z/VM
7.3

CP Planning and Administration

IBM

SC24-6271-73

Note:

Before you use this information and the product it supports, read the information in "Notices" on page 853.

This edition applies to version 7, release 3 of IBM® z/VM® (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2025-06-17

Contents

Figures

Tables

About This Document

This document tells you how to plan and administer the IBM z/VM Control Program (CP). It describes the following tasks:

- System configuration planning and administration
- User planning and administration
- Storage planning and administration

Note: For information about planning your hardware and software I/O configuration, see *z/VM: I/O Configuration*. For information about planning and using saved segments, see *z/VM: Saved Segments Planning and Administration*.

Planning and administration are tasks that overlap. Before you install z/VM, you need to plan:

- What the processor and device configuration will be
- What z/VM functions the system will use
- Which guest operating systems will be used
- How much storage will be required
- What sort of user environments will be available.

During system generation, you will set the proper parameters in the system to implement your plan. Later, after the system is running, you will want to change some of these parameters as conditions change; for example, when new users are allowed access to the system and new devices are brought online. You will also want to monitor the performance of the system and perhaps change tuning parameters to make it run more efficiently.

A good example of the overlap between planning and administration is the user directory. The user directory is a file that identifies each user on the system and contains directory statements that define the environment each user works under. Initially, you will set up this directory for the known users of the system. Later, as new users are given access to the system and others are removed, you will have to update the directory.

Some information provided here is based on the experiences of IBM customers. The recommendations in this document are meant to help installations run operating systems efficiently under z/VM.

Intended Audience

This document is for anyone responsible for planning, installing, and updating a z/VM system.

The reader is expected to have a general understanding of data processing and teleprocessing techniques. This document assumes you have thought about:

- What z/VM functions your site requires.
- What connections you need to other sites and the implications for coordination.
- What your hardware and physical requirements are and the implications for coordination.
- Which guest operating systems you will be running.
- How many users you are going to have and under what sort of environment they will be running their applications.

Syntax, Message, and Response Conventions

The following topics provide information on the conventions used in syntax diagrams and in examples of messages and responses.

How to Read Syntax Diagrams

Special diagrams (often called *railroad tracks*) are used to show the syntax of external interfaces.

To read a syntax diagram, follow the path of the line. Read from left to right and top to bottom.

- The ►►—— symbol indicates the beginning of the syntax diagram.
- The ——► symbol, at the end of a line, indicates that the syntax diagram is continued on the next line.
- The ►—— symbol, at the beginning of a line, indicates that the syntax diagram is continued from the previous line.
- The ——►◄ symbol indicates the end of the syntax diagram.

Within the syntax diagram, items on the line are required, items below the line are optional, and items above the line are defaults. See the examples in Table 1 on page xxvi.

Table 1. Examples of Syntax Diagram Conventions

Syntax Diagram Convention	**Example**
Keywords and Constants A keyword or constant appears in uppercase letters. In this example, you must specify the item KEYWORD as shown. In most cases, you can specify a keyword or constant in uppercase letters, lowercase letters, or any combination. However, some applications may have additional conventions for using all-uppercase or all-lowercase.	►►— KEYWORD —►◄
Abbreviations Uppercase letters denote the shortest acceptable abbreviation of an item, and lowercase letters denote the part that can be omitted. If an item appears entirely in uppercase letters, it cannot be abbreviated. In this example, you can specify KEYWO, KEYWOR, or KEYWORD.	►►— KEYWOrd —►◄
Symbols You must specify these symbols exactly as they appear in the syntax diagram.	***** Asterisk **:** Colon **,** Comma **=** Equal Sign **-** Hyphen **()** Parentheses **.** Period

Table 1. Examples of Syntax Diagram Conventions (continued)

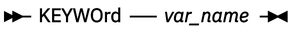
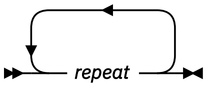
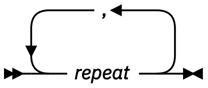
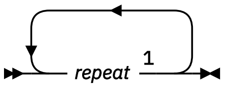
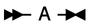
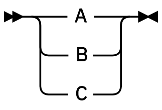
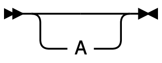
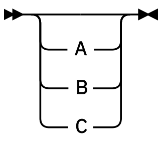
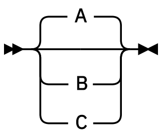
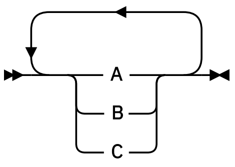
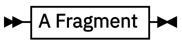
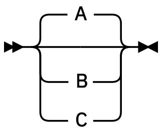
Syntax Diagram Convention	Example
Variables A variable appears in highlighted lowercase, usually italics. In this example, *var_name* represents a variable that you must specify following KEYWORD.	
Repetitions An arrow returning to the left means that the item can be repeated. A character within the arrow means that you must separate each repetition of the item with that character. A number (1) by the arrow references a syntax note at the bottom of the diagram. The syntax note tells you how many times the item can be repeated. Syntax notes may also be used to explain other special aspects of the syntax.	   Notes: [1] Specify *repeat* up to 5 times.
Required Item or Choice When an item is on the line, it is required. In this example, you must specify A. When two or more items are in a stack and one of them is on the line, you must specify one item. In this example, you must choose A, B, or C.	 
Optional Item or Choice When an item is below the line, it is optional. In this example, you can choose A or nothing at all. When two or more items are in a stack below the line, all of them are optional. In this example, you can choose A, B, C, or nothing at all.	 
Defaults When an item is above the line, it is the default. The system will use the default unless you override it. You can override the default by specifying an option from the stack below the line. In this example, A is the default. You can override A by choosing B or C.	
Repeatable Choice A stack of items followed by an arrow returning to the left means that you can select more than one item or, in some cases, repeat a single item. In this example, you can choose any combination of A, B, or C.	

Table 1. *Examples of Syntax Diagram Conventions (continued)*

Syntax Diagram Convention	Example
Syntax Fragment Some diagrams, because of their length, must fragment the syntax. The fragment name appears between vertical bars in the diagram. The expanded fragment appears in the diagram after a heading with the same fragment name. In this example, the fragment is named "A Fragment."	 **A Fragment** 

Examples of Messages and Responses

Although most examples of messages and responses are shown exactly as they would appear, some content might depend on the specific situation. The following notation is used to show variable, optional, or alternative content:

xxx
Highlighted text (usually italics) indicates a variable that represents the data that will be displayed.

[ ]
Brackets enclose optional text that might be displayed.

{ }
Braces enclose alternative versions of text, one of which will be displayed.

|
The vertical bar separates items within brackets or braces.

...
The ellipsis indicates that the preceding item might be repeated. A vertical ellipsis indicates that the preceding line, or a variation of that line, might be repeated.

Where to Find More Information

For more information about z/VM functions, see the books listed in the "Bibliography" on page 857.

If you plan to deploy Linux® on z/VM, read *z/VM: Getting Started with Linux on IBM Z* for important planning information about Linux virtual servers.

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: CP Planning and Administration

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.

SC24-6271-73, z/VM 7.3 (June 2025)

This edition includes changes to support product changes that are provided or announced after the general availability of z/VM 7.3.

[7.3 VM66823] z/VM support for the IBM z17 family

With the PTF for APAR VM66823, z/VM 7.3 provides support for the IBM z17 family. For more information, see z/VM support for the IBM z17 family in the *z/VM: Migration Guide*.

The following topics are updated:

- "DEFINE VSWITCH Statement" on page 107
- "MODIFY PORT Statement" on page 200
- "MODIFY VSWITCH Statement" on page 206
- "NICDEF Directory Statement" on page 565
- "RDEVICE Statement (Special Devices)" on page 247

[VM66832] Single System Image (SSI) prerequisite support for future-server compatibility

The PTF for APAR VM66832 provides infrastructure support in z/VM 7.3 for inclusion of an IBM z17 family server in a Single System Image (SSI) cluster. Proper guest relocation support requires that the PTF is installed on all the members of an SSI cluster before any member runs on an IBM z17 family server.

Miscellaneous updates for June 2025

The following topics are updated:

- "Creating Logo Screens" on page 340
- "DRAWLOGO" on page 838

SC24-6271-73, z/VM 7.3 (July 2024)

This edition includes changes to support product changes that are provided or announced after the general availability of z/VM 7.3.

[VM66749] Relax SCSI alternate path restrictions

With the PTF for APAR VM66749, the uniqueness requirement for the DEVICE *fcp_vdev* operand of the LOADDEV user directory statement and the SET LOADDEV and SET DUMPDEV commands is relaxed. Previously, the device number was required to be unique from any device numbers that were specified in any ALTERNATE path definitions. When the PTF is applied, the *combination of device number and port name* must be unique from those specified in any ALTERNATE path definitions.

