 USER 03/09 04:05:01               BIN-9999
YOURID   0003 A PUN 00000034 001 USYS 09/18 04:31:25               BIN-9999
YOURID   0210 X PUN 00000011 001 NONE 11/08 20:05:09               BIN-9999
YOURID   0501 Z PUN 00000133 001 SYS  07/31 05:01:17               BIN-9999
```

Example: Ordering Files by Form Name

Assume that when you enter the QUERY PRINTER command, CP responds with the following:

```
ORIGINID FILE CLASS RECORDS  CPY  HOLD FORM     DEST    KEEP MSG
YOURID   0012 A PRT 00000150 001  NONE BIG      FLOOR3  ON   ON
YOURID   0023 A PRT 00000150 001  USER STANDARD FLOOR3  OFF  OFF
YOURID   0001 A PRT 00000150 001  SYS  BIG      FLOOR3  ON   ON
YOURID   0045 A PRT 00000150 001  USYS STANDARD FLOOR3  OFF  OFF
YOURID   0034 A PRT 00000150 001  NONE BIG      FLOOR3  ON   ON
YOURID   0056 A PRT 00000150 001  USER STANDARD FLOOR3  OFF  OFF
```

Now assume that you want CP to print file 0001 first, and you want CP to print files with the form name BIG before any of the remaining files. To do this, enter:

```
order printer 0001 form big
```

Where:

0001
: is the spool file identification number of the spool file you want to process first.

form big
: is the form name associated with the spool files you want to process after spool file 0001.

When you reenter the QUERY PRINTER command, CP lists your spool files in their updated order as follows:

```
ORIGINID FILE CLASS RECORDS  CPY  HOLD FORM     DEST    KEEP MSG
YOURID   0001 A PRT 00000150 001  SYS  BIG      FLOOR3  ON   ON
YOURID   0012 A PRT 00000150 001  NONE BIG      FLOOR3  ON   ON
YOURID   0034 A PRT 00000150 001  NONE BIG      FLOOR3  ON   ON
YOURID   0023 A PRT 00000150 001  USER STANDARD FLOOR3  OFF  OFF
YOURID   0045 A PRT 00000150 001  USYS STANDARD FLOOR3  OFF  OFF
YOURID   0056 A PRT 00000150 001  USER STANDARD FLOOR3  OFF  OFF
```

Purging Spool Files

To remove closed spool files from your spooled printer, punch, or reader, use the PURGE command.

Assume, for the next three examples, that when you enter the QUERY PRINTER ALL command, CP responds with the following:

```
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME     NAME    TYPE  DIST
YOURID   0007 X PRT 00000010 001 NONE 09/09 20:05:01               BIN-9999
YOURID   0009 A PRT 00000010 001 USER 09/10 04:05:25               BIN-9999
YOURID   0010 T PRT 00000010 001 SYS  09/10 05:01:09               BIN-9999
YOURID   0222 T PRT 00000010 001 USYS 09/11 04:31:17               BIN-9999
YOURID   0333 T PRT 00000010 001 NONE 09/11 04:32:44               BIN-9999
```

Example: Purging Specific Spool Files

To purge from your printer the spool files with identification numbers 0007 and 0010, enter:

```
purge printer 7 10
```

When you reenter the QUERY PRINTER ALL command, CP responds as follows:

```
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME     NAME    TYPE  DIST
YOURID   0009 A PRT 00000010 001 USER 09/10 04:05:01               BIN-9999
YOURID   0222 T PRT 00000010 001 USYS 09/11 04:31:25               BIN-9999
YOURID   0333 T PRT 00000010 001 NONE 09/11 04:32:09               BIN-9999
```

Example: Purging All Spool Files with the Same Class

To purge from your printer all class T spool files, enter:

```
purge printer class t
```

When you reenter the QUERY PRINTER ALL command, CP responds as follows:

```
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME     NAME    TYPE  DIST
YOURID   0009 A PRT 00000010 001 USER 09/10 04:05:01               BIN-9999
```

Example: Purging All Spool Files

To purge all spool files from your printer, enter:

```
purge printer all
```

When you reenter the QUERY PRINTER command, CP responds as follows:

```
NO PRT FILES
```

Example: Purging All Spool Files with the Same Destination Value

Assume, for this example, that when you enter the QUERY PRINTER command, CP responds with the following:

```
ORIGINID FILE CLASS RECORDS  CPY  HOLD FORM     DEST    KEEP MSG
YOURID   0001 A PRT 00000150 001  SYS  BIG      FLOOR2  ON   ON
YOURID   0012 A PRT 00000150 001  NONE BIG      FLOOR3  ON   ON
YOURID   0034 A PRT 00000150 001  NONE BIG      FLOOR2  ON   ON
YOURID   0023 A PRT 00000150 001  USER STANDARD FLOOR3  OFF  OFF
YOURID   0045 A PRT 00000150 001  USYS STANDARD FLOOR3  OFF  OFF
YOURID   0056 A PRT 00000150 001  USER STANDARD FLOOR2  OFF  OFF
```

Now assume that you want to purge all files that have the destination value of FLOOR3. To do this, enter:

```
purge printer dest floor3
```

When you reenter the QUERY PRINTER command, CP responds as follows:

```
ORIGINID FILE CLASS RECORDS  CPY  HOLD FORM     DEST    KEEP MSG
YOURID   0001 A PRT 00000150 001  SYS  BIG      FLOOR2  ON   ON
YOURID   0034 A PRT 00000150 001  NONE BIG      FLOOR2  ON   ON
YOURID   0056 A PRT 00000150 001  USER STANDARD FLOOR2  OFF  OFF
```

Saving Your Spool Files or System Trace Files on Tape

You can use the SPXTAPE DUMP command to save copies of your spool files or system trace files on tape. When saving spool files, SPXTAPE by default saves the files regardless of their hold status. Open or in-use files cannot be saved.

To save all your closed spool files on tape:

1. Follow the procedure used by your installation to obtain one or more tape drives attached to your virtual machine.

   Note: Depending on the number of files you want to dump, you should consider using more than one tape drive. Using multiple tape drives can significantly reduce the processing time because SPXTAPE can dump files to multiple tape drives simultaneously.
2. Spool your console to wherever you want the SPXTAPE log files to go. For example, spool the console to yourself so the logs are sent to your reader. Enter:

   ```
   spool cons *
   ```

   Note: If you spool your console *after* entering the SPXTAPE command, the logs may be sent to different destinations.
3. If necessary, use the CMS TAPE command to position the tape to where CP can begin dumping files. The tape must be positioned either at the beginning of the volume or following a tape mark.
4. Enter the SPXTAPE DUMP command:

   ```
   spxtape dump vdev std all run
   ```

   where *vdev* is the virtual device number of the tape drive you are using. If you are using more than one tape drive, enter the virtual device numbers as a range, *vdev1-vdev2*. The RUN operand rewinds and unloads the tape when processing to that volume completes.

   In response, CP dumps a copy of each closed reader, printer, and punch spool file to tape without removing the files from their queues on the system. As it processes the files, CP reports its progress at the interval requested with the PROGress_interval option (or about every 15 seconds by default) by displaying a message on the screen.

   Note: If you do not want to retain the spool files on the system after you have saved them on tape, you can have CP purge the files. In that case, enter the SPXTAPE command with the PURGE operand, as follows:

   ```
   spxtape dump vdev std all run purge
   ```

   CP removes each dumped file from the queue and tells you it has purged the file.

   CP records information about the dump in two or more logs. The command summary log contains information about the progress and status of the logical SPXTAPE command. A logical SPXTAPE command can be either one independent command or a sequence of appended commands (entered with the APPEND operand). The volume log contains information about the files processed by the logical SPXTAPE command that are associated with a particular tape volume. Therefore, several volume logs are created for one logical SPXTAPE command if more than one tape drive is used or if more than one tape volume is mounted on any drive.

   If any tape is filled before all the files are dumped, CP sends you a response that:

   a. Asks for the next tape to be mounted on that device.
   b. Indicates the number of files and spool pages processed so far.

Follow the procedure used by your installation to notify the operator to mount another tape. If you are using more than one tape drive, CP continues dumping to the other drives. When the next tape is mounted and readied, CP begins dumping to that drive again. When CP has dumped all the files, it tells you the dump function is complete.

5. If you want to find out what files have been dumped, you can look at the volume logs. Each volume log has a unique name (file name and file type) related to both the tape volume and the SPXTAPE command that caused the creation of the log. For example, the volume log for the first volume processed on virtual device 181 for an SPXTAPE DUMP command has the name:

   ```
   xxyyD181 hhmmss01
   ```

   where *xxyy* is the month and day the command was issued, D indicates the operation was DUMP, 181 is the virtual device number of the tape drive, *hhmmss* is the time stamp, and 01 is the volume sequence number.

   The volume log contains the following entry for each file written to the volume: Some of the entries have been truncated to fit the page.

   ```
   USERID FILE QUEUE FLNAM FLTYP OPNDAT OPNTIM ORIGID SZ SEG_ST HLDSTAT RECCNT C FORM DEST LSTPG
   userid file queue flenam fltyp opndat opntim origid sz COMPLETE hldstat reccnt c form dest
   lstpag
   ```

   Note: The heading is included only once for each uninterrupted series of entries.

   If the SEG_STAT field contains PARTIAL instead of COMPLETE, the file is split between this volume and one or more other volumes, and this is *not* the part that completes the file (the last part dumped). If all parts of the file have dumped correctly, the volume log entry for the last part dumped contains COMPLETE in the SEG_STAT field.

The preceding steps tell you how to dump all your standard spool files to tape. You can also select the spool files to be dumped by queue (reader, printer, or punch), spool file identification number or range, class, destination value, form name, file name pattern, file type pattern, hold status, SSI cluster copy status, or combinations of these attributes. You can select system trace files to be dumped by spool file identification number or range, class, file name pattern, file type pattern, or combinations of these attributes. A file name or file type pattern can be either a complete file name or file type or a string containing wild cards (* and %). The following example shows you how to save a particular class of files from the reader and printer queues.

For more information about the SPXTAPE logs and the options you can specify on the SPXTAPE DUMP command, see *z/VM: CP Commands and Utilities Reference*.

Example: Saving a Particular Class of Reader and Printer Spool Files on Tape

Suppose you have a group of class S reader and printer files you want to save on tape. To do this:

1. Obtain one or more tape drives and have tapes mounted on them.
2. Spool your console. To spool the console to yourself, enter:

   ```
   spool cons *
   ```

3. If necessary, use the CMS TAPE command to position the tape to the beginning of the volume or following a tape mark.
4. Start the dump. Enter:

   ```
   spxtape dump 181 reader class s append
   spxtape dump 181 printer class s run
   ```

   Specifying the APPEND operand on the first command combines these two commands into a single logical command. CP begins processing the files as soon as you enter the first command in the sequence. CP displays progress responses, requests to mount new tapes (if necessary), and a final response when the dump function is complete. Each file dumped is recorded in the volume log for that tape volume.

Example: Saving a Group of System Trace Files on Tape

Suppose you have a group of system trace files in the spool ID range 5000 to 5999 that you want to save on tape. To do this:

1. Obtain one or more tape drives and have tapes mounted on them.
2. Spool your console. To spool the console to yourself, enter:

   ```
   spool cons *
   ```

3. If necessary, use the CMS TAPE command to position the tape to the beginning of the volume or following a tape mark.
4. Start the dump. Enter:

   ```
   spxtape dump 181 trfiles spid 5000 5999 run
   ```

   CP displays progress responses, requests to mount new tapes (if necessary), and a final response when the dump function is complete. Each file dumped is recorded in the volume log for that tape volume.

Restoring Your Spool Files or System Trace Files from Tape

Use the SPXTAPE LOAD command to restore copies of your spool files or system trace files that you previously saved on tape by SPXTAPE DUMP.

To restore all the spool files from tape:

1. Follow the procedure used by your installation to have tapes containing your spool files mounted on one or more tape drives attached to your virtual machine.

   Note: Depending on the number of tapes you want to load, you should consider using more than one tape drive. Using multiple tape drives can significantly reduce the processing time because SPXTAPE can load files from multiple tape drives simultaneously.

   The order in which the tapes are mounted does not matter, as long as the files were dumped with one logical SPXTAPE command and you use one logical SPXTAPE command to load them. A logical SPXTAPE command can be either one independent command or a sequence of appended commands (entered with the APPEND operand). If a file is split across two or more tape volumes, CP fits the parts together correctly.
2. Spool your console to wherever you want the SPXTAPE log files to go. For example, spool the console to yourself so the logs are sent to your reader. Enter:

   ```
   spool cons *
   ```

   Note: If you spool your console *after* entering the SPXTAPE command, the logs may be sent to different destinations.
3. If necessary, use the CMS TAPE command to position the tape to where CP can begin reading data. The tape must be positioned either at the beginning of the volume or at the beginning of a tape file (following the tape mark at the end of the previous tape file).
4. If you want to find out what spool files are on tape before you do the load, and the volume logs for the dump are not available, you can use the SPXTAPE SCAN command.

   a. To scan the tape, enter:

      ```
      spxtape scan vdev std all run
      ```

      where *vdev* is the virtual device number of the tape drive you are using. If you are using more than one tape drive, enter the virtual device numbers as a range, *vdev1-vdev2*. The RUN operand rewinds and unloads the tape when processing to that volume completes.

Note: If you have only one tape to scan on each tape drive, you may prefer to use the REWIND operand, which rewinds the tape but does not unload it from the drive. If the tape contains tape files dumped by multiple logical SPXTAPE DUMP commands, you must use multiple logical SPXTAPE SCAN commands to scan them. In that case, you may want to use the LEAVE operand, which leaves the tape positioned to process the next tape file.

In response, CP scans each reader, printer, and punch file in the selected tape file. As it processes the files, CP reports its progress at the interval requested with the PROGress_interval option (or about every 15 seconds by default) by displaying a message on the screen.

CP records information about the scan in two or more logs. The command summary log contains information about the progress and status of the logical SPXTAPE command. The volume log contains information about the files processed by the logical SPXTAPE command that are associated with a particular tape volume. Therefore, several volume logs are created for one logical SPXTAPE command if more than one tape drive is used or if more than one tape volume is mounted on any drive.

b. If you entered the SPXTAPE SCAN command with the RUN disposition, CP sends you a response when processing of a tape volume is complete that:

   i) Asks for the next tape to be mounted on that device.

   ii) Indicates the number of files and spool pages processed so far.

   If there are other tapes to be processed, follow the procedure used by your installation to notify the operator to mount another tape. If you are using more than one tape drive, CP continues scanning on the other drives. When the next tape is mounted and readied, CP begins scanning on that drive again.

   If there are no other tapes to be mounted, and all tapes currently mounted have finished processing (you have received the "mount next tape" response for each drive), use the SPXTAPE END command to end the operation. Enter:

```
spxtape end vdev
```

   where *vdev* is the same virtual device number you specified on the SPXTAPE SCAN command. If you specified a range (*vdev1-vdev2*) on that command, you must enter the same range on this command. The tape is positioned according to the disposition specified on the SPXTAPE SCAN command.

c. To see the results of the scan, look at the volume logs. Each volume log has a unique name (file name and file type) related to both the tape volume and the SPXTAPE command that caused the creation of the log. For example, the volume log for the second volume processed on virtual device 181 for an SPXTAPE SCAN command has the name:

```
xxyyS181 hhmmss02
```

   where *xxyy* is the month and day the command was issued, S indicates the operation was SCAN, 181 is the virtual device number of the tape drive, *hhmmss* is the time stamp, and 02 is the volume sequence number.

   The volume log contains the following entry for each file scanned on the volume: Some of the entries have been truncated to fit the page.