The following directory statement is updated:

- “LOADDEV Directory Statement” on page 540

SC24-6271-73, z/VM 7.3 (June 2024)

This edition includes terminology, maintenance, and editorial changes.

Miscellaneous updates for June 2024

Casing of the WARNING operand is corrected in the following system configuration statement:

- “PAGING Statement” on page 219

SC24-6271-73, z/VM 7.3 (December 2023)

This edition includes changes to support product changes that are provided or announced after the general availability of z/VM 7.3.

[VM66727] FCP SCSI List-directed IPL alternate paths

With the PTF for APAR VM66727, z/VM 7.3 provides the ability to specify alternate paths for list-directed SCSI IPL. If IPL fails for the primary device path, alternate device paths are used automatically. The new ALTERNATE operand of the **`SET DUMPDEV`** and **`SET LOADDEV`** commands and the LOADDEV directory statement can specify up to three alternate device paths. The responses to the **`QUERY DUMPDEV`** and **`QUERY LOADDEV`** commands indicate the alternate device paths.

The following configuration statements are updated:

- “IPL Directory Statement” on page 519
- “LOADDEV Directory Statement” on page 540

SC24-6271-73, z/VM 7.3 (September 2023)

This edition includes changes to support product changes that are provided or announced after the general availability of z/VM 7.3.

[VM66678, VM66709] Warning Track Interruption Facility

With the PTFs for APARs VM66678 (CP) and VM66709 (Performance Toolkit), z/VM 7.3 exploits a feature of Processor Resource/Systems Manager (PR/SM) called the *warning-track-interruption facility.* z/VM's exploitation of this facility helps improve guest response time and overall performance of workloads that are run on vertical-low or vertical-medium logical processors.

The following configuration statement is updated:

- “SRM Statement” on page 269

[VM66683] Channel Path Usage

In support of APAR VM66683, clarifications about channel path usage are added to the descriptions of these CP directory statements:

- “NICDEF Directory Statement” on page 565
- “SPECIAL Directory Statement” on page 597

SC24-6271-73, z/VM 7.3 (June 2023)

This edition includes terminology, maintenance, and editorial changes.

The following topics are updated:

- “LOADDEV Directory Statement” on page 540
- “Creating the Stand-Alone Dump Utility” on page 395

SC24-6271-73, z/VM 7.3 (May 2023)

This edition includes changes to support product changes that are provided or announced after the general availability of z/VM 7.3.

[VM66424, VM66434, VM66650] Guest Secure IPL Support

With the PTF for APARs VM66424, VM66434, and VM66650, z/VM V7.3 supports guest secure IPL (load and dump) for both ECKD and SCSI devices. A z/VM user can request that the machine loader validate the signed IPL code by using the security keys that were previously loaded by the customer into the HMC certificate store. The validation ensures that the IPL code is intact, unaltered, and originates from a trusted build-time source.

Support is provided for the following guest operating systems:

- This support provides the ability for a Linux guest to exploit hardware to validate the code being booted, helping to ensure it is signed by the client or its supplier.

  Linux on IBM zSystems instances that previously were able to perform secure boot first level on an IBM z15® or IBM LinuxONE III prior to Driver D41C Bundle S73a, or an IBM z16 or IBM LinuxONE 4 prior to Driver D51C Bundle S18, will no longer be able to use secure boot until appropriate additional support is applied to the Linux image. Details are available about the required service level of Linux to properly IPL securely first or second level after driver D41C Bundle S73a or Driver D51C Bundle S18 has been applied. See 230428 Machine Alert for 8561, 8562, 3931, 3932 (https://www-40.ibm.com/servers/resourcelink/lib03020.nsf/pagesByDocid/272B3DD994A65B538525899F005FA0E6?OpenDocument).
- z/OS® is supported in audit mode only. Full exploitation requires Virtual Flash Memory support, which is not available to a guest. In audit mode, the IPL code is checked but the IPL continues even if the code is not valid.

z/VM and the z/VM stand-alone dump utility do not support performing host IPL via List-Directed IPL (LD-IPL) from ECKD. In addition, Secure IPL of the z/VM host and z/VM stand-alone dump are not supported.

The following topics are updated:

- “LOADDEV Directory Statement” on page 540
- “IPL Directory Statement” on page 519
- “OPTION Directory Statement” on page 572

The following topic is new:

- “Guest Secure IPL” on page 384

[VM66677] Increase crashkernel area size

With the PTF for APAR VM66677, z/VM 7.3 increases the size of crashkernel area allocated by stand-alone dump programs in preparation for a future enhancement.

The following topics are updated:

- “Creating the Stand-Alone Dump Utility” on page 395
- “Real Storage Management” on page 633

SC24-6271-73, 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.

Eight-member SSI support

This support increases the maximum size of a single system image (SSI) cluster from four members to eight, enabling clients to grow their SSI clusters to allow for increased workloads and providing more flexibility to use live guest relocation (LGR) for nondisruptive upgrades and workload balancing.

The following configuration statement is new:

- "SSI_CONTROLS Statement" on page 275

The following configuration statement is updated:

- "SSI Statement" on page 273

The following directory statements are updated:

- "DIRECTORY Directory Statement" on page 503
- "SPOOLFILE Directory Statement" on page 605

The following topic is new:

- Chapter 35, "Adding Members to a 4-Member SSI Cluster," on page 811

The following topics are updated:

- "A z/VM SSI Environment" on page 715
- "Creating a z/VM SSI Cluster" on page 735
- Chapter 30, "Cross-System Spool in a z/VM SSI Cluster," on page 747
- "Spool File ID Assignment and Limits" on page 751
- Chapter 32, "Converting a z/VM System to a Single-Member z/VM SSI Cluster," on page 763
- "Task 7: Shut Down and Warm Start" on page 775
- Chapter 34, "Adding a Member to a z/VM SSI Cluster by Cloning an Existing Member," on page 785
- "Task 7: Update the System Configuration File" on page 801

NVMe emulated device (EDEVICE) support

NVMe devices that are connected through PCI Express (PCIe) adapters can be defined and managed as Fixed-Block Architecture (FBA) EDEVICEs. As such, all host and guest FBA functions are supported except for those functions that require stand-alone support such as warm start and checkpoint. Linux guests that exploit EDEVICEs that are defined on NVMe adapters are not eligible for live guest relocation. NVMe adapters are only available on LinuxONE servers. For more information, see "Emulated FBA Disks on NVMe Devices" on page 703.

Remove references to invalid HCPSYS macros

References to invalid HCPSYS macros are removed from the following topics:

- "DEVICES Statement" on page 123
- "Disassociating a User ID from the Retrieval of Accounting Records" on page 355
- "Disassociating a User ID from the Retrieval of EREP Records" on page 357
- "Disassociating a User ID from the Retrieval of Symptom Records" on page 362
- "OPERATOR_CONSOLES Statement" on page 217
- "SYSTEM_USERIDS Statement" on page 294

Miscellaneous updates for z/VM 7.3

The following topics are updated:

- "CRYPTO APVIRTUAL Statement" on page 80

- “DEVICES Statement” on page 123
- “EMERGENCY_MESSAGE_CONSOLES Statement” on page 144
- “Security Considerations and Guidelines” on page 377
- “STORAGE Statement” on page 278
- “Task 6A: Update the User Directory Using DirMaint” on page 796
- Support for 4-character time zones:
  - “TIMEZONE_BOUNDARY Statement” on page 298
  - “TIMEZONE_DEFINITION Statement” on page 300

Part 1. Overview

Chapter 1. Planning and Administration Overview

This topic summarizes planning and administration tasks for the z/VM licensed program. Planning involves deciding how your system will be organized, configured, and used. Administration involves setting up, configuring, and modifying the system. Planning tasks and administration tasks are discussed together because the tasks are interrelated.

When planning for users, for example, it is useful to know how user IDs are added after the system is installed and running. You might then decide to define a minimal set of user IDs initially (perhaps those needed for key personnel, service machines, and guest operating systems), and add other user IDs later.

User IDs and many other aspects of the system configuration are defined by statements in z/VM files. Many administration tasks involve changing those files and then processing the files in some way to activate the changes. This topic contains overviews of the following files:

- System configuration file
- Logo configuration file
- User directory

These files are used in the planning and administration tasks for:

- The system
- Users
- Storage
- Performance

Although this topic does not describe how to perform any planning or administration tasks, it tells you where to find information that does.

This manual covers planning and administration for the base CP system. The complete set of manuals is listed under the "Planning and Administration" section of the "Bibliography" on page 857. Other manuals focus on topics such as connectivity, security, the Shared File System (SFS), and the Group Control System.

Migration Planning

If you are migrating from a previous VM release, see *z/VM: Migration Guide*.

Major z/VM Files

The following z/VM files are used to define and tailor many characteristics of your installation's z/VM system:

- The system configuration file (SYSTEM CONFIG) defines real I/O devices in your system's I/O configuration and operating characteristics such as the layout of the CP system residence disk, lists of DASD volumes that CP uses, the real storage configuration, and information CP requires to determine the correct offset from Coordinated Universal Time (UTC).

  For more information, see Chapter 2, "Configuring Your System," on page 15, Chapter 4, "Defining I/O Devices," on page 27, and Chapter 6, "The System Configuration File," on page 47.
- The logo configuration file (LOGO CONFIG) defines where CP can find:
  - Logos for local, logical, and VTAM-attached devices
  - Status area definition
  - Online message and input area definition information
  - Print separator pages for printers.

For more information, see Chapter 2, “Configuring Your System,” on page 15 and Chapter 7, “The Logo Configuration File,” on page 337.

- The user directory (USER DIRECT) defines the users of the system, their authority to enter various commands, and the resources they can use in the system. Each user who can log on to z/VM must have a virtual machine definition in the z/VM user directory. (You do not need to name your directory USER DIRECT.)

  For more information, see Chapter 20, “Creating and Updating a User Directory,” on page 459.

System Configuration Planning and Administration

The following list briefly describes system configuration planning and administration tasks and where to find detailed information about each task:

- **Define operating characteristics of your system** by coding the system configuration file. Information on the system configuration file can be found in Chapter 2, “Configuring Your System,” on page 15 and Chapter 6, “The System Configuration File,” on page 47.
- **Specify the local time** to provide CP with the information it needs to determine the correct offset from Coordinated Universal Time (UTC). Specify the local time on the TIMEZONE_BOUNDARY and TIMEZONE_DEFINITION statements in the system configuration file. For more information on TIMEZONE_BOUNDARY and TIMEZONE_DEFINITION, see “TIMEZONE_BOUNDARY Statement” on page 298 and “TIMEZONE_DEFINITION Statement” on page 300.
- **Create a system name for the processor on which you run**. Code the SYSTEM_IDENTIFIER and SYSTEM_IDENTIFIER_DEFAULT statements in the system configuration file to create a system name for each processor on which you run z/VM. The system name appears on printed output separator pages and in the status area of the display screen. For more information, see “SYSTEM_IDENTIFIER Statement” on page 287 and “SYSTEM_IDENTIFIER_DEFAULT Statement” on page 290.
- **Set up service virtual machines** for accounting, error recording, symptom record recording, communication controller support for the emulator program, service spool support, and data storage management. Details on setting up these virtual machines can be found in Chapter 8, “Setting Up Service Virtual Machines,” on page 353.

  To use the CMS shared file system, set up file pool service and file pool administration virtual machines. Details on how to do this are in *z/VM: CMS File Pool Planning, Administration, and Operation*.
- **Use SNA communication products to use SNA terminals as virtual machine consoles**. Systems Network Architecture/Console Communications Services (SNA/CCS) provides a total communication structure for transmitting information through a communications network. For details on SNA/CCS, refer to Chapter 9, “Planning for SNA Console Communication Services (SNA/CCS),” on page 367.
- **Plan for security facilities**. Facilities are available in z/VM to help protect the system from security and integrity exposures. For descriptions and use of these facilities, see Chapter 11, “Security and Integrity in z/VM,” on page 377.
- **Improve system availability by defining and saving multiple copies of the CP module**. For more information, see Using the Stand-Alone Program Loader in *z/VM: System Operation*.
- **Create the stand-alone dump utility program**. Refer to Chapter 12, “The Stand-Alone Dump Utility,” on page 395.

Additional system configuration planning and administration tasks are required for multisystem environments. See “Planning for Multisystem Environments” on page 11.

User Planning and Administration

The following list describes user planning and administration tasks and where to find information on each task:

- **Plan for any changes to command privilege classes**. You can extend the privilege class structure of CP commands, DIAGNOSE codes, and certain CP system functions from eight classes to as many as 32 classes. For detailed information on how to extend the privilege class structure, refer to Chapter 19, “Redefining Command Privilege Classes,” on page 449.

- **Create a z/VM user directory**. As mentioned under “Major z/VM Files” on page 3, each user who can log on to z/VM must have a virtual machine definition in the z/VM user directory. You can create your own user directory or update the sample that z/VM provides. See Chapter 20, “Creating and Updating a User Directory,” on page 459 which contains detailed descriptions of the directory statements you can code in the user directory.
- **Define virtual machine modes and processor configurations**. Coding the MACHINE statement in a user or identity entry allows you to specify:

  1. The virtual machine mode (ESA, XA, XC, or Z), which indicates the architecture that the virtual machine simulates.

     Note: XA mode is supported for compatibility and is functionally equivalent to ESA mode. Some CMS applications might require the virtual machine to be in XA mode.
  2. The maximum number of virtual processors the virtual machine can define. To define virtual processors, the virtual machine user must enter the DEFINE CPU command.

  For information, see “MACHINE Directory Statement” on page 546.

  As an alternative to including a MACHINE statement in each user or identity entry, you can define the virtual machine mode and virtual processor capabilities for a group of virtual machines by including the MACHINE directory statement in a profile entry. See “Creating Directory Profiles” on page 470.

  If the MACHINE statement is not included in the user or identity entry, or included in a profile, the default virtual machine mode is XA and the maximum number of virtual processors the virtual machine can define is determined by the number of CPU statements included in the user or identity entry, or in a profile. If no CPU statements are included in the user or identity entry, or a profile, the virtual machine has no virtual multiprocessor capabilities.

  The MACHINE operand on the GLOBALOPTS directory statement allows you to specify a global virtual machine mode for all virtual machines that do not have a MACHINE statement in their user or identity entry and are not included in a profile entry that contains a MACHINE statement. For more information, see the “GLOBALOPTS Directory Statement” on page 509.
- **Determine how you want real processor resources dedicated**. The system operator can use the DEDICATE command to dedicate virtual CPUs to real processors.

  To dedicate specific virtual CPUs to real processors at logon, add a DEDICATE operand for each CPU directory statement that is to have automatic dedication.

  For information on coding the CPU directory statement, see Chapter 20, “Creating and Updating a User Directory,” on page 459.
- **Make sure that the user IDs specified in the system configuration file have virtual machine definitions**. The sample system configuration file that is shipped on the z/VM System DDR media or DVD defines the following user IDs (on the SYSTEM_USERIDS statement), and the sample directories contain virtual machine definitions for them:

  – OPERATOR as the user ID for the primary system operator

  – DISKACNT as the user ID for the accounting virtual machine

  – EREP as the user ID for an error recording virtual machine

  – OPERATNS as the user ID for a virtual machine that receives system dumps.

  If you change these or other user IDs specified in the system configuration file, make sure you change the user directory.
- **Decide which users are to be enrolled in a file pool**. Detailed instructions on enrolling users in a file pool can be found in *z/VM: CMS File Pool Planning, Administration, and Operation*.
- **Create a PROFILE EXEC for the system bring-up virtual machine**. The system bring-up machine (which, by default, has a user ID of AUTOLOG1) provides an optional way to automatically log on

virtual machines during system initialization. As part of system initialization, CP automatically logs on AUTOLOG1. AUTOLOG1, in turn, can automatically log on other virtual machines, assuming that:

- A PROFILE EXEC for AUTOLOG1 issues the XAUTOLOG command for each virtual machine AUTOLOG1 is to log on. (The XAUTOLOG command is asynchronous.) The system, however, automatically logs on the error recording, accounting, and symptom record recording virtual machines without receiving a command from AUTOLOG1's PROFILE EXEC. For information on creating CMS PROFILE EXEC files, see *z/VM: CMS User's Guide*. For information on the XAUTOLOG command, see *z/VM: CP Commands and Utilities Reference*. For information on the XAUTOLOG directory statement, see "XAUTOLOG Directory Statement" on page 624.

- **Determine which operating system will be in each virtual machine**. *z/VM: Running Guest Operating Systems* contains information on various operating systems as guests running in virtual machines.
- **Decide what the default POSIX authorizations should be**. The USER_DEFAULTS system configuration file statement determines what the defaults will be for querying other users' POSIX database information and having their POSIX security values changed. For more information, see "USER_DEFAULTS Statement" on page 309.

Real and Virtual Storage Planning and Administration

The following list describes real and virtual storage tasks and where to find information about each task:

- **Configure real storage**. The INITIAL and RESERVED real storage for the logical partition in which z/VM is installed are set on the IBM Hardware Management Console (HMC). There are also actions you can take on z/VM to limit or increase the amount of real storage available to z/VM. You can also take actions to define this storage so that it can be removed from the system later.