```
USERID FILE QUEUE FLNAM FLTYP OPNDAT OPNTIM ORIGID SZ SEG_ST HLDSTAT RECCNT C FORM DEST
LSTPG
userid file queue flnam fltyp opndat opntim origid sz COMPLETE hldstat reccnt c form dest
lstpag
```

   Note: The heading is included only once for each uninterrupted series of entries.

   If the SEG_STAT field contains PARTIAL instead of COMPLETE, it means one of the following:

   - The file is split between this volume and one or more other volumes, and this part is *not* the one that completes the file (the last part scanned). If all parts of the file have scanned correctly, the volume log entry for the last part scanned contains COMPLETE in the SEG_STAT field.

Note: The last part scanned may contain the end of the file, the beginning, or some section in between, depending on the order in which the tape volumes were mounted.

- CP encountered an I/O error or some other problem trying to read the file from the tape. In that case, you also receive error messages describing the problem. The error messages are recorded in the command summary log.

5. If necessary, use the CMS TAPE command to position the tape to the beginning of the tape file you want to load.
6. Load all the spool files. Enter:

   ```
   spxtape load vdev std all run
   ```

   where *vdev* is the virtual device number of the tape drive you are using. If you are using more than one tape drive, enter the virtual device numbers as a range, *vdev1-vdev2*. The RUN operand rewinds and unloads the tape when processing of that volume completes.

   Note: If the tape contains tape files dumped by multiple logical SPXTAPE DUMP commands, you must use multiple logical SPXTAPE LOAD commands to load them. In that case, you may prefer to use the LEAVE operand, which leaves the tape positioned to process the next tape file.

   In response, CP reads each reader, printer, and punch file from the selected tape file and adds it to the appropriate spooling system queue. As it processes the files, CP reports its progress at the interval requested with the PROGress_interval option (or about every 15 seconds by default) by displaying a message on the screen. As it loads each file, CP assigns a new spool file identification number to it and sends you a response (if you have IMSG set ON).

   CP records information about the load in two or more logs. The command summary log contains information about the progress and status of the logical SPXTAPE command. The volume log contains information about the files processed by the logical SPXTAPE command that are associated with a particular tape volume. Therefore, several volume logs are created for one logical SPXTAPE command if more than one tape drive is used or if more than one tape volume is mounted on any drive.
7. If you entered the SPXTAPE LOAD command with the RUN disposition, CP sends you a response when processing of a tape volume is complete that:

   a. Asks for the next tape to be mounted on that device.
   b. Indicates the number of files and spool pages processed so far.

   If there are other tapes to be processed, follow the procedure used by your installation to notify the operator to mount another tape. If you are using more than one tape drive, CP continues loading from the other drives. When the next tape is mounted and readied, CP begins loading from that drive again.

   If there are no other tapes to be mounted, and all tapes currently mounted have finished processing (you have received the "mount next tape" response for each drive), use the SPXTAPE END command to end the operation. Enter:

   ```
   spxtape end vdev
   ```

   where *vdev* is the same virtual device number you specified on the SPXTAPE LOAD command. If you specified a range (*vdev1-vdev2*) on that command, you must enter the same range on this command. The tape is positioned according to the disposition specified on the SPXTAPE LOAD command.
8. If you want to find out what files have been loaded, you can look at the volume logs. Each volume log has a unique name (file name and file type) related to both the tape volume and the SPXTAPE command that caused the creation of the log. For example, the volume log for the second volume processed on virtual device 182 for an SPXTAPE LOAD command has the name:

   ```
   xxyyL182 hhmmss02
   ```

   where *xxyy* is the month and day the command was issued, L indicates the operation was LOAD, 182 is the virtual device number of the tape drive, *hhmmss* is the time stamp, and 02 is the volume sequence number.

The volume log contains the following entry for each file loaded from the volume: Some of the entries have been truncated to fit the page.

```
USERID FILE QUEUE FLNAM FLTYP OPNDAT OPNTIM ORIGID SZ SEG_ST HLDSTAT RECCNT C FORM DEST LSTPG
userid file queue flnam fltyp opndat opnti origid sz COMPLETE hldstat reccnt c form dest
lstpag
```

Note: The heading is included only once for each uninterrupted series of entries.

If the SEG_STAT field contains PARTIAL instead of COMPLETE, it means one of the following:

- The file is split between this volume and one or more other volumes, and this part is *not* the one that completes the file (the last part loaded). If all parts of the file have loaded correctly, the volume log entry for the last part loaded contains COMPLETE in the SEG_STAT field.

  Note: The last part loaded may contain the end of the file, the beginning, or some section in between, depending on the order in which the tape volumes were mounted.
- CP encountered an I/O error or some other problem trying to read the file from the tape or write it to DASD. In that case, you also receive error messages describing the problem. The error messages are recorded in the command summary log.

The preceding steps tell you how to restore all the standard spool files dumped to tape. You can also select the spool files to be loaded by queue (reader, printer, or punch), spool file identification number or range, class, destination value, form name, file name pattern, file type pattern, hold status, or combinations of these attributes. You can select system trace files to be dumped by spool file identification number, class, file name pattern, file type pattern, or combinations of these attributes. A file name or file type pattern can be either a complete file name or file type or a string containing wild cards (* and %). The following example shows you how to restore a particular class of files to the reader and printer queues.

For more information about the SPXTAPE logs and the other options you can specify on the SPXTAPE LOAD command, see *z/VM: CP Commands and Utilities Reference*.

Example: Restoring a Particular Class of Reader and Printer Spool Files from Tape

Suppose you previously saved all your spool files on tape. Now you decide you want to restore only the class T reader and printer files. To do this:

1. Obtain one or more tape drives and have the tapes mounted on them.
2. Spool your console. To spool the console to yourself, enter:

   ```
   spool cons *
   ```
3. If necessary, use the CMS TAPE command to position the tape to the beginning of the tape file that contains the files you want to load.
4. Start the load. Enter:

   ```
   spxtape load 181 reader class t append
   spxtape load 181 printer class t run
   ```

   Specifying the APPEND operand on the first command combines these two commands into a single logical command. CP begins processing the files after you enter the last command in the sequence. CP displays progress responses and requests to mount new tapes. Each file loaded is recorded in the volume log for that tape volume.

   Note: The previous procedure assumes the files were saved on tape using one logical SPXTAPE DUMP command. If you used independent SPXTAPE DUMP commands to save the files (for example, if you saved them at different times), you must use independent (not appended) SPXTAPE LOAD commands to restore them. Wait for the first command to finish processing before you enter the second command. The disposition operand (RUN, REWIND, or LEAVE) you use on each command depends on whether the files are contained on one tape volume or multiple tape volumes.

5. When there are no more tapes to be processed (all tapes currently mounted have finished processing and there are no more tapes to be mounted), end the SPXTAPE LOAD operation. Enter:

```
spxtape end 181
```

Spool File Attributes

When a spool file is created, it is assigned certain attributes, some of which can be specified by the user. The spool file attributes are:

- **File name and file type or data set name.** For a spool input file, this attribute is specified by the user in the identification card that must precede the card input deck that is read and placed in a spool input file. For a spool output file, this attribute can be specified in the CLOSE command that terminates creation of a spool output file. If these attributes are not specified for a spool output file, the defaults used are CMS file name and file type, if they exist, or blanks.

  File name and file type are attributes that identify a CMS disk file and can be specified for spool files that are processed by CMS. A data set name of up to 16 characters can be assigned to a spool file that is processed by MVS or VSE. (Up to 24 characters may be specified, but the name is truncated to 16 characters.) File name and file type can each be from one to eight alphanumeric characters. These attributes do not uniquely identify a spool file. Hence, duplicate names are permitted.
- **Owner ID.** This is the user ID of the virtual machine to which the spool file belongs. For a spool input file, owner ID is taken from the user-supplied ID card that must precede each card input deck or from the virtual machine to which the spool file was transferred (by way of a SPOOL or TRANSFER command). For a spool output file, the user ID of the virtual machine that created the file is assigned.
- **Originating user.** This is the user ID of the virtual machine that created the spool file if the file has been transferred to the spooled virtual card reader, printer, or punch of the current owner from a spooled virtual reader, punch, printer, or console of a different virtual machine.
- **Device type.** This attribute specifies the type of spool unit record device that is associated with the file (RDR, PUN, PRT, or CON).
- **Spool ID.** This is a number between 1 and 9999 that is automatically assigned to a spool file by CP when a new spool file is closed. Spool IDs are assigned consecutively until the spool file maximum established by the SPOOLFILE statement in the user's z/VM directory entry or 9999 (the default value) is reached. Assignment then begins again at 1. The spool ID of each spool file is unique to each user and is used as a part of the identifier of a spool file. In order to identify a specific spool file in a spool command, the spool device type, user ID, and spool ID usually must be given.
- **Creation date and time at which the spool file was opened.**
- **Spool class.**
- **Logical record size.**
- **Logical record count.** This is the number of cards or lines in the spool file, which can be a value up to 999 million.
- **Number of copies.** For a spool output file, this is the number of copies requested by the user, from 1 to 255, or a default of 1. If the number is preceded by an asterisk and the file is printed on a real 3800 printer, the 3800 printer prints each page of the spool output file the specified number of times before going on to the next page. If the file is not printed on a real 3800, the asterisk is ignored and duplication is performed normally.
- **Distribution code.** This is for output separator identification.
- **Hold status.** This attribute specifies whether a file has been placed in hold status by the user, by the system, by neither, or by both. An output spool file that is in either user hold or system hold cannot be printed. An input spool file that is in user hold status will not be purged after it is read by a virtual card reader. A file that is in user hold can be released by either the general user or by the spooling operator using the CHANGE command. A file that is in system hold can be released only by the spooling operator.

- **Keep status.** This attribute specifies whether the user wishes to keep a file in user hold status after it has been processed. After an output file in keep status has been printed or punched by a real printer or punch, and after an input file has been read by a virtual card reader, the file is placed in user hold status.
- **Message status.** This attribute specifies whether the user wishes to be informed when an output spool file prints or punches.
- **Unconverted status.** When a spool file is selected, it is converted to a data stream. The converted information is stored in CMS files on minidisks shared by the print-server virtual machines. A user or an operator who decides the conversion process should be redone can use the CHANGE command with the UNCONV option. CP gives the file's data to a new file marked *not converted*, which is now eligible to be selected for conversion again.
- **User form name.** This is the 1- to 8-character name of the form on which a spool output file is printed or punched, as known to the user. Each user form name corresponds to an operator form name as specified in the USERFORM statement during system generation. In order to print on a real printer, the spool class and operator form name of a spool file must match the spool class and operator form name specified by the operator for the real printer.
- **Operator form name.** This is the 1- to 8-character name of the form on which a spool output file is printed or punched, as known to the operator. It corresponds to a user form name as specified in the USERFORM statement during system generation.
- **Destination.** This is the 1- to 8-character name of a real or virtual output spooling device. Each spool file can have one destination value to specify the device(s) that processes it. A virtual printer, punch, or console device can have a destination value that is copied to spool files created on the device. Each real punch and real printer (owned by either CP or a print server) starts with up to four destination values for selecting files to process. This value does not apply to either real or virtual input devices (readers).
- **Character set name(s).** This attribute specifies the name or names of the character sets that are used when printing the spool file on a real 3800 printer. One to four character set names may be specified; each name may consist of 1 to 4 characters. The character set names must be contained in the image library used by the real 3800 printer.

  The character set attribute is ignored by CP for impact printers. The operator must make sure the correct universal character set (UCS) buffers are loaded in advance of starting the 4245 and 4248 printers. For all other impact printers, the operator must specify the correct UCS buffers on the START command.
- **Forms control buffer (FCB) name.** This is the named forms control buffer, or vertical spacing (in lines per inch) used when printing the spool file and its separator pages on a real 3800 printer. Vertical spacing values of 6, 8, 10, and 12 lines per inch can be specified. However, an FCB of 10 is only valid when the file is printed on a 3800 Model 3 printer. The named forms control buffer must be contained in the image library used by the real 3800 printer.

  The forms control buffer attribute is ignored by CP for impact printers. The operator must make sure the correct forms control buffers are loaded and must specify the correct forms control buffer on the START command.
- **Forms overlay name.** This is the name of the 3800 forms overlay whose image is superimposed onto the specified number of copies of the output when printing the spool file on a real 3800 printer. A forms overlay name may be from 1 to 4 characters in length; the number of copies specified may range from 1 to 255. The default is to superimpose the forms overlay image onto all copies of the printed output. The forms overlay must be mounted in the 3800 printer by the operator.
- **Copy modification name.** This is the 1- to 4-character name of the 3800 copy modification module used when printing the spool file on a real 3800 printer. The copy modification feature allows a user to specify that the 3800 should add or delete text from selected copies of an output file. A number from 0 to 3 may be specified after the name. This number selects one of the character sets specified to be used for the copy specification text. Zero selects the first character set, 1 selects the second, and so on. If no number is specified, the first character set is used. The copy modification module must be contained in the image library used by the real 3800 printer.

The QUERY READER|PRINTER|PUNCH command displays on the virtual operator's console one line of attribute information about each spool file that currently exists for the spooled virtual reader, printer, or punch of the virtual machine. Each line contains the user ID of the user who created the file, spool file ID, spool class and originating device type, number of logical records in the spool file, number of copies, the hold status, the user form name, and destination. The ALL, EXP (or TBL), and PSF operands can be used with the QUERY READER|PRINTER|PUNCH command to obtain information about the other attributes of a spool file.

Managing Spooled Devices

The rest of this chapter explains how to manage your spooled devices. Specifically, the following sections tell you how to use CP commands if you need to:

- Display the current setting of a spooled device

  Note: The spooling options that apply to spool files also apply to spooled devices
- Change the spooling options for a spooled device
- Set a spooled device so that files you send to it are automatically sent to other users or other devices
- Add and detach spooled devices
- Change the forms control buffer for a spooled 3203 printer
- Prevent I/O requests from being sent to a specific spooled device.

Displaying the Spooling Options of Your Reader

"Displaying Information about Your Spool Files" on page 78 explains how to display information about spool files queued for your spooled reader. If, however, you want to display the current spooling options for your spooled reader (instead of spool files queued for your reader), enter:

```
query virtual reader
```

or

```
query virtual vdev
```

Where *vdev* is the virtual device number of your reader.

In response to this command, CP displays information similar to the following example.

Note: The actual output you receive varies according to how your spooled reader is defined.