  For more information, see Chapter 21, "Host Storage Planning and Administration," on page 633.
- **Provide virtual storage to virtual machines**. The amount of virtual storage CP assigns to a virtual machine when the virtual machine logs on is determined by an operand on the USER or IDENTITY statement in the virtual machine's definition, or through the STORAGE or MAXSTORAGE statements in the user, identity, or subconfiguration entries. The USER or IDENTITY statement can also set the maximum storage amount that each particular virtual machine user can define using the DEFINE STORAGE command.

  For more information, see "USER Directory Statement" on page 616, "IDENTITY Directory Statement" on page 510, "MAXSTORAGE Directory Statement" on page 548, and "STORAGE Directory Statement" on page 608.
- **Plan for using saved segments**. A saved segment is an area of virtual storage that holds data or reentrant code. Defining frequently-used data as saved segments provides several advantages.

  For more information, see *z/VM: Saved Segments Planning and Administration*.

Auxiliary Storage Planning and Administration

The following sections describe planning and administration tasks for CP-owned direct access storage devices (DASDs), dedicated DASDs, DASDs for minidisks, DASDs for shared file pools, and virtual disks in storage.

CP-Owned DASDs

CP-owned DASDs are used for the CP system residence volume, real system paging, spooling, directory and dump space, and temporary disk space for virtual machines. Do the following for CP-owned DASDs:

- **Use HCM and HCD to define the DASD or, if needed, code an RDEVICE statement in the system configuration file for the DASD**. For more information on HCM and HCD, see z/OS and z/VM: Hardware Configuration Manager User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/eequ100_v2r5.pdf) and *z/VM: I/O Configuration*. For more information on coding the RDEVICE statement, see "RDEVICE Statement (DASD)" on page 236.

- **Format CP-owned DASDs**. Before CP can use a DASD as a CP-owned DASD, you must use the CPFMTXA utility, or the Device Support Facilities (ICKDSF) program to format the DASD. However, using ICKDSF is the recommended method to format DASD volumes for CP use. For more information on CPFMTXA, refer to *z/VM: CP Commands and Utilities Reference*. For more information on ICKDSF, refer to *Device Support Facilities User's Guide and Reference*.
- **Define a list of CP-owned volumes on the CP_OWNED system configuration file statement**. For more information, see "CP_OWNED Statement" on page 73.
- **Define the layout of the CP system residence volume on the FEATURES and SYSTEM_RESIDENCE statements in the system configuration file**. For more information, see "FEATURES Statement" on page 158 and "SYSTEM_RESIDENCE Statement" on page 292.
- **Create a parm disk** You must store system configuration files on a parm disk, which is a CMS-formatted minidisk on the IPL volume. For more information, see "Using Configuration Files" on page 15.
- **Allocate temporary disk space to virtual machines**. Use the MDISK directory statement to define temporary disk space for virtual machines. Virtual machine users can enter the DEFINE command to define temporary disk space when they need it. For more information, see "MDISK Directory Statement" on page 550.

Dedicated DASDs

A dedicated DASD is a DASD that CP allocates exclusively to a virtual machine. Do the following for a dedicated DASD:

- **Use HCM and HCD to define the DASD or, if needed, code an RDEVICE statement in the system configuration file for the DASD**. For more information on HCM and HCD, see z/OS and z/VM: Hardware Configuration Manager User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/eequ100_v2r5.pdf) and *z/VM: I/O Configuration*. For more information on coding the RDEVICE statement, see "RDEVICE Statement (DASD)" on page 236.
- **Code a DEDICATE statement in the virtual machine definition of the virtual machine to which you dedicate the DASD**. To dedicate the DASD after the user logs on, use the ATTACH command.
- **Set up a dedicated DASD so it can be shared with an operating system running on another processor (using reserve/release)**. For more information, see "Sharing DASD between One Virtual Machine and Other Systems Using Real Reserve/Release" on page 670.

Note: If the DASD has not been used previously by the virtual machine's operating system, it must be initialized by the user after it is attached.

DASDs Used for Minidisks

To use a DASD to provide minidisks for virtual machines, do the following:

- **Use HCM and HCD to define the DASD or, if needed, code an RDEVICE statement in the system configuration file for the DASD**. For more information on HCM and HCD, see z/OS and z/VM: Hardware Configuration Manager User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/eequ100_v2r5.pdf) and *z/VM: I/O Configuration*. For more information on coding the RDEVICE statement, see "RDEVICE Statement (DASD)" on page 236.
- **Define on the USER_VOLUME_LIST statement in the system configuration file a list of volumes you use to contain minidisks**. For more information, see "USER_VOLUME_LIST Statement" on page 316.

  Note that you can also define minidisks on CP-owned volumes if the CP-owned volume is formatted appropriately. Although you may use the CPFMTXA utility, it is recommended that you use the Device Support Facilities (ICKDSF) program to format DASD volumes for CP use. For more information on the CPFMTXA utility, see *z/VM: CP Commands and Utilities Reference*. For more information on ICKDSF, see *Device Support Facilities User's Guide and Reference*.
- **Define minidisks for users**. Use the MDISK directory statement to define a minidisk for a virtual machine. Use the LINK directory statement to define a link to another user's minidisk. The virtual machine for which the minidisk is defined is responsible for formatting it. For more information, see "MDISK Directory Statement" on page 550 and "LINK Directory Statement" on page 533.

In an SSI-enabled directory, MDISK definitions can be global or local. A global MDISK definition is included in a user or identity entry and the minidisk being defined can be linked by virtual machines on any member of the SSI cluster. A local MDISK definition is included in a subconfiguration entry and the minidisk being defined can be linked only by virtual machines on the SSI member to which the subconfiguration entry applies.

Note: CP and the directory program do *not* prevent you from defining minidisks that overlap. If you define such overlap, you assume responsibility for data integrity. You can use the Directory Maintenance Facility (DirMaint) optional feature of z/VM to assist in managing the user directory.

- **Set up a minidisk so it can be shared with an operating system running on another processor and with other virtual machines (using reserve/release)**. For more information, see "Sharing DASD among Multiple Virtual Machines by Using Virtual Reserve/Release" on page 667.

DASD Space Used for Shared File Pools

To define file pools, you need to estimate how large each file pool will become. Also define one or more virtual machines to be used for file pool service machines. In so doing, you must decide what the initial DASD storage for the file pool will be.

For information on generating a file pool, refer to *z/VM: CMS File Pool Planning, Administration, and Operation*.

Virtual Disks in Storage

A virtual disk in storage is the temporary simulation of an FBA minidisk in an address space in host storage. Because a virtual disk in storage is not mapped to a real DASD, having a real FBA DASD in the system configuration is not required. By avoiding the I/O overhead, virtual disks in storage may be faster to use than other minidisks.

There are two ways to define a virtual disk in storage:

- Using the DEFINE command. This creates a private (nonshareable) virtual disk in storage that is destroyed when the user detaches it or logs off.

  There is a limit on the amount of storage that can be allocated for virtual disks in storage created by a single user using the DEFINE command. This is called the user limit. The built-in default for the user limit is 0. You can override the built-in default by defining the user limit on the FEATURES statement in the system configuration file or by using the SET VDISK command. For more information, see "FEATURES Statement" on page 158.

- Using the MDISK directory statement. This creates a shareable virtual disk in storage. Use the LINK directory statement to define a link to another user's virtual disk in storage. A shareable virtual disk in storage is created when the first user links to it (the owner links to it by logging on) and destroyed when the last user detaches it or logs off. The first user must initialize or format the virtual disk in storage. For more information, see "MDISK Directory Statement" on page 550 and "LINK Directory Statement" on page 533.

  Note: An MDISK statement for a virtual disk in storage can be included only in a user entry or subconfiguration entry. The scope of the virtual disk in storage is local, which means it can be shared only with users on the system where it is created.

  There is a limit on the total amount of storage that can be allocated for virtual disks in storage on the system. This is called the system limit. The built-in default for the system limit is the minimum of:

  – The amount of virtual storage that can be represented by one-quarter of the usable dynamic paging area (DPA) below 2 GB (based on the fact that each gigabyte of virtual disk defined requires 2050 pages of real storage below 2 GB)

  – The amount of storage represented by one-quarter of the paging space defined for CP use.

  You can override the built-in default by defining the system limit on the FEATURES statement in the system configuration file or by using the SET VDISK command. For more information, see "FEATURES Statement" on page 158.

Guests, servers, and other applications that use FBA minidisks can use virtual disks in storage without recoding. Because of their volatility, virtual disks in storage should not be used for permanent data; work files and other files that hold temporary results may be appropriate for virtual disks in storage. For an example of coding VSE guests to use a virtual disk in storage for storing label information areas and the cross system communication file (lock file), see *z/VM: Running Guest Operating Systems*.

Terminals

The following list describes tasks for terminals and where to find information about each task:

- **For all real terminals use HCM and HCD to define them or code an RDEVICE statement in the system configuration file**. For more information on HCM and HCD, see z/OS and z/VM: Hardware Configuration Manager User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/eequ100_v2r5.pdf) and *z/VM: I/O Configuration*. Otherwise, see "RDEVICE Statement (Graphic Display Devices)" on page 239.
- **Define primary and alternate system consoles using HCM and HCD or define primary and alternate system consoles on either the OPERATOR_CONSOLES or EMERGENCY_MESSAGE_CONSOLES statement in the system configuration file**. For more information on HCM and HCD, see z/OS and z/VM: Hardware Configuration Manager User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/eequ100_v2r5.pdf) and *z/VM: I/O Configuration*. Otherwise, see either "EMERGENCY_MESSAGE_CONSOLES Statement" on page 144 or "OPERATOR_CONSOLES Statement" on page 217.
- **To define a virtual machine operator console for a virtual machine, code the CONSOLE directory statement in the virtual machine definition for a user**. For more information, see "CONSOLE Directory Statement" on page 485.
- **Define terminals for virtual machines**. To define a terminal for a virtual machine, code the DEDICATE or SPECIAL directory statements, or both, in the virtual machine definition for a user. For more information, see "DEDICATE Directory Statement" on page 500 and "SPECIAL Directory Statement" on page 597.

Unit Record Devices

The tasks for a unit record device depend on whether the unit record device is (a) dedicated to a virtual machine or (b) used for spooling.

Dedicated Unit Record Devices

To dedicate a unit record device to a virtual machine, perform the following:

- **Use HCM and HCD to define the device or code an RDEVICE statement in the system configuration file for the device**. For more information on HCM and HCD, see z/OS and z/VM: Hardware Configuration Manager User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/eequ100_v2r5.pdf) and *z/VM: I/O Configuration*. Otherwise, see:
  - "RDEVICE Statement (Card Punches)" on page 230
  - "RDEVICE Statement (Card Readers)" on page 232
  - "RDEVICE Statement (Impact Printers)" on page 243
  - "RDEVICE Statement (3800 Printers)" on page 255
- **Code a DEDICATE statement in the virtual machine's definition**. The virtual machine user is responsible for other device tasks (such as creating image libraries for dedicated 3800 printers, defining forms control buffers, and so on). For more information, see "DEDICATE Directory Statement" on page 500.

Spooled Unit Record Devices

To use a unit record device for spooling, perform the following:

- **Use HCM and HCD to define the device or code an RDEVICE statement in the system configuration file for each unit record device you use for spooling**. For more information on HCM and HCD, see z/OS and z/VM: Hardware Configuration Manager User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/eequ100_v2r5.pdf) and *z/VM: I/O Configuration*. Otherwise, see “RDEVICE Statement” on page 227.
- **Create a list of user form names and their corresponding operator form numbers on either the FORM_DEFAULT or USERFORM statements**. For more information, see “FORM_DEFAULT Statement” on page 173 or “USERFORM Statement” on page 308.
- **Specify classification titles for specific classes of spooled output on the PRINTER_TITLE statement**. For more information, see “PRINTER_TITLE Statement” on page 221.
- **Define forms control buffers (FCBs) and universal character sets (UCSs) for 3203, 3211, 3262, 4245, and 4248 printers**. For more information, see Chapter 13, “Creating and Modifying Image Libraries for Printers,” on page 403.
- **Define image libraries for 3800 and impact printers**. For more information, see Chapter 13, “Creating and Modifying Image Libraries for Printers,” on page 403.
- **Define spooled unit record devices for virtual machines**. For more information, see “SPOOL Directory Statement” on page 602.

Performance Planning and Administration

The performance characteristics of an operating system are dependent on such factors as choice of hardware, the total number of users on the system during peak periods, functions being performed by the system, and the way system parameters are set up. You can improve performance to some degree by the choice of hardware and system options. The following general tasks pertain to improving your z/VM system efficiency:

- Plan how you will handle performance monitoring, measurements, improvements, and problems. Become familiar with the CP monitor facility and the facilities you can manipulate to change the performance characteristics of the system as a whole or of selected virtual machines.
- Before you decide which performance options to apply, monitor the system's current performance. This will help you determine which options would most likely give the system a performance gain and where performance bottlenecks are occurring. The CP monitor facility collects such data, which can then be processed to produce statistics to give you an understanding of system operation.
- Perform system tuning to do any of the following:
  - Process a larger or more demanding work load without increasing the system configuration
  - Obtain better system response or throughput
  - Reduce processing costs without affecting service to users.

Details on system tuning can be found in *z/VM: Performance*.

I/O Reconfiguration in z/VM

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

z/VM support allows the system administrator or system operator to use the HCM/HCD interface or CP's dynamic I/O configuration commands to change the I/O configuration of the processor without a re-IPL of z/VM or a power-on reset of the processor.

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

- Query the logical partitions on a machine
- Query the channel paths

- Query the status of channel paths to devices

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

OpenExtensions Planning

Planning for OpenExtensions involves setting up the OpenExtensions facilities in z/VM that allow users to run POSIX applications. These tasks involve assigning POSIX security values to users (by specifying certain system configuration file statements and CP directory statements) and setting up the OpenExtensions Byte File System. For an overview of how to set up OpenExtensions, see *z/VM: OpenExtensions User's Guide*.

Planning for Multisystem Environments

Additional planning and configuration tasks are required to enable a z/VM system to participate in a multisystem environment:

- **Plan for a z/VM single system image (SSI) cluster**. A z/VM SSI cluster offers a comprehensive clustering solution that includes multisystem installation, single point of maintenance, autonomic minidisk cache management, and virtual server mobility. For additional information, see Chapter 29, “Setting Up z/VM Single System Image Clusters,” on page 715.
- **Plan for cross-system link (XLINK)**. The XLINK function extends CP link protocols to control normal, stable, and exclusive read or read/write access across multiple z/VM systems for minidisks on shared DASD. For additional information, see Appendix C, “Using Cross-System Link (XLINK),” on page 843.

Part 2. System Planning and Administration

Chapter 2. Configuring Your System

This chapter provides an introduction to the files that define the configuration of your system.

Specifying System Configuration Information

Most of the information about the configuration of your system is defined in the system configuration file, which is a CMS data file. Additional configuration information is defined in the user directory files. You can also define the characteristics of screen logos in a logo configuration file. While the system is running, you can dynamically change many of these system characteristics by issuing various CP commands.

This chapter provides an overview of the configuration files and what they contain. The following chapters provide additional information about configuring your system:

- Chapter 4, "Defining I/O Devices," on page 27 discusses considerations for defining I/O devices.
- Chapter 6, "The System Configuration File," on page 47 describes the system configuration file in more detail.
- Chapter 7, "The Logo Configuration File," on page 337 describes the logo configuration file in more detail.
- Chapter 19, "Redefining Command Privilege Classes," on page 449 describes the user class structure.
- Chapter 20, "Creating and Updating a User Directory," on page 459 describes how to define the user directory.

Using Configuration Files

The system configuration file and the logo configuration file, as well as any files that these configuration files imbed, are stored on the *parm disk*. The parm disk is a regular CMS-formatted minidisk that you identify to CP by writing an allocation map on the IPL volume marking the location of the minidisk. The parm disk is accessed at IPL time, and configuration information is read from the configuration files stored on the disk.

To place configuration information on the parm disk, you must do the following:

1. Choose a CMS-formatted minidisk on the IPL volume on which you plan to place configuration files. (If the minidisk is not CMS-formatted, the system will enter a wait state when you try to use it.)
2. Run the ICKDSF utility to rewrite the allocation map on the IPL volume to mark a parm disk extent that covers the location of the minidisk. See the *Device Support Facilities User's Guide and Reference* for details on formatting and allocating DASD volumes using ICKDSF.
3. Place a system configuration file (called SYSTEM CONFIG by default), a logo configuration file (called LOGO CONFIG by default), or both, on the parm disk. Other names can be used for these files. You can specify the name of the system configuration file when you IPL. For more information, see Passing IPL Parameters in *z/VM: System Operation*. You can specify a name for the logo configuration file inside the system configuration file. Place any logo picture files on this disk also.