```
RDR vdev CL a NOCONT NOHOLD   EOF       READY
    vdev type          CLOSED    NOKEEP NORESCAN
```

Where:

vdev
is the virtual device number of your spooled reader.

CL *a*
specifies the class of spool file this reader processes.

NOCONT
means that CP signals your machine when it reaches the end of a spool file. (This lets you process one file at a time.)

NOHOLD
means that after you process a reader file, CP purges the file unless the file has the KEEP or HOLD option specified for it, or the reader has the KEEP option in effect.

EOF
means that CP reflects a unit exception to your virtual machine when it tries to read the first record after the end of a file.

READY
: means that your spooled reader is operating (that is, information can be read from your spooled reader).

type
: is the device type of your spooled reader.

CLOSED
: means that there is no open spool file active on your reader.

NOKEEP
: means that CP will not automatically keep the file in user-hold status after processing. If the file has the KEEP option, then the file is kept in user hold. If either the reader or the file has the HOLD option, the file is kept without a change in its user-hold status, even if the NOKEEP option is specified.

NORESCAN
: means that scanning will resume at the next logical file in the virtual reader queue.

Note: For more information on spooling options, see *z/VM: CP Commands and Utilities Reference*.

Changing the Spooling Options of Your Reader

Use the SPOOL command to change a spooled reader's spool option settings. The general format of the SPOOL command is:

```
spool vdev options
```

Where:

vdev
: is the virtual device number of the reader.

options
: are the new options you want to assign to the reader.

Or, if you have one or more spooled readers and you want to change the spool option settings on all of them, enter:

```
spool reader options
```

Where *options* are the new options you want to assign to the reader.

Example: Changing the Class of Spool Files You Can Read

If you want to set your spooled reader so that you can read only a particular class of spool file (for example, class A), enter:

```
spool reader class a
```

To reset your reader so that you can read any class of spool file, enter:

```
spool reader class *
```

Example: Keeping Spool Files After They Have Been Processed

If, after you read spool files, you want those spool files to remain in your spooled reader and remain eligible for further processing, enter:

```
spool reader keep
```

To reset this option, enter:

```
spool reader nokeep
```

Example: Holding Spool Files After They Have Been Processed

If, after you read spool files, you want those spool files to remain in your spooled reader in HOLD status, enter:

```
spool reader hold
```

To reset this option, enter:

```
spool reader nohold
```

To release particular files from HOLD status, you must change the HOLD status of the file, not the device. To change a spool file's HOLD status, see "Example: Preventing CP from Processing a Spool File (HOLD and NOHOLD)" on page 83.

Displaying the Spooling Options of Your Printer or Punch

"Displaying Information about Your Spool Files" on page 78 tells you how to display information about spool files queued for your spooled printer or punch. If, however, you want to display the current spooling options for your spooled printer or punch (instead of spool files queued for your printer or punch), enter:

```
query virtual printer
query virtual punch
```

or

```
query virtual vdev
```

Where *vdev* is the virtual device number of your printer or punch.

In response to these commands, CP displays information similar to the following example.

Note: The actual output you receive varies depending on how your spooled printer or punch is defined.

```
PRT  vdev CL a NOCONT NOHOLD CPY   *nnn    READY FORM formname
     vdev TO userid   dev DIST distcode    FLASHC ccc DEST dest
     vdev FLASH ovly  CHAR char MDFY cmod n FCB fcb  LPP nnn
     vdev SIZE width length 4WCGM CFS NODATCK
     vdev type   EOF CLOSED    NOKEEP NOMSG NONAME
     vdev SUBCHANNEL=vsub
```

Where:

vdev
: is the virtual device number.

CL *a*
: specifies the class CP assigns to your spool files as they are created.

NOCONT
: means that the CLOSE command can be used without the EOF option to close open spool files.

NOHOLD
: means that CP can process spool files once they are closed.

COPY **nnn*
: specifies the number (*nnn*) of copies you want CP to print or punch. When the optional asterisk (**nnn*) appears, it specifies the number of copies of each page that a 3800 printer prints before printing the next page.

READY
: means that your spooled printer or punch is operating (that is, information can be sent to the device).

FORM *formname*
: displays the form name that is associated with the device and assigned to the spool files as they are created.

TO *userid dev*
: specifies that when you print or punch spool files, CP sends them to the spooled device of *userid*. (Unless you specify otherwise, *userid* defaults to your own user identification.) The *dev* variable tells you the type of spooled device to which spooled output goes (PRT for spooled printer, PUN for spooled punch, and RDR for spooled reader).

DIST *distcode*
: is an installation-dependent code (1 to 8 alphanumeric characters) that your installation can use as it wants (for example, to specify where output is sent after CP processes it on a real printer or punch).

FLASHC *ccc*
: is the number of copies printed while the forms overlay frame is in place.

DEST *dest*
: is the destination value to be assigned to each spool file created on the device. If the destination value was not changed by a SPOOL command, the default is OFF.

FLASH *ovly*
: is the name of the forms overlay frame superimposed on the output.

CHAR *char*
: is the name of the character set that you use to determine the size and style of the characters on your printed output. Up to four names may be specified.

MDFY *cmod n*
: is the name of the copy modification module and the number (0, 1, 2, or 3) of the copy modification character arrangement table you are using to alter output text.

LPP *nnn*
: indicates the lines per page setting for the virtual printer. The value can be a decimal number in the range 30 to 255, or 'OFF', which indicates that internal defaults will be used.

FCB *fcb*
: is the name of the forms control buffer you use to control the vertical formatting of pages CP prints on 3800 printers.

SIZE *width length*
: is the width and length of the paper to be used (*displayed for virtual 3800 printers only*).

4WCGM
: is the number of writable-character generation modules available to your character set (*displayed for virtual 3800 printers only*). It can also be 2WCGM.

CFS
: specifies a continuous forms stacker (*displayed for virtual 3800 printers only*). The other option is BTS (burster-trimmer-stacker).

NODATCK
: specifies that data checks are not reflected to your virtual machine (*displayed for virtual 3800 printers only*). NODATCK is the default setting and is usually preferred, since specifying DATCK severely increases the overhead associated with simulation of WRITE and SKIP CCWs to the virtual 3800. Generally, the reflection of data checks due to overprinting and invalid EBCDIC codes is not necessary; therefore, specify DATCK *only* when absolutely necessary.

type
: is the device type.

EOF
: means that CP automatically closes the device when 50,000 records are placed in the file that is currently being created.

CLOSED
: means CP is not processing an open spool file for the device. If CP is processing an open spool file, this field says OPEN *spid* (where *spid* is the identification number of the open spool file).

NOKEEP
: means that CP deletes the spool file that was created on this virtual device after it processes the file on a real printer or punch.

NOMSG
means that CP does not inform you when it prints or punches spool files created on this virtual device.

NONAME
means that CP does not assign the spool file a file name or file type when it processes the file.

Note: For more information on spooling options, see *z/VM: CP Commands and Utilities Reference*.

Changing the Spooling Options of Your Printer or Punch

When you change the options on a spooled printer or punch, you change the options that CP assigns to all printer or punch files subsequently created on that device. You also change the options on any file that is currently open on the device. If you want to change the options on a spool file you have already created, see "Changing the Spooling Options for a Spool File" on page 82. Use the SPOOL command to change a spool option setting on a spooled printer or punch. The general format of the SPOOL command is:

```
spool vdev options
```

Where:

vdev
is the virtual device number of the device.

options
are the new options you want to assign to the device.

If you have one or more spooled printers or punches, and you want to change the options on all of them, you can enter the corresponding command:

```
spool printer options
spool punch options
```

where *options* are the new options you want to assign to the device.

Example: Changing the Hold Status on a Spooled Printer or Punch

When you place a spooled printer or punch in user-hold status, CP places in user-hold status all files you create on that spooled printer or punch.

To place your spooled printer or punch in user-hold status, enter:

```
spool printer hold
```

or

```
spool punch hold
```

To remove a spooled printer or punch from user-hold status, enter:

```
spool printer nohold
```

or

```
spool punch nohold
```

Note: Changing the hold status on a spooled printer or punch does *not* change the status of closed spool files already queued for the device. To change the hold status on a closed spool file, see "Example: Preventing CP from Processing a Spool File (HOLD and NOHOLD)" on page 83.

Example: Changing the Class of a Spooled Printer or Punch

When you change the class of a spooled printer or punch, CP assigns that class to all files you create on that device. To change the class of a spool file you have already created, see "Example: Changing a Spool File's Class" on page 85.

To change the class of your spooled printer or punch, enter:

```
spool printer class x
```

or

```
spool punch class x
```

where *x* is the new class that you want to assign.

Example: Changing the Destination Value of a Spooled Printer or Punch

When you change the destination value of a spooled printer or punch, CP assigns that destination value to all files you create on that device.

To change the destination value of your spooled printer or punch, enter:

```
spool printer dest dddddddd
spool punch dest dddddddd
```

or

```
spool printer dest any
spool punch dest any
```

or

```
spool printer dest off
spool punch dest off
```

Where:

dddddddd
specifies a 1- to 8-character destination value for resulting spool files.

any
specifies that the resulting spool files can be processed on any CP output device that meets other selection criteria, regardless of the device's destination setting.

off
specifies that the resulting spool files will only be processed by a device specifically started with, or defaulted to, the OFF option.

Sending Output to Other Users

To send printer or punch output to another user's spooled device, use the SPOOL command. (To send a specific spool file that you already created to another user or device, see "Sending Spool Files to Other Users and Devices" on page 86.)

Example: Sending Printed Output to Another User's Reader

If, instead of sending output to your spooled printer, you want to send output to the spooled reader of the user whose identification is HUNDLEY, first enter:

```
spool printer to hundley reader
```

Then print the information using the command appropriate to the operating system you are running.

Example: Sending Printed Output to Another User's Printer

To send output to HUNDLEY's spooled printer, rather than your spooled printer, first enter:

```
spool printer for hundley printer
```

Then print the information using the command appropriate to the operating system you are running.

Example: Sending Punched Output to Another User's Reader

If, instead of sending output to your spooled punch, you want to send output to the spooled reader of the user whose identification is HICKMAN, first enter:

```
spool punch to hickman reader
```

Then punch the information using the command appropriate to the operating system you are running.

Example: Sending Punched Output to Another User's Punch

To send output to HICKMAN's spooled punch, rather than your spooled punch, first enter:

```
spool punch for hickman
```

Then punch the information using the command appropriate to the operating system you are running.

Resetting Your Printer or Punch

If, after you set your spooled printer or punch to send output to another user, you want to resume sending output to your own spooled device, enter:

```
spool printer to * printer
```

or

```
spool punch to * punch
```

Note: If an ESM is installed, you may not be authorized to enter the SPOOL command with the TO or FOR options. For additional information, contact your security administrator.

Adding Spooled Devices

This section explains how to temporarily define spooled devices other than 3800 printers. For more information on how to temporarily define spooled 3800 printers, see "Defining a Virtual 3800 Printer" on page 105. (When CP logs off your virtual machine, it detaches any devices you used the DEFINE command to create. To add a device to your virtual machine I/O configuration permanently, you must update your z/VM user directory entry.)

To temporarily define a spooled printer (other than a spooled 3800 printer), punch, or reader, enter:

```
define devtype as vdev
```

Where:

devtype
is one of the following:

READER
PRINTER
PUNCH
1403
3203
3211
3262
3505
3525
3800
3800-1
3800-3
4245

4248
VAFP.

vdev
is the virtual device number you assign the device.

Example: Defining a Spooled 3203 Printer

To temporarily define a spooled 3203 printer and assign it virtual device number 888, enter:

```
define 3203 as 888
```

Detaching Spooled Devices

To detach a virtual unit record device from your virtual machine, enter:

```
detach vdev
```

where *vdev* is the virtual device number of the device you want to detach.

Changing the Forms Control Buffer for a Spooled 3203, 3211, 3262, 4245, or 4248 Impact Printer

To set up a forms control buffer (FCB) for a spooled 3203, 3211, 3262, 4245, or 4248 impact printer, enter:

```
loadvfcb vdev fcb name image imagelib
```

Where:

vdev
is the virtual device number of your virtual 3203, 3211, 3262, 4245, or 4248 impact printer.

name
is one of the IBM*-provided buffers (FCB1, FCB8, or FCBS) or the name of a forms control buffer that your installation has defined.

imagelib
is the name of the image library that the FCB resides in.

Note:

1. When you specify the FCB for your spooled 3203, 3211, 3262, 4245, or 4248 impact printer, specify the same FCB that will be available on the real printer when the file is actually printed. If the forms control buffers are not the same, printing errors occur.
2. To load the FCB with a specified image for a *real* IBM 3203 Model 5, 3211, 3262, 4245, or 4248 printer, use the LOADBUF command.
3. For more information on how to define a forms control buffer, see *z/VM: CP Planning and Administration*.
4. For more information on the image library, see *z/VM: CP Planning and Administration*.

Defining a Virtual 3800 Printer

You can use the DEFINE command to add a virtual 3800 Model 1 or 3800 Model 3 printer to your virtual machine I/O configuration. Before you attempt to use a virtual 3800, remember you:

- Do not need to define a virtual 3800 printer to have spool files printed on a real 3800 printer.
- Must provide the image libraries that contain the data for any 3800 load modules. You select the load modules for the file through the SPOOL or CHANGE command.

- Cannot define a 3800 Model 3 with two writable-character generation modules (WCGMs). You must always have four WCGMs with a 3800 Model 3.

You must use the CP DEFINE command to define a virtual 3800 printer, but if you are running CMS in your virtual machine you can use the SETPRT command to specify various characteristics of the printer. For more information about the SETPRT command, see *z/VM: CMS Commands and Utilities Reference*.

Example 1: Defining a Virtual 3800 Using Default Attributes

Assume that you want to define a virtual 3800 Model 1 or 3800 Model 3 at virtual device number 555 with the following attributes:

- Forms 14-7/8 inches wide by 11 inches long (378 mm by 280 mm)
- Four writable-character generation modules (WCGMs)
- Continuous forms stacker (CFS)
- No data checks reflected to your virtual machine.

Enter the following command to define a virtual 3800 Model 1:

```
define 3800 as 555
```

CP assigns by default the attributes listed above.

Enter the following command to define a virtual 3800 Model 3:

```
define 3800-3 as 555
```

CP assigns by default the attributes listed above.

Example 2: Defining a Virtual 3800 Model 1 with Two WCGMs

Assume that you want to define a virtual 3800 Model 1 at virtual device number 555 with the following attributes:

- Forms 14-7/8 inches wide by 11 inches long (378 mm by 280 mm)
- Two writable-character generation modules (WCGMs)
- Continuous forms stacker (CFS)
- No data checks reflected to your virtual machine.

Enter the following command:

```
define 3800 as 555 2wcgm
```

CP assigns by default the size of the paper and the type of stacker.

Example 3: Defining a Virtual 3800 with a Burster–Trimmer–Stacker

Assume that you want to define a virtual 3800 Model 1 or 3800 Model 3 at virtual device number 654 with the following attributes:

- Forms 14-7/8 inches wide by 11 inches long (378 mm by 280 mm)
- Four writable-character generation modules (WCGMs)
- Burster-trimmer-stacker (BTS)
- No data checks reflected to your virtual machine.

Enter the following command to define a virtual 3800 Model 1:

```
define 3800 as 654 bts
```

CP assigns by default the size of the paper and the number of WCGMs.

Enter the following command to define a virtual 3800 Model 3:

```
define 3800-3 as 654 bts
```

CP assigns by default the size of the paper and the number of WCGMs.

Example 4: Defining a Virtual 3800 and Reflecting Data Checks

Assume that you want to define a virtual 3800 Model 1 or 3800 Model 3 at virtual device number 455 with the following attributes:

- Forms 14-7/8 inches wide by 11 inches long (378 mm by 280 mm)
- Four writable-character generation modules (WCGMs)
- Continuous forms stacker (CFS)
- Data checks reflected to your virtual machine.

Enter the following command to define a virtual 3800 Model 1:

```
define 3800 as 455 datck
```

Note: Use DATCK only when absolutely necessary. It severely increases the overhead associated with simulation of WRITE and SKIP CCWs to the virtual 3800. In general, the reflection of data checks due to overprinting and invalid EBCDIC codes is not necessary.

CP assigns by default the size of the paper, the number of WGGMs, and the type of stacker.

Enter the following command to define a virtual 3800 Model 3:

```
define 3800-3 as 455 datck
```

CP assigns by default the size of the paper, the number of WGGMs, and the type of stacker.

Example 5: Defining a Virtual 3800 with Optional Forms Size

Assume that you want to define a virtual 3800 Model 1 or 3800 Model 3 at virtual device number 555 with a form size other than the default value and the following attributes:

- Four writable-character generation modules (WCGMs)
- Continuous forms stacker (CFS)
- No data checks reflected to your virtual machine.

Enter the following command to define a virtual 3800 Model 1:

```
define 3800 as 555 size width length
```

Enter the following command to define a virtual 3800 Model 3:

```
define 3800-3 as 555 size width length
```

Where:

width
is one of the following hexadecimal codes:

Code (Hex)	Width in Inches	Width in Millimeters
01	6-1/2	165
02	Reserved	180
04	8-1/2	215

Code (Hex)	Width in Inches	Width in Millimeters
06	9-1/2	235
07	9-7/8	250
08	10-5/8	270
09	11	280
0A	12	305
0B	Reserved	322
0D	13-5/8	340
0E	14-3/10	363
0F	14-7/8	378

length
is the length of the paper in half-inches. For example, if you want 11-inch paper, specify the length as 22.

For more information on the DEFINE (Spooling Devices), see *z/VM: CP Commands and Utilities Reference*.

Chapter 8. Dumping, Displaying, Altering Virtual Machine Storage

This chapter contains descriptions on:

- Using the DUMP and DISPLAY commands to dump and display:
  - Virtual machine storage
  - Virtual machine general-purpose, floating-point, control, and access registers
  - Virtual machine program status words (PSWs)
  - Subchannel information blocks (SCHIBs)
- Using the STORE command to store data in:
  - Virtual machine storage
  - Virtual machine general-purpose, floating-point, control, and access registers

Dumping Information from Your Virtual Machine's Storage

CP has two commands you can use to create virtual machine dumps:

- VMDUMP
- DUMP

Use the VMDUMP command to create and format dumps you plan to view using z/VM dump viewing facility. For more information about the VMDUMP command and the dump viewing facility, see *z/VM: CP Commands and Utilities Reference* and *z/VM: Dump Viewing Facility*.

In order to view CP type information in a dump you will need to use the VM Dump Tool. For more information on this tool see *z/VM: VM Dump Tool*.

Use the DUMP command to dump to your virtual printer the contents of any of the following:

- Virtual machine storage
- Virtual machine registers
- Old and new PSWs
- Storage keys
- Subchannel information blocks

Example: Dumping Storage

To dump the contents of the fullword of virtual machine storage at location 20000, enter:

```
dump 20000.4
```

Example: Dumping Storage

To dump the contents of virtual machine storage locations 45600 to 52000, enter:

```
dump 45600-52000
```

Example: Dumping Prefix Register

To dump the contents of your virtual machine's prefix register, enter:

```
dump prefix
```

Example: Dumping Current PSW

To dump the contents of your virtual machine's current PSW, enter:

```
dump psw
```

If the virtual machine is in z/Architecture or z/XC mode, enter:

```
dump pswg
```

Example: Dumping Old and New PSWs

To dump the contents of all of your virtual machine's old and new PSWs, enter:

```
dump psw all
```

If the virtual machine is in z/Architecture or z/XC mode, enter:

```
dump pswg all
```

Note: Substitute one of the following operands for ALL to dump the contents of specific PSWs:

EXT
Dumps old and new external interrupt PSWs

I/O
Dumps old and new I/O interrupt PSWs

MCH
Dumps old and new machine check PSWs

PRG
Dumps old and new program check PSWs

SVC
Dumps old and new SVC PSWs

Example: Dumping General Registers

To dump the contents of your virtual machine's general registers, enter:

```
dump g
```

This command dumps the full 32-bit registers for an ESA/390 mode or ESA/XC mode virtual machine, or the lower half of the registers for a z/Architecture or z/XC mode virtual machine.

To dump the full 64-bit registers for a z/Architecture or z/XC virtual machine, enter:

```
dump gg
```

Example: Dumping Control Registers

To dump the contents of your virtual machine's control registers, enter:

```
dump x
```

This command dumps the full 32-bit registers for an ESA/390 mode or ESA/XC mode virtual machine, or the lower half of the registers for a z/Architecture or z/XC mode virtual machine.

To dump the full 64-bit registers for a z/Architecture or z/XC virtual machine, enter:

```
dump xg
```

Example: Dumping Access Registers

To dump the contents of your virtual machine's access registers, enter:

```
dump ar
```

Example: Dumping Floating-Point Registers

To dump the contents of your virtual machine's floating-point registers, enter:

```
dump y
```

Example: Dumping Subchannel Information Blocks (SCHIBs)

To dump the contents of your virtual machine's subchannel information blocks (XA virtual machines only), enter:

```
dump schib all
```

Note: To dump the contents of particular subchannel information blocks, replace ALL with the addresses of those blocks (for example, DUMP SCHIB 0001 or DUMP SCHIB 0001-0011).

Example: Dumping Linkage Stack

To dump the state entries of a guest's linkage stack, enter:

```
dump lks
```

This command is valid only if the virtual machine is an XA, ESA, or Z virtual machine.

Displaying Information from Your Virtual Machine's Storage

If, instead of dumping information to your virtual printer, you want to display that information, use the DISPLAY command. The DISPLAY command displays at your virtual machine operator's console the same information that the DUMP command dumps to your printer.

Use the LOCATEVM command to search for certain data (strings) in your virtual machine's storage. For more information about the LOCATEVM command, see *z/VM: CP Commands and Utilities Reference*.

Example: Displaying Storage

To display the contents of the fullword of virtual machine storage at location 20000, enter:

```
display 20000.4
```

To display the contents of virtual machine storage locations 45600 to 52000, enter:

```
display 45600-52000
```

Example: Displaying Prefix Register

To display the contents of your virtual machine's prefix register, enter:

```
display prefix
```

Example: Displaying Current PSW

To display the contents of your virtual machine's current PSW, enter:

```
display psw
```

If the virtual machine is in z/Architecture or z/XC mode, enter:

```
display pswg
```

Example: Displaying Old and New PSWs

To display the contents of all of your virtual machine's old and new PSWs, enter:

```
display psw all
```

If the virtual machine is in z/Architecture or z/XC mode, enter:

```
display pswg all
```

Note: Substitute one of the following operands for ALL to display the contents of specific PSWs:

EXT
Displays old and new external interrupt PSWs

I/O
Displays old and new I/O interrupt PSWs

MCH
Displays old and new machine check PSWs

PRG
Displays old and new program check PSWs

SVC
Displays old and new SVC PSWs

Example: Displaying General Registers

To display the contents of your virtual machine's general registers, enter:

```
display g
```

This command displays the full 32-bit registers for an ESA/390 mode or ESA/XC mode virtual machine, or the lower half of the registers for a z/Architecture or z/XC mode virtual machine.

To display the full 64-bit registers for a z/Architecture or z/XC virtual machine, enter:

```
display gg
```

Note: If UNDERSCORE is set ON, an underscore character will be used to separate the 16-digit output into two 8-digit entities for readability (for example, `AAAAAAAA_00F7D140`). Use the QUERY UNDERSCORE command to determine the current setting. Use the SET UNDERSCORE command to change the setting.

Example: Displaying Control Registers

To display the contents of your virtual machine's control registers, enter:

```
display x
```

This command displays the full 32-bit registers for an ESA/390 mode or ESA/XC mode virtual machine, or the lower half of the registers for a z/Architecture or z/XC mode virtual machine.

To display the full 64-bit registers for a z/Architecture or z/XC virtual machine, enter:

```
display xg
```

Note: If UNDERSCORE is set ON, an underscore character will be used to separate the 16-digit output into two 8-digit entities for readability (for example, AAAAAAAA_00F7D140). Use the QUERY UNDERSCORE command to determine the current setting. Use the SET UNDERSCORE command to change the setting.

Example: Displaying Access Registers

To display the contents of your virtual machine's access registers, enter:

```
display ar
```

Example: Displaying Floating-Point Registers

To display the contents of your virtual machine's floating-point registers, enter:

```
display y
```

Example: Displaying Subchannel Information Blocks

To display the contents of your virtual machine's subchannel information blocks, enter:

```
display schib all
```

Note: To display the contents of particular subchannel information blocks, replace ALL with the addresses of those blocks (for example, `display schib 0001` or `display schib 0001-0011`).

Example: Displaying Linkage Stack

To display the state entries of a guest's linkage stack, enter:

```
display lks
```

This command is valid only if the virtual machine is an XA, ESA, or Z virtual machine.

Example: Displaying CPU Timer and Clock Comparator

To display the contents of the CPU timer and the clock comparator, use two commands.

Attention: Using the SYSTEM STORE STATUS command can change values contained in your virtual machine's low storage and make it impossible for your virtual machine to continue to execute.

First, enter:

```
system store status
```

Then, for an ESA/390 (31-bit) virtual machine, enter:

```
display d8.10
```

Or, for a z/Architecture or z/XC (64-bit) virtual machine, enter:

```
display 1328.10
```

Altering the Contents of Your Virtual Machine's Storage

To change the contents of your virtual machine's storage or registers, use the STORE command.

Note: You can also use the STORE command to change your virtual machine's current PSW, CSW, or CAW. For more information, see *z/VM: CP Commands and Utilities Reference*.

Example: Altering Storage

To store the hexadecimal value 0000 00AA at the fullword at virtual machine storage location 20000, enter:

```
store 20000 aa
```

Example: Altering General Registers

To store the hexadecimal value 0000 0010 into your virtual machine's general register 1, enter:

```
store g1 00000010
```

This command stores the value into the 32-bit register for an ESA/390 mode or ESA/XC mode virtual machine, or into the lower half of the 64-bit register for a z/Architecture or z/XC virtual machine.

To store the hexadecimal value 00000010 00000010 into the full 64-bit register for a z/Architecture or z/XC mode virtual machine, enter:

```
store gg1 00000010_00000010
```

Note: The underscore separator is optional.

Example: Altering Control Registers

To store the hexadecimal value 0000 0020 into your virtual machine's control register 11, enter:

```
store x11 00000020
```

This command stores the value into the 32-bit register for an ESA/390 mode or ESA/XC mode virtual machine, or into the lower half of the 64-bit register for a z/Architecture or z/XC virtual machine.

To store the hexadecimal value 00000010 00000020 into the full 64-bit register for a z/Architecture or z/XC mode virtual machine, enter:

```
store xg11 00000010_00000020
```

Note: The underscore separator is optional.

Example: Altering Access Registers

To store the hexadecimal value 0000 0010 into your virtual machine's access register 5, enter:

```
store ar5 00000010
```

Example: Altering Floating-Point Registers

To store Avogadro's number (6.023×10^{23}) into floating-point register 6, enter:

```
store y6 547f8abf98bdd5ae
```

Chapter 9. Tracing Programs in Your Virtual Machine

The virtual machine tracing facility allows you to follow or trace the execution of almost anything that takes place while you are using your virtual machine. For example, you can trace instructions, I/O interrupts, branches, and changes to storage. You can limit tracing to a specified storage range and to events that affect a specific device.

This chapter explains how to use the TRACE command. It describes what events you can trace, how to create trace sets and trace traps, and how you can manipulate your trace sets once you begin tracing. As you become familiar with the TRACE command, use the information in Chapter 10, “Creating Trace Traps (Examples),” on page 125, to help you create trace traps.

This topic contains descriptions about:

- Trace sets and trace traps
- Events you can trace using the TRACE command
- Creating a trace set
- Creating trace traps
- Modify existing trace traps
- Passing control between new and existing trace sets.

Tracing: Two Terms to Know

Before you can use the TRACE command, you must know the meaning of a *trace trap* and *trace sets*.

A *trace trap* is a unit of control that allows you to tell CP what event you want to trace and the conditions under which you want the event traced. Create trace traps by entering a command similar to the following:

```
trace operand option
```

Where:

operand
: is a keyword that specifies the type of event you want to trace (it can also be called a primary operand).

option
: are keywords that specify the conditions under which you want the event traced, or an action you want CP to perform when it traces a particular event.

For example, if you want to trace all instructions that occur between storage locations 20000 and 30000, you need to create the following trace trap:

```
trace instruction from 20000-30000
```

Where:

instruction
: is the operand that specifies the type of event you want to trace (in this case, instructions).

from 20000–30000
: is an option that specifies the conditions under which you want to trace instructions (in this case, only when the instructions occur between storage locations 20000 and 30000).

A *trace set* is a collection of one or more trace traps that are active at the same time. To trace different types of events, you can pass control between different trace sets. A trace set can also contain no trace traps, in which case the trace set is called a *null trace set*. You can use null trace sets to temporarily stop tracing.

Tracing: An Overview

This chapter describes one method you can use for tracing programs. This is *not* the only method, nor is it necessarily the best method. You should use whatever method works best for you. The following is a general method for tracing:

1. Choose an event (or events) that you want to trace. Although you must decide what events you want to trace, this chapter describes the events you can trace.
2. Name a trace set and create the appropriate traps.
3. Run the program that you want to trace. (You can trace anything that runs in your virtual machine.) Modify your existing trace sets or create new sets as needed.

Note: Unless you specify otherwise, every time CP *traps* an event it stops your virtual machine and displays the status indicator CP READ on the bottom right of your display screen. (You can use any of the following options to control the way CP stops your virtual machine after it traps an event: RUN, NORUN, SKIP, PASS, STEP, or STOP.) While your virtual machine is stopped, you can enter any CP command. For example, you may want to display the contents of your registers or of a particular storage location.

When you are ready to restart the program you are tracing, enter:

```
begin
```

A Sample Trace Session

The following sections describe a sample trace session. The purpose of this sample is to show how, when, and why to use the TRACE command to trace programs you run in your virtual machine.

Step 1. Choose an Event to Trace

When you begin tracing a program, the first question you must ask yourself is, *What events do I want to trace?* This question may lead to another: *What events can I trace?*

The table shown in "Events That You Can Trace" on page 125 lists all the events you can trace and the TRACE command operand that corresponds to each event. For example:

- If you want to trace branch instructions, use the BRANCH operand.
- If you want to trace all I/O instructions and I/O interrupts, or both, use the I/O operand.

You may use more than one operand to trace certain events. For example:

- To trace all instructions, use the INSTRUCTION operand.
- To trace only I/O instructions, use the I/O operand.
- To trace only SIO instructions, use the SIO operand.

Step 2. Create a Trace Set and Trace Traps

Assume for this sample trace session that you decide to trace all instructions a program issues. Referring to the table shown in "Events That You Can Trace" on page 125, you can see that the first operand listed (INSTRUCTION) lets you trace instructions.

After you decide what event you want to trace, the next thing to do is create a trace set and trace traps. To create a trace set named SET1, enter:

```
trace goto set1
```

Where:

goto
    tells CP to create a trace set.

set1
is what you want CP to name the trace set.

Now you must create a trace trap to tell CP that you want to trace all instructions. Again, referring to "Events That You Can Trace" on page 125, you can see that "Using TRACE INSTRUCTION" on page 126 contains examples of using the INSTRUCTION operand. The first example in "Using TRACE INSTRUCTION" on page 126 shows that to trace all instructions you must enter:

```
trace instruction
```

Where:

instruction
is the operand that tells CP what event to trace.

Note: If you do not create a trace set before you create a trace trap, CP creates a trace set for you, and gives it the default name of INITIAL.

Step 3. Tracing

After creating your trace set and trace trap, run the program you want to trace. Remember, the program can be anything you run in your virtual machine (for example, your operating system, a program product, or a user application). The first time the program issues an instruction, CP *traps* it, displays some information about the instruction, stops your virtual machine, and displays CP READ on the lower right of your display screen.

As shown in the following example, the instruction executed at storage location 20000, the assembler mnemonic for the instruction was B, the machine code for the instruction was 47F0C00A, the instruction branched to location 0002000A, and the condition code was 0. For complete information about interpreting trace output, see *z/VM: CP Commands and Utilities Reference*.

```
-> 00020000  B      47F0C00A -> 0002000A    CC 0
```

To restart the program you are running, enter:

```
begin
```

As instructions continue to run, CP continues to *trap* them, displays information about them, stops your virtual machine, displays CP READ on the lower right of your display screen, and waits for you to restart your virtual machine.

When your virtual machine displays CP READ in the lower right of your display, you can use the DISPLAY command described in Chapter 8, "Dumping, Displaying, Altering Virtual Machine Storage," on page 109 to display information about your virtual machine.

You must continue to enter BEGIN to restart your virtual machine.

Example: Modifying an Existing Trace Trap

Tracing every instruction your virtual machine issues is a slow process. Hundreds of instructions may run before you find one of interest. Therefore, you may want to modify your existing trace trap to do the following:

- Use the PSWA option to tell CP to trace only those instructions your program issues within a given storage range.

  Note: Remember, when you create a trace trap, you can include various options that limit the conditions under which an event is traced. For example, PSWA is an option that specifies a storage range in which events are traced. See "Using TRACE I/O" on page 135 for descriptions on the options you can use with the INSTRUCTION operand.
- Use the STEP option to tell CP to stop your virtual machine only after it traces a given number of instructions.

To modify an existing trace trap, you must first find out the identification of the trace trap. To do this, enter:

```
query trace
```

In response to this command, CP displays information similar to the following:

```
NAME  SET1        (ACTIVE)

1     INSTR   PSWA  00000000-7FFFFFFF
      TERM    NOPRINT  NORUN  SIM
      SKIP 00000  PASS 00000  STOP 00000  STEP 00000
      CMD  NONE
```

In this case, the identification of the trace trap is 1. (The rest of the information tells you the options that are currently in effect for your trace trap.)

Now, assume that you want to modify trace trap 1 so that:

- CP traces only instructions your program issues between storage locations 30000 and 30100
- CP stops your virtual machine only after every 10 instructions run.

To do this, enter:

```
trace trap 1 pswa 30000-30100 step 10
```

Where:

trap 1
tells CP what trace trap you want to modify.

pswa 30000–30100
tells CP to trace only those instructions your virtual machine issues between storage locations 30000 and 30100.

step 10
tells CP to stop your virtual machine when it traps every 10th event. For the other nine events, CP displays data but does not stop your virtual machine.

If you reenter the QUERY TRACE command, CP responds with information similar to the following:

```
NAME  SET1        (ACTIVE)

1     INSTR   PSWA  00030000-00030100
      TERM    NOPRINT  NORUN  SIM
      SKIP 00000  PASS 00000  STOP 00010  STEP 00010
      CMD  NONE
```

The information after PSWA, STOP, and STEP is changed. STOP 10 means CP stops your virtual machine after the first 10 events. Using the STEP option automatically sets the STOP option unless you specify a different value with STOP.

Example: Adding Trace Traps to a Trace Set

In addition to modifying existing trace traps, you can create new trace traps. For example, assume that you decide to trace instructions between storage locations 40000 and 40100, as well as between storage locations 30000 and 30100. Also assume that you do not want CP to stop your virtual machine when it traps this new set of instructions. To do this, enter:

```
trace instruction pswa 40000-40100 run
```

Where:

pswa 40000–40100
tells CP to trace all instructions that your virtual machine issues between storage locations 40000 and 40100.

run
: is an option that tells CP not to stop your virtual machine when it traps instructions between locations 40000 and 40100. (CP still displays data about the instructions.)

If, after you add this trap, you reenter the QUERY TRACE command, CP responds with information similar to the following:

```
NAME  SET1        (ACTIVE)

1     INSTR   PSWA  00030000-00030100
      TERM    NOPRINT  NORUN  SIM
      SKIP 00000  PASS 00000  STOP 00010  STEP 00010
      CMD  NONE

2     INSTR   PSWA  00040000-00040100
      TERM    NOPRINT  RUN    SIM
      SKIP 00000  PASS 00000  STOP 00000  STEP 00000
      CMD  NONE
```

Example: Deleting a Trace Trap

Now assume that you no longer need to trace instructions between storage locations 30000 and 30100. To delete this trace trap from your trace set, enter:

```
trace delete 1
```

where delete 1 tells CP to delete trace trap 1 from your trace set.

If, after you delete trace trap 1, you reenter the QUERY TRACE command, CP responds with information similar to the following:

```
NAME  SET1        (ACTIVE)

2     INSTR   PSWA  00040000-00040100
      TERM    NOPRINT  RUN    SIM
      SKIP 00000  PASS 00000  STOP 00000  STEP 00000
      CMD  NONE
```

Example: Passing Control to a New Trace Set

One general strategy you may choose to follow when tracing is to look for ways to make your traps more specific. This sample trace session starts by tracing all instructions. It then becomes more specific by limiting the storage range in which instructions are traced.

Now assume that when you are tracing instructions and you find a problem: bad data is being stored in location 41000. To create a trap for this problem, you can modify SET1. You can also pass control to a new trace set.

To pass control to a new trace set named SET2, enter:

```
trace goto set2
```

After you name your trace set, create a trap for all instructions that change storage location 41000. The table in "Events That You Can Trace" on page 125 shows that the operand STORE corresponds to instructions that change storage, and that "Using TRACE STORE" on page 132 contains examples of using the STORE operand.

To create a trace trap for all instructions that change storage location 41000, enter:

```
trace store into 41000
```

where into 41000 tells CP to trace only those instructions that store data into storage location 41000.

If, after you name your trace set SET2 and create your trace trap, you enter the QUERY TRACE command, CP responds with information similar to the following:

```
NAME  SET2        (ACTIVE)

  1     STORE   FROM  00000000-7FFFFFFF
                INTO  00041000
        TERM    NOPRINT  NORUN  SIM
        SKIP 00000  PASS 00000  STOP 00000  STEP 00000
        CMD  NONE
```

As you continue tracing, try finding ways to make your trace trap more specific. The following examples show some possibilities.

Example 1

If you know that the instruction storing the bad data is running in a specific storage range (for example, between 40000 and 40500), enter:

```
trace trap 1 from 40000-40500
```

Where:

trap 1
tells CP you want to modify the trap with the identification 1.

from 40000–40500
tells CP to trace only those instructions that run between storage locations 40000 and 40500.

Example 2

If you want CP to display the value of storage location 41000 after it traps instructions, enter:

```
trace trap 1 cmd display 41000
```

Where:

trap 1
tells CP you want to modify the trap with the identification 1.

cmd display 41000
tells CP to run the DISPLAY 41000 command after it traps an instruction.

Example 3

If you discover that the FFFF is the bad data being stored in location 41000–41001, delete trap 1 and create the following trap:

```
trace store into 41000 data ffff from 40000-40500
```

Where:

into 41000 data ffff
tells CP to trap only those instructions that store the value FFFF in location 41000-41001.

from 40000–40500
tells CP to trace only those instructions that execute between storage locations 40000 and 40500.

To delete trap 1, enter:

```
trace delete 1
```

Note: For a list of the options modified by the TRACE TRAP command, see Appendix A, “Trace Trap Common Options,” on page 143.

In this example, the options INTO and DATA are used. Therefore, a new trap is created and the old trap is deleted.

Example: Temporarily Stopping Tracing

Assume now that you find and fix all the instructions that are storing the value FFFF in location 41000. You now want to run your program without tracing any events; however, you also want CP to save your existing trace sets. To do this, pass control to a null trace set. (A null trace set has no traps.) To pass control to a null trace set, enter:

```
trace goto name
```

where *name* is a name you assign to the null trace set. (You can assign any 1- to 8-character name.)

After you enter this command, restart the program you are tracing. Because a null trace set has no traps, CP does not trap any events. If the program runs satisfactorily, you can end your trace session. If the program still has problems, you can pass control to a new or existing trace set.

Example: Ending Your Trace Session

Finally, when you are done tracing, enter:

```
trace end
```

This deletes all of your existing trace sets and traps.

Tracing—A Summary

The following sections summarize the steps and tasks you may need to perform while tracing programs in your virtual machine.

Choose an Event to Trace

Use the table in "Events That You Can Trace" on page 125 to find:

- Events you can trace
- TRACE command operands that correspond to events you can trace
- The page in Chapter 10, "Creating Trace Traps (Examples)," on page 125 that contains examples of how to use a specific operand to trace events.

Create a Trace Set and Trace Traps

Trace traps must be contained in a trace set. Therefore, before you can create a trace trap, you must create a trace set. (If you do not specifically name a trace set, the first time you create a trace trap, CP places it in a trace set named INITIAL.) To create a trace set, enter:

```
trace goto name
```

where *name* is the 1- to 8-character name for your trace set.

A *trace trap* is a unit of control that allows you to tell CP what event you want to trace and the conditions under which you want the event traced. Create trace traps by entering a command similar to the following:

```
trace operand option1 option2...
```

Where:

operand
is a keyword that specifies the type of event you want to trace. (It can also be called a primary operand.)

option1 option2...
are keywords that specify the conditions under which you want the event traced, or an action you want CP to perform when it traces a particular event.

Again, use "Events That You Can Trace" on page 125 to find out the:

- Operand you must use to trace a specific event
- Page in Chapter 10, "Creating Trace Traps (Examples)," on page 125 that describes the operand and options that apply to it.

There are two types of options: options that apply only to specific operands, and options common to all operands. Briefly, the common options let you do the following:

- Specify a storage range in which instructions are traced (PSWA, FROM, and RANGE options)
- Specify a specific guest address space in which events are traced (PRI, SECO, HOME, ASTE*raddr*, STD*hexword*, STO*raddr*, ASN*asn*, AREG*areg*, ALET*hexword*, and ALET*hexword*.AL*raddr* options)
- Specify how often CP stops your virtual machine as it traces events (RUN, NORUN, SKIP, PASS, STEP, or STOP options)
- Specify where CP sends information about events it traces (TERMINAL, NOTERMINAL, PRINTER, NOPRINTER, and BOTH options)
- Specify a CP command that CP executes when it traces an event (CMD option)
- Limit tracing to particular virtual machine *states* or *modes* (SUPERVISOR, PROBLEM, DAT, and NODAT options)
- Limit tracing to certain instructions or a sequence of instructions (DATA option)
- Count the successful trace events in your virtual machine (COUNT option)
- Display a list of up to six of the last successful branch instructions that were executed while branch tracing was active (TABLE option)
- Merge traps from the named trace set with the current trace set (APPEND option).

There are too many common options to explain how to use all of them with each operand. Therefore, see "Examples of Using Trace Trap Common Options" on page 126 for more information on what each common option does, and for examples of how to use each common option with the TRACE INSTRUCTION operand.

Tracing

Once you create your trace set and trace traps, you are ready to run the program that you want to trace. Whenever CP encounters an event that you created a trap for, CP stops your virtual machine, displays CP READ at the lower right of your display, and displays various data about the event. (You can specify one of the following options to control how frequently CP stops your virtual machine: RUN, NORUN, SKIP, PASS, STEP, or STOP.)

To restart your virtual machine after CP has stopped it, enter:

```
begin
```

Displaying Your Current Trace Settings

Use the QUERY TRACE command to obtain information about your trace sets and trace traps.

Example: Displaying Information about Your Active Trace Set

To determine what events you are currently tracing and the options you specified for those events, enter:

```
query trace
```

Example: Displaying Information about All Your Trace Sets

To display the contents of all your trace sets, enter:

```
query trace all
```

Example: Displaying Information about a Specific Trace Set

To display the contents of a specific trace set, enter:

```
query trace name
```

where *name* is the name of the trace set you want to find out about.

Example: Displaying the Names of All Your Trace Sets

To display the names of all your existing trace sets, enter:

```
query trace sets
```

Changing the Options on a Trace Trap

Use the TRACE TRAP command to change options on a particular trap. The general format of TRACE TRAP is:

```
trace trap name options
```

Where:

name
is the name of the trace trap you want to change. It can be an optional 1- to 4-character name that you give the trace trap or a unique number that CP gives the trace trap. (Use the QUERY TRACE ALL command to display the names and contents of all of your trace traps.)

options
are the options you want to change.

Note: For a list of the options modified by the TRACE TRAP command, see Appendix A, “Trace Trap Common Options,” on page 143.

To change other options, you must create a new trap and delete the old one. If you use the TRACE TRAP command to modify options other than those listed in Appendix B, you receive a message similar to the following:

```
INVALID OPERAND - option
option is the option you cannot use the TRACE TRAP
command to modify
```

Deleting Trace Traps from a Trace Set

To delete a particular trap from a trace set, enter:

```
trace delete name
```

where *name* is the name of the trace trap you want to delete. It can be an optional 1- to 4-character name that you give a trace trap or a unique number that CP gives the trace trap. (Use the QUERY TRACE ALL command to display the names and contents of all of your trace traps.)

Passing Control to a New or Existing Trace Set

During a particular tracing session, you can define several trace sets, each containing a number of trace traps. However, only one trace set can be active at a time. There are three methods you can use to pass control to a new or existing trace set; this book describes one of these methods. (You can also use the TRACE CALL and TRACE RETURN commands to pass control among trace sets. For more information about these commands, see *z/VM: CP Commands and Utilities Reference*.)

To pass control to a new or existing trace set, enter:

```
trace goto name
```

where *name* is the 1- to 8-character name of the trace set to which you want to pass control.

Note:

1. You can define a maximum of 100 trace sets and traps in any combination during a particular trace session. (For example, you can create two trace sets with no more than 98 traps between them, five trace sets with no more than 95 traps among them, 20 trace sets with no more than 80 traps among them, and so on.)
2. To find out the names of your existing trace sets, enter QUERY TRACE SETS.
3. If you pass control to an existing trace set, CP puts that trace set's trace traps into effect.
4. If you pass control to a new trace set, you must create the trace traps you want.
5. If you have defined multiple virtual processors for your virtual machine, different trace sets can be active on different virtual processors.

Temporarily Stopping Tracing

To temporarily stop CP from tracing any events, create a null trace set. (A null trace set is a trace set that does not contain any trace traps.) To pass control to a null trace set, enter:

```
trace goto name
```

where *name* is the 1- to 8-character name you assign the null trace set.

Since a null trace set contains no trace traps, the program you are running runs without interruption. To resume tracing, pass control to a new or existing trace set.

Ending a Trace Session

To end a trace session and delete all your existing trace sets, enter:

```
trace end
```

When you enter this command, CP deletes all your existing trace sets and trace traps.

Tracing During Transactional Execution

Events arising from TRACE INSTRUCTION, STORE, BRANCH, G, GG, and AR that occur when the virtual machine is running in transactional execution mode will cause the transaction to abort. To prevent such events from aborting transactions, you can use the TXSUSPEND keyword on the TRACE command. Enter:

```
trace txsuspend
```

Trace suspension during transactional exection can be turned off using the NOTXSUSPEND keyword on the TRACE command. Enter:

```
trace notxsuspend
```

The current state of trace suspension during transactional execution is displayed when trace is queried. Enter:

```
query trace
```

Chapter 10. Creating Trace Traps (Examples)

This topic contains examples of trace traps you can create. If you are unfamiliar with the TRACE command, see Chapter 9, "Tracing Programs in Your Virtual Machine," on page 115 before reading this topic. For more information on the syntax of the TRACE command, see *z/VM: CP Commands and Utilities Reference*. Using the information in this topic, you should be able to create a trace trap for any of the events or operands (or both) listed under "Events That You Can Trace" on page 125.

Events That You Can Trace

Table 1 on page 125 shows:

- Events you can trace
- Operands that go with the events
- The topic that contains examples of how to use the operand.

Note: You may use more than one operand to trace certain events. For example, to trace all instructions, use the INSTRUCTION operand. To trace only I/O instructions, use the I/O operand. To trace only SIO instructions, use the SIO operand.

Table 1. Events You Can Trace (Tracing Programs)

Operand	**Events Traced**	**Examples**
INSTRUCTION	Execution of instructions	"Using TRACE INSTRUCTION" on page 126
ALL	Execution of instructions as well as program, external, and I/O interruptions	"Using TRACE ALL" on page 131
BRANCH	Successful branch instructions, including LPSW and SVC instructions that do not load a wait PSW	"Using TRACE BRANCH" on page 131
STORE	Alterations to virtual machine storage	"Using TRACE STORE" on page 132
GPR	Alterations to any or all of your virtual machine's general-purpose registers	"Using TRACE GPR" on page 132
AR	Alterations to your virtual machine's access registers	"Using TRACE AR" on page 133
SVC DIAGNOSE MC	Supervisor calls (SVC) DIAGNOSE instructions Monitor calls (MC)	"Using TRACE SVC, DIAGNOSE, MC" on page 134
SIO SIOF SSCH RIO RSCH	SIO instructions SIOF instructions SSCH instructions RIO instructions RSCH instructions	"Using TRACE SIO, SIOF, SSCH, RIO, RSCH" on page 137
TPI	Test-Pending-Interrupt instructions	"Using TRACE TPI" on page 138

Table 1. Events You Can Trace (Tracing Programs) (continued)

Operand	Events Traced	Examples

MNEMONIC1	The following assembler mnemonics:
	`BSA PGOUT SSK`
	`BSG PR SSM`
	`CONCS PT STAP`
	`CSP PTI STCPS`
	`DISCS PTLB STCRW`
	`EPSW RCHP STCTG`
	`ISK RDD STCTL`
	`IUCV RDP STFL`
	`LASP RRB STIDC`
	`LCTL RRBE STIDP`
	`LCTLG SAL STNSM`
	`LPSW SCHM STOSM`
	`LPSWE SCK STPX`
	`LPSWEY SERVC STSI`
	`PALB SIE TB`
	`PC SIGP TPROT`
	`PGIN SPX WRD`
		"Using TRACE MNEMONIC1" on page 138

MNEMONIC2	The following assembler mnemonics:
	`CLRCH HIO TCH`
	`CLRIO HSCH TIO`
	`CSCH MSCH TSCH`
	`HDV SIGA STSCH`
		"Using TRACE MNEMONIC2" on page 139

`EXT` `PROG`	External interrupts Program interrupts	"Using TRACE EXTERNAL, PROGRAM" on page 140
I/O	I/O instructions or I/O interrupts (or both) with or without CCW tracing	"Using TRACE I/O" on page 135
MCH	Machine-check interrupts	"Using TRACE MCH" on page 141

Using TRACE INSTRUCTION

Use TRACE INSTRUCTION to trace instructions in a program, whether they run successfully or not.

Example: Tracing All Instructions

To create a trace trap for all instructions, whether they run successfully or not, enter:

```
trace instruction
```

Examples of Using Trace Trap Common Options

Trace trap common options are options that apply to all trace trap operands. They let you specify the conditions under which CP traces events. Some of these options you may use all the time; others you may never use. With this in mind, use this section to learn what each operand does and when you may want to use it. This section also describes how to use each common option with the TRACE INSTRUCTION operand.

Example: Naming a Trace Trap (ID ident)

Use ID *ident* to assign a 1- to 4-character name to a trap. This name helps you identify a trap if you later decide to change one or more of its options. If you do not specify a name for a trap, CP assigns it a unique number from 1 to 9999.

Note: Do not name a trap ALL.

For example, if you want to name a trap *BOB,* enter:

```
trace instruction id bob
```

Example: Specifying a Storage Range for Traps (PSWA Range/FROM Range/ RANGE Range)

Use PSWA, FROM, and RANGE to trace events only when the instruction counter (PSWA) points to an instruction in the range you specify. (PSWA, FROM, and RANGE perform identical functions.) Specify the range as a single address, a pair of addresses separated by a (-), or an address followed by a period followed by a byte count.

For example, to trace instructions that occur between storage locations 20000 and 30000, enter:

```
trace instruction pswa 20000-30000
```

or

```
trace instruction from 20000-30000
```

or

```
trace instruction range 20000-30000
```

Example: Modifying a Storage Range for Traps

Use the PRI, SECO, HOME, ASTE*raddr*, STD*hexword*, STO*raddr*, ASN*asn*, AREG*areg*, ALET*hexword*, and ALET*hexword*.AL*raddr* options to specify the guest address space to be traced by a trace trap.

For example, to trace instructions that occur in the guests's current primary address space between storage locations 20000 and 30000, enter:

```
trace instruction pswa pri 20000-30000
```

or

```
trace instruction from pri 20000-30000
```

or

```
trace instruction range pri 20000-30000
```

Note: For more information on these options, see the “TRACE: Common Options,” in *z/VM: CP Commands and Utilities Reference*.

Example: Specifying Whether Your Virtual Machine Is to Stop after a Trap (RUN/NORUN)

Use the RUN option to specify whether CP stops your virtual machine when it traps an event. Use the NORUN option only to override an existing trap for which you specified RUN.

For example, if you want to trace all instructions, but you do not want CP to stop your virtual machine after each one, enter:

```
trace instruction run
```

Note: You cannot use the RUN option with the STEP or STOP options.

Example: Specifying How Frequently You Want to Trace an Event (SKIP n/ PASS n)

Use SKIP *n* if you want CP to ignore the associated event the *first n* times it occurs. Use PASS *n* if you want CP to ignore the associated event except for *every n*th time it occurs.

When used with the NOSIM option, SKIP *n* and PASS *n* specify the number of times CP simulates an instruction.

For example, if you want to trace all instructions after the seventh instruction, enter:

```
trace instruction skip 7
```

If you want to trace only every eighth instruction, enter:

```
trace instruction pass 7
```

If you want to trace all instructions after the 50th instruction and then every 8th instruction, enter:

```
trace instruction skip 50 pass 7
```

Example: Specifying How Frequently Your Virtual Machine Is to Stop (STOP n/STEP n)

Unless you specify otherwise, CP stops your virtual machine every time it traps an event. If you use the RUN option, CP does not stop your virtual machine when it traps events.

Use STOP *n* if you want CP to stop your virtual machine only after the *first n* events. Use STEP *n* if you want CP to stop your virtual machine only after *every n* events.

For example, if you want CP to stop your virtual machine only after it has trapped seven instructions, enter:

```
trace instruction stop 7
```

If you want CP to stop your virtual machine after every seven instructions it traps, enter:

```
trace instruction step 7
```

If you want CP to stop your virtual machine after it has traced 20 instructions and then after every seventh instruction, enter:

```
trace instruction stop 20 step 7
```

Example: Specifying Where CP Sends Trace Output (PRINTER/NOPRINTER, TERMINAL/NOTERMINAL, BOTH)

Unless you specify otherwise, CP sends information about events it traps to your virtual machine operator's console. Use the PRINTER and NOTERMINAL options to change where CP sends information about events. Use NOPRINTER and TERMINAL only if you need to override existing options. Use the BOTH option to send information about events to your virtual console and your virtual printer.

For example, if you want CP to send trace output about instructions it traps to your virtual machine operator's console *and* your virtual printer, enter:

```
trace instruction terminal printer
```

or

```
trace instruction both
```

If you want CP to send trace output to your virtual printer but not to your virtual machine operator's console, enter:

```
trace instruction noterminal printer
```

Example: Limiting Traps to Supervisor or Problem State (SUPERVISOR/ PROBLEM)

Use the SUPERVISOR and PROBLEM options if you want to trace events only while your virtual machine is in supervisor or problem state.

For example, if you want CP to trace instructions that occur only while your virtual machine is in problem state, enter:

```
trace instruction problem
```

If you want CP to trace instructions that occur only while your virtual machine is in supervisor state, enter:

```
trace instruction supervisor
```

Example: Limiting Tracing to When DAT Is On or Off (DAT/NODAT)

Use the DAT and NODAT options if you want to limit tracing to events occurring while your virtual machine has dynamic address translation on or off.

For example, if you want CP to trace instructions that occur only while your virtual machine has dynamic address translation on, enter:

```
trace instruction dat
```

If you want CP to trace instructions that occur only while your virtual machine has dynamic address translation off, enter:

```
trace instruction nodat
```

Example: Entering CP Commands When You Trace Events (CMD/NOCMD)

Unless you specify otherwise, CP stops your virtual machine after every event it traps. When CP stops your virtual machine, it displays

```
CP READ
```

at the bottom right of your display screen. You can then enter any CP commands. To restart the program you are tracing, enter BEGIN.

Use the CMD option if there is a CP command or commands that you want to run every time CP traps an event. Use NOCMD to delete CMD fields from existing trace traps.

Note: If you use the CMD option, it *must* be the last option you specify.

For example, if you want CP to display the contents of storage location 20 every time it traps an instruction, enter:

```
trace instruction cmd display 20
```

Or, if you want CP to display the contents of your general registers every time it traps an instruction, enter:

```
trace instruction cmd display g
```

Example: Limiting Tracing to Certain Instructions or a Sequence of Instructions (DATA)

If you want CP to trace all LR 3,15 instructions, enter:

```
trace instr data 183F
```

If you want CP to trace all LR 3,15 instructions only if they are followed by a STORE (ST) instruction, enter:

```
trace instr data 183F50
```

Example: Displaying the Current Value of the Trace Count or Initiating the Trace Count (COUNT)

Use the TRACE COUNT function to count the number of successful trace events in the current trace set. While TRACE COUNT is active, no other trace output is produced. Your virtual machine does not stop after each event, even if NORUN was specified.

When a valid TRACE command is entered, the value of the current count is displayed, and TRACE COUNT terminates.

To display the current value of the trace count, enter:

```
trace count
```

If TRACE COUNT is entered when COUNT is already active, then the current count is displayed, and the count resumes from 0.

Example: Displaying a List of Up to Six of the Most Recent Branch Instructions Run (TABLE)

To display the most recent branch instructions run, enter:

```
trace table
```

The most recent branch is displayed as the last entry in the table.

Example: Merging Traps from the Named Trace Set with the Current Trace Set (APPEND)

Use TRACE APPEND to merge the trace traps from two trace sets into one trace set.

For example, if you want CP to merge the trace traps from the trace set NINA with the current trace set, enter:

```
trace append nina
```

Using TRACE AIF

Use TRACE AIF to trace interruptions generated by the Adapter Interruption Facility.

Example: Tracing

Entering:

```
cp trace aif run
```

establishes the trace, with the ***run*** option so execution will continue automatically. The first line below represents an AIF event that was reflected to this virtual machine. Execution begins at 00BE2368 (the I/O NEW PSW). The rest of the output is the result of an earlier trace trap (cp trace i r BE2368.20).