Note: Although the system control file (HCPSYS ASSEMBLE), system real I/O configuration file (HCPRIO ASSEMBLE), and system logo definition file (HCPBOX ASSEMBLE) are still supported, IBM strongly recommends that you use configuration files to define your system. Using the ASSEMBLE files is more difficult and error-prone, requires availability and knowledge of the necessary level of the IBM High Level Assembler licensed program (or an equivalent product), does not support recent CP enhancements, and requires rebuilding the CP module after making changes.

The HCPSYS ASSEMBLE file supplied with z/VM contains no system definitions. The supplied HCPRIO ASSEMBLE file contains only an RIOGEN CONS=DYNAMIC macro to indicate that console addresses are defined in the system configuration file. The supplied HCPBOX ASSEMBLE contains default system logos.

For information about the system definition macros, see the z/VM 6.2 edition of this document.

A default SYSTEM CONFIG file is created at installation time. You can update this file or create your own system configuration file using the editor of your choice.

During the IPL process, CP tries to access the first parm disk that is marked in the allocation map on the IPL volume. If there is a parm disk marked, and if the disk locations matching the extent represent a CMS-formatted minidisk, CP attempts to read the system configuration file from the disk.

The format of the system configuration file is described in "What You Can Specify in the System Configuration File" on page 16. If CP finds a file called SYSTEM CONFIG (or the name you specify on IPL), it examines the contents of the file and applies any changes requested in the file to the system configuration. Values defined in the system configuration file override any values defined in HCPSYS ASSEMBLE and HCPRIO ASSEMBLE. If CP finds a file called LOGO CONFIG (or the name defined in the system configuration file), it uses the contents of that file to override all the logo characteristics set in HCPBOX ASSEMBLE.

Before the first prompt, which asks you what method of system initialization you need, the system displays information about the parm disk. After the system is up and running, you can use the QUERY CPLOAD command to find out whether CP accessed a parm disk, and if so, what that disk was.

What You Can Specify in the System Configuration File

The system configuration file defines the basic characteristics of your system and allows you to define many system options, such as:

- The name and members of a single system image (SSI) cluster.
- The devices CP should bring online at IPL time
- The time zone CP should select at IPL time from a list of time zones
- Whether CP should autolog special user IDs at IPL time, such as the accounting and symptom user IDs
- The contents of most translation tables defined in CP
- Whether CP should automatically attempt a warm start without changing the TOD clock at IPL time
- The number of buffers CP provides for the retrieving of command lines
- The characters used as default terminal characters, such as the line end character and the line delete character
- The defaults that will be used for querying other users' POSIX database information and having their POSIX security values changed
- The IODF that HCD will use to control CP's I/O configuration
- Whether multithreading is enabled

Note: You cannot define the size of real storage in the system configuration file. You can define an amount of storage less than the configured real storage by specifying the STORE parameter when you IPL. For more information, see Passing IPL Parameters in *z/VM: System Operation*.

Contents of the System Configuration File

The following sections contain a brief description of some of the statements that you can specify in the system configuration file. For complete descriptions of all the system configuration statements and general rules for coding a system configuration file, see Chapter 6, "The System Configuration File," on page 47.

The Bare Minimum

You must include the following statements for the system to come up:

- CP_OWNED defines a volume in the CP-owned volume list. After the system is running, you can use the DEFINE CPOWNED command to change this list. (To make the change permanent, update the system configuration file.)
- SYSTEM_RESIDENCE specifies the location of the checkpoint and warm start areas.

- OPERATOR_CONSOLE defines a list of console addresses from which CP should choose the operator console.

These three statements are all that the system really needs to come up and start running.

Other Important Statements

You may also want to include the following statements:

- SYSTEM_IDENTIFIER and SYSTEM_IDENTIFIER_DEFAULT define system names for the processors on which you run z/VM.
- TIMEZONE_DEFINITION and TIMEZONE_BOUNDARY enable the system to choose the correct local time zone definition.
- IODF indicates that HCM and HCD will control the hardware and optionally the software I/O configuration.
- SSI identifies the name and members of a single system image (SSI) cluster, and identifies the location of the persistent data record (PDR) for the cluster.

Real Devices

The statements previously listed are the most basic ones that you need to get your system running and to give it an identifier and a local time. You can customize your system further with the other statements. For example, if you are not using HCM and HCD to control your software I/O configuration, you can define real devices to the system and establish characteristics for them using the following statements:

- RDEVICE defines real I/O devices that do not respond to a sense ID request and I/O devices that need more defining information than a sense ID request returns (for example, printers and communications controllers). After the system is running, you can use the SET RDEVICE command to define or redefine real devices.

  See Chapter 4, "Defining I/O Devices," on page 27 for information about which devices may need to be defined with RDEVICE statements.
- DEVICES tells CP whether to bring specified devices online, accept and build real device blocks for specified devices, and use the information returned from a sense ID request to help define a device. It is recommended that you let CP bring all devices online at IPL, and that you let CP use the sense information.
- HOT_IO_RATE specifies the maximum unsolicited interrupt rate for real devices.

You can control operations on real DASD using the following statements:

- DRAIN stops new allocations of certain types of space on the DASD.
- START restarts a DASD after it has been drained.

After the system is running, you can use the DRAIN (Disk) and START (Disk) commands to perform the same functions as the DRAIN and START statements.

User Volume List

You can generate and change the user volume list using the following statements:

- USER_VOLUME_LIST generates the user volume list.
- USER_VOLUME_EXCLUDE and USER_VOLUME_INCLUDE specify volumes to be excluded from or included in the user volume list. These statements can use wildcard characters (% and *) in defining volume serial identifiers.

Cross-System Link Operations

You can control cross-system link (XLINK) operations using the following statements:

- XLINK_DEVICE_DEFAULTS changes the defaults for the location and format of the CSE area for specific DASD types.
- XLINK_SYSTEM_INCLUDE and XLINK_SYSTEM_EXCLUDE specify the systems to be included in or excluded from cross-system link protection.
- XLINK_VOLUME_INCLUDE and XLINK_VOLUME_EXCLUDE specify the DASD volumes to be included in or excluded from cross-system link protection.

Other System Attributes

You can set other attributes of your system using the following statements:

- EMERGENCY_MESSAGE_CONSOLES specifies a list of consoles that CP should notify if there is an impending abnormal end or other system emergency. If you do not include this statement, CP uses the list specified on the OPERATOR_CONSOLES statement for the list of emergency consoles as well.
- STORAGE allocates real storage and trace frames.
- JOURNALING specifies characteristics of the system's journaling facility. After the system is running, you can use the QUERY and SET JOURNAL commands to work with the journaling facility.
- PRIV_CLASSES changes the privilege classes authorizing certain CP functions.
- SYSTEM_USERIDS specifies user IDs that perform special functions during and after IPL. These user IDs identify the virtual machines that handle accounting records, system dump files, EREP records, and symptom records; the operator and startup user IDs are also specified.
- FEATURES sets several different attributes of the system. These attributes include:
  - Trying an automatic warm start without changing the TOD clock
  - Clearing TDISK DASD space automatically
  - Enabling the new LOGMSG support that causes CP to read log message information from disk files instead of the warm start area
  - Giving end users the authority to change their own privilege classes, and privileged users the authority to change the privilege classes of other users logged on to the system
  - Establishing the largest number of users who may be logged on to the system at one time
  - Enabling the facility that suppresses the password on the command line for AUTOLOG, LOGON, and LINK commands.
  - Allowing PCI functions to come online to the system for use.
- CHARACTER_DEFAULTS specifies default characters, such as the line end character and the character delete character.
- USER_DEFAULTS determines what the defaults will be for:
  - The global lines per page value for all virtual printers and consoles that are defined on the system, and
  - Querying other users' POSIX database information and having their POSIX security values changed.
  - TRANSLATE_TABLE specifies replacements for the standard translation tables that CP uses to accomplish certain tasks.
- CPXLOAD loads a file containing customer-written CP routines.
- MULTITHREADING specifies whether multithreading is enabled or disabled, and defines system options that can be customized if multithreading is enabled.
- CRYPTO APVIRTUAL specifies the APs and domains for shared crypto use.
- PRODUCT defines a product or feature to the system.

Semantic and Syntactic Statements

The following statements affect the way that your system processes the system configuration file:

- BEGIN and END identify blocks of statements that apply to particular systems.

- EQUATE creates names for groups of systems that all have something in common and should, therefore, be treated similarly. You can use system names or nicknames created by EQUATE statements to preface statements that should only apply to certain systems.
- IMBED specifies the name and type of a file to be imbedded into the system configuration file.
- TOLERATE_CONFIG_ERRORS controls whether a section of the system configuration file must be without errors when CP processes it.

Spool File Processing

The following statements affect the way that your system processes spool files:

- FORM_DEFAULT defines default user form names for CP to use when it creates files on virtual printers, punches, consoles, or real card readers.
- PRINTER_TITLE specifies the printed output classes that are to contain classification titles.
- USERFORM creates a list of user form names and their corresponding operator form numbers, and can specify the forms as NARROW so that a narrow separator page is printed.

CP File System

CP_ACCESS statements direct CP to access CMS-formatted minidisks that contain information that CP can use at run time. Each CP_ACCESS statement specifies a CMS-formatted minidisk that CP should access after it brings the user directory online. After the system is up and running, you can use the CPACCESS command to perform the same function as the CP_ACCESS statement.

Logo Processing

LOGO_CONFIG specifies the name of a logo configuration file for CP to use.

What You Can Specify in the Logo Configuration File

You can use the logo configuration file to override all of the default logo information specified in the HCPBOX ASSEMBLE file included in the CP nucleus. You can change four logo pictures:

- Normal logo
- Minimum screen logo
- Basic logo
- Spooling logo

You can use statements in the logo configuration file to choose logo pictures for logical devices, SNA terminals, and locally attached terminals. CP can choose pictures for logical devices based on the user ID of the virtual machine that created the device, for SNA terminals based on the user ID of the VSM that is managing the terminal, and for locally attached terminals based on their device addresses. It can also choose pictures for locally attached terminals or logical devices based on the size of their screens.

You can also use the logo configuration file to define the contents of the following fields:

- The command area
- The input area at the bottom of the logo screen, and the layout of this area
- The online message found at the top of each logo screen
- The status areas (such as `VM READ` and `CP READ`)

Unless you specify a different file name and file type in the system configuration file, CP looks for a file called LOGO CONFIG after it has processed the system configuration file. If CP has accessed a parm disk successfully, it attempts to process a logo configuration file even if it does not find a system configuration file on the parm disk. After the system is up and running, you can use the REFRESH LOGOINFO command to change this information.

Contents of the Logo Configuration File

The logo configuration file contains the following four statements:

- CHOOSE_LOGO
- INPUT_AREA
- ONLINE_MESSAGE
- STATUS

For complete descriptions of these statements and information about creating a logo configuration file, see Chapter 7, “The Logo Configuration File,” on page 337.

Choosing Logo Picture Files

The CHOOSE_LOGO statement is the most complex of these statements. The CHOOSE_LOGO statement defines the logo picture files that CP uses for the following:

- Locally attached terminals. You can select certain pictures for certain terminal addresses.
- Terminals logging on through logical devices
- Terminals managed by certain VTAM® service machines (VSMs)
- Separator pages of printed output.

You can also select a different logo picture file for each locally attached terminal and different picture files for terminals logging on through different user IDs. You can specify a logo picture file to appear on screens that are too small to accommodate the picture that another statement has specified for them.

Input Area

The INPUT_AREA statement contains information defining the layout of the input area that contains the user ID, password, and command areas of the logo screen. You can specify the number of lines each field should have, where the fields should appear, and the text that surrounds each area. You can use the CHOOSE_LOGO statement to put specific text into the command area; this text appears by default on the logo screen. This option is helpful for systems on which users often dial through to PVM or VTAM. For example, you can include the text DIAL VTAM in the command area.

Online Messages

The ONLINE_MESSAGE statement specifies the file that contains the online message that you want to appear at the top of the logo screen. You can change this message to show the date that the system was brought online, the version of z/VM that you are running, or whatever other information you may want.

Status Area

The STATUS statement specifies the text that appears in the bottom right corner of the screen. You can use this statement to change the contents of the status area to mixed case (for example, `RUNNING` to `Running`) or to change the text entirely.

Chapter 3. Understanding the CP File System

This chapter describes:

- Initial state of the CP file system
- Changing the list of disks accessed by CP
- Performance considerations
- Displaying the contents of CP-accessed minidisks
- Changing information on CP-accessed minidisks.

Initial State of the CP File System

After CP has read and processed the system and logo configuration files, it releases the temporary link and access that it established to the parm disk. The IPL process continues as it would if there had been no parm disk present. CP then prompts the operator with the options for the type of start:

```
17:10:58 Start ((Warm|Force|COLD|CLEAN) (DRain) (DIsable) (NODIRect))
17:10:58         (NOAUTOlog)) or (SHUTDOWN)
```

If you choose to start with no options or with any option other than NODIRECT, CP tries to bring the user directory online after it has performed the requested spool file initialization. After the user directory is brought online, CP tries to access any minidisks specified on the CP_ACCESS statements in the system configuration file. If no such statements exist, CP will not have any minidisks accessed. For example, to determine what minidisks (if any) CP has accessed, issue the following command:

```
query cpdisks
```

If no disks are accessed, you receive the following response:

```
No disks accessed.
Ready;
```

If a FEATURES statement in your system configuration file instructed CP to use log messages from files on CP-accessed minidisks, there would be no log message data because CP does not have any minidisks accessed. For example, to determine if any log message data exists, issue the following command:

```
query logmsg
```

And receive the response:

```
There is no logmsg data
Ready;
```

If CP had processed a logo configuration file at IPL time and the logo configuration file statements referred to logo picture files, an input area file, or an online message file, CP displays the standard system logo defined in HCPBOX ASSEMBLE on all terminals (because CP does not have any minidisks accessed).

Therefore, depending on information in your system and logo configuration files, CP may look for files on CP-accessed minidisks when it needs to find log message files or logos. If you had placed one or more CP_ACCESS statements in your system configuration file, the response to the QUERY CPDISKS command after an IPL might be similar to the following:

```
Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
PROD1  MAINT    0300  A   R/O  ESARES 0A0F ECKD       2400       2449
BACKUP MAINT    0301  B   R/O  ESARES 0A0F ECKD       2450       2499
Ready;
```

This response indicates that MAINT's 300 and 301 disks are accessed by CP. Unless you specify otherwise, CP accesses these disks in stable-read (SR) mode so that while CP only has read-only access to the minidisks, no other user can obtain a write link to the minidisk. In this case, if a user requests a copy of the log messages, CP reads files on the CP-accessed minidisks (MAINT's 300 and 301) and displays the contents of one or more files at the user's terminal. If a user switches on a terminal, CP can construct a logo from information contained in logo picture files, an input area file, and an online message file on the CP-accessed minidisks.

Changing the List of Disks Accessed by CP

As previously discussed, the CP_ACCESS statements in the system configuration file define the initial list of disks accessed by CP. After the system is up and running, you can change the list of disks that CP searches for log message and logo information without having to shut down and restart the system. The CPACCESS and CPRELEASE commands are similar to the CMS ACCESS and RELEASE commands in that they enable privileged users to modify the list of CP accessed disks. However, unlike the CMS ACCESS and RELEASE commands, the CPACCESS and CPRELEASE do not affect any operations in the virtual machine where the command was issued. Instead, they instruct CP to modify its list of accessed minidisks. For example, to release a disk accessed by CP, issue the following command:

```
cprelease a
```

You receive these responses:

```
CPRELEASE request for disk A scheduled.
Ready;
CPRELEASE request for disk A completed.
```

If you issue QUERY CPDISKS, you see that in this case, CP has only a single disk accessed, and all the logo and log message information have to be on this disk.

```
Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
BACKUP MAINT    0301  B   R/O  ESARES 0A0F ECKD       2450       2499
Ready;
```

You can also add to the list of minidisks accessed by CP. For example, to add MAINT's 400 disk to the list of disks accessed by CP, issue the following command:

```
cpaccess maint 400 a
```

You receive these responses:

```
CPACCESS request for mode A scheduled.
Ready;

CPACCESS request for MAINT's 0400 in mode A completed.
```

To determine which disks are now accessed, issue QUERY CPDISKS. You receive this response:

```
Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
PROD2  MAINT    0400  A   R/O  XAUSR8 0B12 CKD        2300       2349
BACKUP MAINT    0301  B   R/O  ESARES 0A0F ECKD       2450       2499
Ready;
```

You are only authorized to request that CP should access a minidisk if you have the authority to link to the minidisk yourself. To have CP access the minidisk in SR mode (the default), you must be authorized to link to the disk in SR mode yourself.

Whenever CP needs to display the log message or create a logo, it tries to read files from the minidisks that it currently has accessed.

Performance Considerations

Because CP tries to read files from CP-accessed minidisks whenever it displays log messages or logos, it may read certain files on the minidisks many times. Rather than reading these files from the minidisks each time, CP can cache the files in storage. Then, every time the files have to be read, CP does not have to wait for a copy of the files to be read from DASD because their contents are already in storage.

For example, to cache certain files that reside on CP-accessed minidisk, issue the following CPCACHE command:

```
cpcache sna% logms* a
2 file(s) cached.
Ready;
```

The response to the command indicates that 2 files were placed in storage as a result of the CPCACHE command. CPCACHE accepts the use of wildcard characters (% and *) to specify files that it should cache.