```
*** 00BE2242    ADAPTER INT  -> 00BE2368    ISC 0   SCH TYPE 0
 -> 00BE2368  STM   90BC0000 >> 00000000    CC 0
    00BE236C  BALR  05C0        CC 0
    00BE236E  STM   900FC0BA >> 00BE2428    CC 0
    00BE2372  MVC   D207C0E60000 >> 00BE2454    00000000    CC 0
    00BE2378  L     58E0C24E    00BE25BC    CC 0
    00BE237C  LA    41E0E001  = 000000CB    CC 0
    00BE2380  ST    50E0C24E >> 00BE25BC    CC 0
    00BE2384  MVC   D207C0FA0038 >> 00BE2468    00000038    CC 0
```

Using TRACE ALL

Use TRACE ALL to trace all instructions, as well as all program, external, and I/O interruptions.

Example: Tracing All Program, External, and I/O Interruptions and All Instructions

To create a trace trap for all instructions, as well as all program, external, and I/O interruptions, enter:

```
trace all
```

Note: You can also use any of the trace trap common options described in "Examples of Using Trace Trap Common Options" on page 126 to modify the conditions under which CP traces events. A list of these common options are in Appendix A, "Trace Trap Common Options," on page 143.

Using TRACE BRANCH

Use TRACE BRANCH to trace successful branch instructions, including LPSW and SVC instructions that do not load a wait PSW.

Example: Tracing All Branch Instructions

To trace all branch instructions, enter:

```
trace branch
```

Example: Tracing Branches into Specific Storage Locations

To trace all branch instructions that branch to a specific storage location (for example, storage locations 10000 to 11000), enter:

```
trace branch into 10000-11000
```

Note: You can also use any of the trace trap common options described in "Examples of Using Trace Trap Common Options" on page 126 to modify the conditions under which CP traces branch instructions. A list of these common options is in Appendix A, "Trace Trap Common Options," on page 143.

The following examples show two ways you can use the FROM or PSWA options. For examples describing how to use other common options, see "Examples of Using Trace Trap Common Options" on page 126.

Example: Specifying a Storage Range for Traps

To trace all branch instructions that occur between storage locations 24000 and 30000, enter:

```
trace branch from 24000-30000
```

To trace all branch instructions that occur between storage locations 24000 and 30000, *and* branch into a storage location between 10000 and 11000, enter:

```
trace branch into 10000-11000 from 24000-30000
```

Using TRACE STORE

Use TRACE STORE to trace alterations to virtual machine storage.

Note: A TRACE STORE may not catch a storage alteration caused by:

- All DIAGNOSE instructions
- All Cryptographic facility instructions
- Asynchronous I/O instructions
- IUCV/VMCF/APPC
- COMPRESSION CALL (CMPSC)
- MOVE LONG (MVCL)
- MOVE LONG EXTENDED (MVCLE)
- MOVE STRING INSTRUCTION (MVST).
- Convert Unicode to UTF-8 (UUTF)
- Convert UTF-8 to Unicode (CUTFU)
- Translate Extended (TRE)

Example: Tracing All Changes to Storage

To trace all changes to storage, enter:

```
trace store
```

Example: Tracing All Changes to Specific Storage Locations

To trace all changes to storage location 15000, enter:

```
trace store into 15000
```

To trace all changes to storage locations 15000 to 20000, enter:

```
trace store into 15000-20000
```

Example: Tracing Specific Changes to Storage

To trace a specific change to storage (for example, every time the value X'FFFF' is stored into storage location 15000-15004), enter:

```
trace store into 15000-15004 data FFFF
```

Note: You can also use any of the trace trap common options described in "Examples of Using Trace Trap Common Options" on page 126 to modify the conditions under which CP traces events. A list of these common options is in Appendix A, "Trace Trap Common Options," on page 143.

Using TRACE GPR

Use TRACE GPR to trace alterations to your virtual machine's general-purpose registers. TRACE G will trap alterations to the rightmost four bytes of a register. TRACE GG will trap alterations to four-or eight- byte registers.

Example: Tracing Alterations to All General Registers

To trace all alterations to all general-purpose registers, enter:

```
trace gg
```

Example: Tracing Alterations to a Range of General Registers

To trace all alterations to general-purpose registers 11, 12, 13, 14, 15, 0, 1, and 2, enter:

```
trace gg11-2
```

Example: Tracing Alterations to a Specific General Register

To trace all alterations to general-purpose register 4, enter:

```
trace gg4
```

Example: Tracing Specific Alterations to General Registers

To trace all occurrences of the value 0 being stored into general-purpose registers 0–5, enter:

```
trace gg0-5 data 0
```

To trace every time a value between 0 and 2000 is stored into general-purpose registers 0–5, enter:

```
trace gg0-5 data 0-2000
```

Note: You can also use any of the trace trap common options described in "Examples of Using Trace Trap Common Options" on page 126 to modify the conditions under which CP traces events. A list of these common options is in Appendix A, "Trace Trap Common Options," on page 143.

Using TRACE AR

Use TRACE AR to trace alterations to your virtual machine's access registers.

Example: Tracing Alterations to All Access Registers

To trace all alterations to all access registers, enter:

```
trace ar
```

Example: Tracing Alterations to a Range of Access Registers

To trace all alterations to access registers 10, 11, 12, 13, 14, 15, 0, 1, and 2, enter:

```
trace ar10-2
```

Example: Tracing Alterations to a Specific Access Register

To trace all alterations to access register 4, enter:

```
trace ar4
```

Example: Tracing Specific Alterations to Access Registers

To trace all occurrences of the value 0 being stored into access registers 0–5, enter:

```
trace ar0-5 data 0
```

To trace every time a value between 0 and 2000 is stored into access registers 0–5, enter:

```
trace ar0-5 data 0-2000
```

Note: You can also use any of the trace trap common options described in "Examples of Using Trace Trap Common Options" on page 126 to modify the conditions under which CP traces events. A list of these common options is in Appendix A, "Trace Trap Common Options," on page 143.

Using TRACE SVC, DIAGNOSE, MC

Use TRACE SVC, DIAGNOSE, MC to trace supervisor calls (SVC), DIAGNOSE instructions, or monitor calls (MC).

Note: The following examples show how to create traps for DIAGNOSE codes. To create traps for SVC instructions, substitute SVC for DIAGNOSE. To create traps for monitor calls, substitute MC for DIAGNOSE.

Example: Tracing the Execution of All DIAGNOSE Codes

To create a trap for all DIAGNOSE codes, enter:

```
trace diagnose
```

Example: Tracing a Hexadecimal Range of DIAGNOSE Codes

To create a trap for a hexadecimal range of DIAGNOSE codes (for example, DIAGNOSE X'08' to DIAGNOSE X'4C'), enter:

```
trace diagnose 8-4c
```

Example: Tracing Specific DIAGNOSE Codes

To create a trap for a specific DIAGNOSE instruction (for example, DIAGNOSE X'0C'), enter:

```
trace diagnose c
```

Example: Tracing DIAGNOSE Codes Using the NOSIM Option

Your virtual machine does not run DIAGNOSE codes; instead, when your virtual machine issues a DIAGNOSE code, CP intercepts it and performs the requested function on behalf of your virtual machine.

Use the NOSIM option if you want CP to trap DIAGNOSE instructions but not perform the function that your virtual machine requested.

Unless you specify NOSIM, CP automatically simulates DIAGNOSE instructions that your virtual machine issues. Use the SIM option only to override a trap you created using the NOSIM option.

To create a trap where CP traps all DIAGNOSE instructions but does not simulate them, enter:

```
trace diagnose nosim
```

To create a trap where CP traps a range of DIAGNOSE instructions (for example, DIAGNOSE X'08' to DIAGNOSE X'4C') but does not simulate them, enter:

```
trace diagnose 08-4c nosim
```

To create a trap where CP traps a specific DIAGNOSE instruction (for example, DIAGNOSE X'0C') but does not simulate it, enter:

```
trace diagnose 0c nosim
```

Note: You can also use any of the trace trap common options described in "Examples of Using Trace Trap Common Options" on page 126 to modify the conditions under which CP traces events. A list of these common options is in Appendix A, "Trace Trap Common Options," on page 143.

Using TRACE I/O

Use TRACE I/O to trace I/O instructions or I/O interrupts (or both) to a specific I/O device, a range of devices, or all I/O devices.

Example: Tracing All I/O Instructions and Interrupts

To create a trap for all I/O instructions and interrupts to all I/O devices, enter:

```
trace i/o
```

If you want this trap to display CCWs when an RIO, SIO, SIOF, SSCH, or RSCH instruction successfully initiates I/O, enter:

```
trace i/o ccw
```

To trace all I/O instructions and interrupts for a range of devices (for example, virtual device numbers 190 to 19A), enter:

```
trace i/o 190-19a
```

If you want this trap to display CCWs when an RIO, SIO, SIOF, SSCH, or RSCH instruction successfully initiates I/O, enter:

```
trace i/o 190-19a ccw
```

To trace all I/O instructions and interrupts for a specific device (for example, virtual device number 009), enter:

```
trace i/o 009
```

If you want this trap to display CCWs when an RIO, SIO, SIOF, SSCH, or RSCH instruction successfully initiates I/O, enter:

```
trace i/o 009 ccw
```

Example: Tracing I/O Instructions (but not I/O Interrupts)

To trace all I/O instructions (but not I/O interrupts) to all I/O devices, enter:

```
trace i/o instruction
```

If you want this trap to display CCWs when an RIO, SIO, SIOF, SSCH, or RSCH instruction successfully initiates I/O, enter:

```
trace i/o instruction ccw
```

If you want data displayed for each CCW handled by the Virtual Channel Simulator, enter:

```
trace i/o inst int ccw iodata nnnn
```

where *nnnn* is any number from 0 - 3800 (with a default of 64) that specifies the amount of data to be displayed for each CCW.

To trace all I/O instructions (but not I/O interrupts) for a range of devices (for example, virtual device numbers 190 to 19A), enter:

```
trace i/o 190-19a instruction
```

If you want this trap to display CCWs when an RIO, SIO, SIOF, SSCH, or RSCH instruction successfully initiates I/O, enter:

```
trace i/o 190-19a instruction ccw
```

To trace all I/O instructions (but not I/O interrupts) for a specific virtual device (for example, virtual device number 009), enter:

```
trace i/o 009 instruction
```

If you want this trap to display CCWs when an RIO, SIO, SIOF, SSCH, or RSCH instruction successfully initiates I/O, enter:

```
trace i/o 009 instruction ccw
```

Example: Tracing I/O Interrupts (but not I/O Instructions)

To trace all I/O interrupts (but not I/O instructions) to all I/O devices, enter:

```
trace i/o interruption
```

To trace all I/O interrupts (but not I/O instructions) for a range of devices (for example, virtual device numbers 190 to 19A), enter:

```
trace i/o 190-19a interruption
```

To trace all I/O interrupts (but not I/O instructions) for a specific virtual device (for example, virtual device number 009), enter:

```
trace i/o 190-19a interruption
```

Example: Tracing I/O Instructions Using the NOSIM Option

Your virtual machine cannot perform I/O instructions; instead, when your virtual machine issues an I/O instruction, CP must simulate it for your virtual machine. Use the NOSIM option if you want CP to trap I/O instructions but not simulate them for your virtual machine.

Unless you specify NOSIM, CP automatically simulates I/O instructions that your virtual machine issues. Use the SIM option only to override a trap you created using the NOSIM option.

To create a trap where CP traps all I/O instructions but does not simulate them, enter:

```
trace i/o instruction nosim
```

To create a trap where CP traps all I/O instructions for a range of devices (for example, devices 190 to 19A) but does not simulate them, enter:

```
trace i/o 190-19a instruction nosim
```

To create a trap where CP traps all I/O instructions for a specific device (for example, device 009) but does not simulate them, enter:

```
trace i/o 009 instruction nosim
```

Note: You can also use any of the trace trap common options described in "Examples of Using Trace Trap Common Options" on page 126 to modify the conditions under which CP traces events. A list of these common options is in Appendix A, "Trace Trap Common Options," on page 143.

Using TRACE SIO, SIOF, SSCH, RIO, RSCH

Use TRACE SIO, SIOF, SSCH, RIO, or RSCH to trace the SIO (START I/O), SIOF (START I/O FAST RELEASE), SSCH (START SUBCHANNEL), RIO (RESUME I/O), or RSCH (RESUME SUBCHANNEL) instructions for all input/output (I/O) devices or for a range of devices. Tracing is limited to instructions that do not produce an operation or privileged operation exception.

Note: The following examples show how to create traps for the SIO instruction. To create traps for any of the other instructions listed above, substitute the instruction you want to trace for SIO.

Example: Tracing All SIO Instructions

To create a trap for all SIO instructions issued to all devices, enter:

```
trace sio
```

Example: Tracing SIO Instructions Issued to a Range of Devices

To create a trap for all SIO instructions issued to a range of devices (for example, virtual device numbers 009 to 00A), enter:

```
trace sio 009-00a
```

Example: Tracing SIO Instructions Issued to a Specific Device

To create a trap for all SIO instructions issued to a specific device (for example, virtual device number 299), enter:

```
trace sio 299
```

Example: Tracing SIO Instructions Using the NOSIM Option

Your virtual machine cannot perform SIO instructions; instead, when your virtual machine issues an SIO instruction, CP must simulate it for your virtual machine. Use the NOSIM option if you want CP to trap SIO instructions but not simulate them for your virtual machine.

Unless you specify NOSIM, CP automatically simulates SIO instructions that your virtual machine issues. Use the SIM option only to override a trap you created using the NOSIM option.

To create a trap where CP traps all SIO instructions but does not simulate them, enter:

```
trace sio nosim
```

To create a trap where CP traps all SIO instructions for a range of devices (for example, devices 009 to 00A) but does not simulate them, enter:

```
trace sio 009-00a nosim
```

To create a trap where CP traps all SIO instructions for a specific device (for example, device 299) but does not simulate them, enter:

```
trace sio 299 nosim
```

Note: You can also use any of the trace trap common options described in "Examples of Using Trace Trap Common Options" on page 126 to modify the conditions under which CP traces events. A list of these common options is in Appendix A, "Trace Trap Common Options," on page 143.

Using TRACE TPI

Use TRACE TPI to trace Test-Pending-Interrupt instructions. TRACE TPI is accepted for any virtual machine mode, however, tracing is limited to those instructions that do not produce an operation or privileged-operation exception.

Example: Tracing All TPI Instructions

To create a trap for all TPI instructions issued to all devices, enter:

```
trace tpi
```

Example: Tracing TPI Instructions Issued to a Range of Devices

To create a trap for all TPI instructions issued to a range of devices (for example, virtual device numbers 009 to 00A), enter:

```
trace tpi 009-00a
```

Example: Tracing TPI Instructions Issued to a Specific Device

To create a trap for all TPI instructions issued to a specific device (for example, virtual device number 299), enter:

```
trace tpi 299
```

Example: Tracing TPI Instructions Using the NOSIM Option

Your virtual machine cannot perform TPI instructions; instead, when your virtual machine issues a TPI instruction, CP must simulate it for your virtual machine. Use the NOSIM option if you want CP to trap TPI instructions but not simulate them for your virtual machine.

Unless you specify NOSIM, CP automatically simulates TPI instructions that your virtual machine issues. Use the SIM option only to override a trap you created using the NOSIM option.

To create a trap where CP traps all TPI instructions but does not simulate them, enter:

```
trace tpi nosim
```

To create a trap where CP traps all TPI instructions for a range of devices (for example, devices 009 to 00A) but does not simulate them, enter:

```
trace tpi 009-00a nosim
```

To create a trap where CP traps all TPI instructions for a specific device (for example, device 299) but does not simulate them, enter:

```
trace tpi 299 nosim
```

Note: You can also use any of the trace trap common options described in "Examples of Using Trace Trap Common Options" on page 126 to modify the conditions under which CP traces events. A list of these common options is in Appendix A, "Trace Trap Common Options," on page 143.

Using TRACE MNEMONIC1

Use TRACE MNEMONIC1 to trace the execution of the following assembler language instructions. CP limits tracing to instructions that do not produce an operation or privileged operation exception.

```
BSA          PGOUT        SSK
BSG          PR           SSM
CONCS        PT           STAP
CSP          PTI          STCPS
DISCS        PTLB         STCRW
EPSW         RCHP         STCTG
ISK          RDD          STCTL
IUCV         RDP          STFL
LASP         RRB          STIDC
LCTL         RRBE         STIDP
LCTLG        SAL          STNSM
LPSW         SCHM         STOSM
LPSWE        SCK          STPX
LPSWEY       SERVC        STSI
PALB         SIE          TB
PC           SIGP         TPROT
PGIN         SPX          WRD
```

Note: The following examples show how to create traps for the mnemonic LPSW. To create traps for any of the other mnemonics listed above, substitute the mnemonic you want to trap for LPSW.

Example: Tracing All LPSW Instructions

To create a trap for all LPSW instructions, enter:

```
trace lpsw
```

Example: Tracing LPSW Instructions Using the NOSIM Option

Your virtual machine cannot perform LPSW instructions; instead, when your virtual machine issues a LPSW instruction, CP must simulate it for your virtual machine. Use the NOSIM option if you want CP to trap LPSW instructions but not simulate them for your virtual machine.

Unless you specify NOSIM, CP automatically simulates LPSW instructions that your virtual machine issues. Use the SIM option only to override a trap you created using the NOSIM option.

To create a trap where CP traps all LPSW instructions but does not simulate them, enter:

```
trace lpsw nosim
```

Note: You can also use any of the trace trap common options described in "Examples of Using Trace Trap Common Options" on page 126 to modify the conditions under which CP traces events. A list of these common options is in Appendix A, "Trace Trap Common Options," on page 143.

Using TRACE MNEMONIC2

Use TRACE MNEMONIC2 to trace the execution of the following I/O instructions:

```
CLRCH     HIO      TCH
CLRIO     HSCH     TIO
CSCH      MSCH     TSCH
HDV                STSCH
```

Note: The following examples show how to create traps for the mnemonic CLRIO. To create traps for any of the other mnemonics listed above, substitute the mnemonic you want to trace for CLRIO.

Example: Tracing All CLRIO Instructions

To create a trap for all CLRIO instructions issued to all devices, enter:

```
trace clrio
```

Example: Tracing CLRIO Instructions Issued to a Range of Devices

To create a trap for all CLRIO instructions issued to a range of devices (for example, virtual device numbers 009 to 00A), enter:

```
trace clrio 009-00a
```

Example: Tracing CLRIO Instructions Issued to a Specific Device

To create a trap for all CLRIO instructions issued to a specific device (for example, virtual device number 299), enter:

```
trace clrio 299
```

Example: Tracing CLRIO Instructions Using the NOSIM Option

Your virtual machine cannot perform CLRIO instructions; instead, when your virtual machine issues a CLRIO instruction, CP must simulate it for your virtual machine. Use the NOSIM option if you want CP to trap CLRIO instructions but not simulate them for your virtual machine.

Unless you specify NOSIM, CP automatically simulates CLRIO instructions that your virtual machine issues. Use the SIM option only to override a trap you created using the NOSIM option.

To create a trap where CP traps all CLRIO instructions but does not simulate them, enter:

```
trace clrio nosim
```

To create a trap where CP traps all CLRIO instructions for a range of devices (for example, devices 009 to 00A) but does not simulate them, enter:

```
trace clrio 009-00a nosim
```

To create a trap where CP traps all CLRIO instructions for a specific device (for example, device 299) but does not simulate them, enter:

```
trace clrio 299 nosim
```

Note: You can also use any of the trace trap common options described in "Examples of Using Trace Trap Common Options" on page 126 to modify the conditions under which CP traces events. A list of these common options is in Appendix A, "Trace Trap Common Options," on page 143.

Using TRACE EXTERNAL, PROGRAM

Use TRACE EXTERNAL, PROGRAM to trace external or program interrupts, or a range of external or program interrupts. When possible, CP also displays the instruction that causes program interrupts.

Example: Tracing External Interruptions

To create a trap for all external interrupts, enter:

```
trace external
```

To create a trap for a range of external interruptions, enter:

```
trace external xx-yy
```

Where *xx-yy* are the first and last values, in hexadecimal, of the external interrupt you want to trace. To create a trap for a specific external interrupt, enter:

```
trace external zz
```

where *zz* is the value, in hexadecimal, of the external interrupt you want to trace.

Example: Tracing Program Interrupts

To create a trap for all program interrupts, enter:

```
trace program
```

To create a trap for a range of program interrupts, enter:

```
trace program xx-yy
```

where *xx-yy* are the first and last values, in hexadecimal, of the program interrupts you want to trace.

To create a trap for a specific program interrupt, enter:

```
trace program zz
```

where *zz* is the value, in hexadecimal, of the program interrupt you want to trace.

Note: You can also use any of the trace trap common options described in "Examples of Using Trace Trap Common Options" on page 126 to modify the conditions under which CP traces events. A list of these common options is in Appendix A, "Trace Trap Common Options," on page 143.

Using TRACE MCH

Use TRACE MCH to trace machine-check interrupts.

Example: Tracing Machine-Check Interrupts

To create a trace trap for all machine-check interrupts, enter:

```
trace mch
```

Note: You can also use any of the trace trap common options described in "Examples of Using Trace Trap Common Options" on page 126 to modify the conditions under which CP traces events. A list of these common options is in Appendix A, "Trace Trap Common Options," on page 143.

Appendix A. Trace Trap Common Options

The following list contains the trace trap common options that apply to all the TRACE command operands (shown under "Events That You Can Trace" on page 125):

PSWA *range*/FROM *range*/RANGE *range*
PRI
SECO
HOME
ASTE*raddr*
STD*hexword*
STO*raddr*
ASN*asn*
AREG*areg*
ALET*hexword*
ALET*hexword*.AL*raddr*
ASCE*hexdblword*
SUPERVISOR/PROBLEM
RUN/NORUN
DAT/NODAT
SKIP *n*/PASS *n*
CMD/NOCMD
STOP *n*/STEP *n*
TERMINAL/NOTERMINAL
PRINTER/NOPRINTER
BOTH
IDENTIFIER

Notices

Trademarks

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see:

- The section entitled **IBM Websites** at IBM Privacy Statement (https://www.ibm.com/privacy)
- Cookies and Similar Technologies (https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see *z/VM: General Information*.

Where to Get z/VM Information

The current z/VM product documentation is available in IBM Documentation - z/VM (https://www.ibm.com/docs/en/zvm).

z/VM Base Library

Overview

- *z/VM: License Information*, GI13-4377
- *z/VM: General Information*, GC24-6286

Installation, Migration, and Service

- *z/VM: Installation Guide*, GC24-6292
- *z/VM: Migration Guide*, GC24-6294
- *z/VM: Service Guide*, GC24-6325
- *z/VM: VMSES/E Introduction and Reference*, GC24-6336

Planning and Administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6261
- *z/VM: CMS Planning and Administration*, SC24-6264
- *z/VM: Connectivity*, SC24-6267
- *z/VM: CP Planning and Administration*, SC24-6271
- *z/VM: Getting Started with Linux on IBM Z*, SC24-6287
- *z/VM: Group Control System*, SC24-6289
- *z/VM: I/O Configuration*, SC24-6291
- *z/VM: Running Guest Operating Systems*, SC24-6321
- *z/VM: Saved Segments Planning and Administration*, SC24-6322
- *z/VM: Secure Configuration Guide*, SC24-6323

Customization and Tuning

- *z/VM: CP Exit Customization*, SC24-6269
- *z/VM: Performance*, SC24-6301

Operation and Use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6260
- *z/VM: CMS Primer*, SC24-6265
- *z/VM: CMS User's Guide*, SC24-6266
- *z/VM: CP Commands and Utilities Reference*, SC24-6268

- *z/VM: System Operation*, SC24-6326
- *z/VM: Virtual Machine Operation*, SC24-6334
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6337
- *z/VM: XEDIT User's Guide*, SC24-6338

Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6256
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6257
- *z/VM: CMS Application Multitasking*, SC24-6258
- *z/VM: CMS Callable Services Reference*, SC24-6259
- *z/VM: CMS Macros and Functions Reference*, SC24-6262
- *z/VM: CMS Pipelines User's Guide and Reference*, SC24-6252
- *z/VM: CP Programming Services*, SC24-6272
- *z/VM: CPI Communications User's Guide*, SC24-6273
- *z/VM: ESA/XC Principles of Operation*, SC24-6285
- *z/VM: Language Environment User's Guide*, SC24-6293
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6295
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6296
- *z/VM: OpenExtensions Commands Reference*, SC24-6297
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6298
- *z/VM: OpenExtensions User's Guide*, SC24-6299
- *z/VM: Program Management Binder for CMS*, SC24-6304
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6313
- *z/VM: REXX/VM Reference*, SC24-6314
- *z/VM: REXX/VM User's Guide*, SC24-6315
- *z/VM: Systems Management Application Programming*, SC24-6327
- *z/VM: z/Architecture Extended Configuration (z/XC) Principles of Operation*, SC27-4940

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6255
- *z/VM: CP Messages and Codes*, GC24-6270
- *z/VM: Diagnosis Guide*, GC24-6280
- *z/VM: Dump Viewing Facility*, GC24-6284
- *z/VM: Other Components Messages and Codes*, GC24-6300
- *z/VM: VM Dump Tool*, GC24-6335

z/VM Facilities and Features

Data Facility Storage Management Subsystem for z/VM

- *z/VM: DFSMS/VM Customization*, SC24-6274
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6275
- *z/VM: DFSMS/VM Messages and Codes*, GC24-6276
- *z/VM: DFSMS/VM Planning Guide*, SC24-6277

- *z/VM: DFSMS/VM Removable Media Services*, SC24-6278
- *z/VM: DFSMS/VM Storage Administration*, SC24-6279

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6281
- *z/VM: Directory Maintenance Facility Messages*, GC24-6282
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6283

Open Systems Adapter

- Open Systems Adapter/Support Facility on the Hardware Management Console (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC14-7580-02.pdf), SC14-7580
- Open Systems Adapter-Express ICC 3215 Support (https://www.ibm.com/docs/en/zos/2.3.0?topic=osa-icc-3215-support), SA23-2247
- Open Systems Adapter Integrated Console Controller User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC27-9003-02.pdf), SC27-9003
- Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/ioa2z1f0.pdf), SA22-7935

Performance Toolkit for z/VM

- *z/VM: Performance Toolkit Guide*, SC24-6302
- *z/VM: Performance Toolkit Reference*, SC24-6303

The following publications contain sections that provide information about z/VM Performance Data Pump, which is licensed with Performance Toolkit for z/VM.

- *z/VM: Performance*, SC24-6301. See z/VM Performance Data Pump.
- *z/VM: Other Components Messages and Codes*, GC24-6300. See Data Pump Messages.

RACF® Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6305
- *z/VM: RACF Security Server Command Language Reference*, SC24-6306
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6307
- *z/VM: RACF Security Server General User's Guide*, SC24-6308
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6309
- *z/VM: RACF Security Server Messages and Codes*, GC24-6310
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6311
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6312
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6316
- *z/VM: RSCS Networking Exit Customization*, SC24-6317
- *z/VM: RSCS Networking Messages and Codes*, GC24-6318
- *z/VM: RSCS Networking Operation and Use*, SC24-6319
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6320

TCP/IP for z/VM

- *z/VM: TCP/IP Diagnosis Guide*, GC24-6328
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6329
- *z/VM: TCP/IP Messages and Codes*, GC24-6330
- *z/VM: TCP/IP Planning and Customization*, SC24-6331
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *z/VM: TCP/IP User's Guide*, SC24-6333

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ickug00_v2r5.pdf), GC35-0033

Related Products

XL C++ for z/VM

- *XL C/C++ for z/VM: Runtime Library Reference*, SC09-7624
- *XL C/C++ for z/VM: User's Guide*, SC09-7625

z/OS

IBM Documentation - z/OS (https://www.ibm.com/docs/en/zos)

Index

Product Number: 5741-A09

Printed in USA

SC24-6334-74