You can also give CP a list of files to read into storage whenever it accesses the minidisk. You do this by creating a file called CPCACHE FILES, which contains the list of files, and placing the file on the minidisk that CP accesses. This is an example of a CPCACHE FILES file:

```
*      logms*     /* Cache all log message related files */

sna%   logo       /* Cache all SNA logo picture files    */

online message    /* Cache the file that contains the
                     "VM/ESA Online" message          */
```

CP attempts to read CPCACHE FILES on any minidisk it accesses, whether the access attempt is the result of a CPACCESS command or CP_ACCESS statements found in the system configuration file.

CP removes the appropriate cached files from storage when a CPRELEASE or CPACCESS command is issued for the minidisk from which the files were loaded. If the files were loaded into storage as a result of the CPCACHE command (rather than a CPCACHE FILES file), you have to issue the CPCACHE command again when CP reaccesses the disk.

Displaying the Contents of CP-Accessed Minidisks

Just as you can use the CMS LISTFILE command to display what files are on accessed minidisks, privileged users can use the CP CPLISTFILE command to display what files are on CP-accessed minidisks. For example, to display the LOGO files found on a CP-accessed minidisk, issue the following CPLISTFILE command:

```
cplistfile * logo a
```

The response displays the file name and file type of the file, its format and size, the date it was last modified, and whether the file is currently cached:

```
Filename Filetype FM Fmt LRecL    Records   Date      Time   Cache
DEFAULT  LOGO     A  F      80         15 10/30/91 21:15:15 No
08E6     LOGO     A  F      80         23 02/20/92 08:12:56 No
LDEV     LOGO     A  F      80         15 11/30/91 21:14:50 No
LOCAL    LOGO     A  F      80         15 09/16/91 23:17:42 No
MOD5     LOGO     A  F     132         15 10/30/91 21:13:53 No
PVM      LOGO     A  F      78         15 10/30/91 21:15:02 No
SNA1     LOGO     A  F      78         15 11/29/91 12:32:35 Yes
SNA2     LOGO     A  F      78         15 10/30/91 21:14:35 Yes
SYSTEM   LOGO     A  F      80         15 10/30/91 21:14:29 No
TCPIP    LOGO     A  F      78         15 01/27/92 20:22:09 No
Ready;
```

To display certain statistics about files on CP-accessed minidisks, issue:

```
cplistfile sna* logo a statistics
```

The response looks similar to this:

```
Filename Filetype FM      Opens     Closes
SNA1     LOGO     A          27         26
SNA2     LOGO     A          57         56
Ready;
```

Finally, to display the contents of a file that resides on a CP-accessed minidisk, issue:

```
cptype cpcache files a
```

And, receive this response:

```
*      logms*     /* Cache all log message related files */

sna%   logo       /* Cache all SNA logo picture files    */

online message    /* Cache the file that contains the
                        "VM/ESA Online" message         */
Ready;
```

You are authorized to display the contents of files using the CPTYPE command only if you have the authority to link to the minidisk on which the files reside.

Changing Information on CP-Accessed Minidisks

If your system configuration file contains a FEATURES statement that instructs CP to construct log messages from information in files on CP-accessed minidisks, you can no longer use the SET LOGMSG command to alter the contents of the log message. Instead, you must do the following:

1. Issue a CPRELEASE command to have CP remove the minidisk from its list of accessed disks.
2. Issue the CP LINK command and CMS ACCESS command to link and access the target minidisk in write mode.
3. Edit the files that contain the log messages for your system.
4. Issue the CMS RELEASE command and the CP DETACH command to release the minidisk and detach it from your virtual machine.
5. Issue a CPACCESS command to have CP add the minidisk to its list of accessed disks.

Even though you could have CP maintain read-only access to the disk while you are editing the files, it is recommended that you remove the disk from CP's list of accessed disks first. When CP accesses a minidisk, it reads the minidisk's file directory into storage. Any subsequent changes to the minidisk are not reflected into the file directory CP keeps in storage until you issue a CPACCESS command to have CP reaccess the minidisk. If you change files on the minidisk while CP has access to it, users may receive incorrect output in response to commands such as QUERY LOGMSG.

On the other hand, you may encounter another problem if you tell CP to remove the minidisk from its list of accessed minidisks before you make your change. If the minidisk you are changing is the only one that contained log message information, and a user issued a QUERY LOGMSG command while you were in the process of changing the log message, CP would find no log message files and would respond that no log message data existed.

For example,

```
query cpdisks

Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
PROD1  MAINT    0300  A   R/O  ESARES 0A0F ECKD       2400       2449
Ready;

cprelease a
CPRELEASE request for disk A scheduled.
Ready;
CPRELEASE request for disk A completed.

query logmsg
There is no logmsg data
Ready;
```

It is recommended that to change the contents of minidisks that are accessed by CP, you should work with at least two disks. When you instruct CP to release one minidisk in order to change files on the minidisk, CP can still read information from the second minidisk.

For example:

```
query cpdisks

Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
PROD1  MAINT    0300  A   R/O  ESARES 0A0F ECKD       2400       2449
BACKUP MAINT    0301  B   R/O  ESARES 0A0F ECKD       2450       2499
Ready;

cprelease a
CPRELEASE request for disk A scheduled.
Ready;
CPRELEASE request for disk A completed.

link maint 300 300 wr
Ready;

access 300 a
Ready;

xedit system logmsg a
Ready;

release a (detach
Ready;

cpaccess maint 300 a
CPACCESS request for mode A scheduled.
Ready;
CPACCESS request for MAINT's 0300 in mode A completed.
```

You can now tell CP to release the B disk (MAINT's 301) and update it to contain the new SYSTEM LOGMSG file that you placed on MAINT's 300 disk.

You can update logo picture files, input area files, and online message files in a similar manner. CP uses the changed files as soon as the CPACCESS request for the minidisk completes.

Chapter 4. Defining I/O Devices

There are four methods that can be used to define I/O devices to CP during IPL:

- Allow CP to dynamically sense the devices
- Use RDEVICE statements or EDEVICE statements in the system configuration file
- Use RDEVICE macroinstructions in the HCPRIO ASSEMBLE file
- Use Hardware Configuration Manager (HCM) and Hardware Configuration Definition (HCD) to define the devices.

The first three methods can be used in combination with each other, but typically CP senses the devices and only those devices requiring additional definition will have an RDEVICE statement or an EDEVICE statement in the system configuration file. Generally, the HCPRIO ASSEMBLE file is used only in special circumstances, because updating the file requires rebuilding the CP module. It is recommended that the tables in this chapter be used to determine how, and if, a given device should be defined in the system configuration file. If a configuration file statement is listed as "not required", then no definition is required and VM will dynamically determine the device characteristics through device sensing. There may be cases where a configuration file statement, although "not required", should be used to specify additional options or features, or both. Devices not found in the tables are either unsupported or require an RDEVICE macroinstruction in HCPRIO.

To use HCM and HCD to define devices to CP, an IODF statement that specifies both an IODF and an *osconfig* name must be specified in the system configuration file. In this case, all devices are defined by HCD (except older and unsupported devices, which still can be defined in HCPRIO ASSEMBLE) and device sensing and RDEVICE statements or EDEVICE statements in the system configuration file cannot be used. For information about using HCM and HCD, see z/OS and z/VM: Hardware Configuration Manager User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/eequ100_v2r5.pdf) and *z/VM: I/O Configuration*.

Device Support

A real device can be supported or unsupported. A supported device can be supported for CP and guest use or only for dedicated use by a single guest.

A supported device is one of those listed in *z/VM: General Information* along with the type of support it receives. If that book does not list a device, assume the device is not supported. The use of listed devices is fully supported by IBM through z/VM service support.

A device supported for CP and guest use is one that CP and virtual machines can use. CP provides system services for the device, including error recovery for guest DIAGNOSE I/O requests and a full command set (that is, you can use all device-oriented CP commands for the device). Such a device can also be shared among multiple guests if appropriate (for example, DASD) or can be dedicated to the exclusive use of a single guest.

A device supported for dedicated-only use by a single guest can only be logically attached to a single guest virtual machine at any one time. The guest must be fully capable of running with the device. CP cannot use the device itself, and DIAGNOSE I/O services are not supplied to the guest.

Unsupported Devices

An unsupported device is any device not listed in *z/VM: General Information*. An unsupported device must be dedicated to a single guest, but proper operation with z/VM and the guest is the customer's responsibility; IBM does not guarantee that unsupported devices will run properly with z/VM, and service support for such device attachments is not supplied.

An unsupported device must be defined to z/VM as some unrecognizable device type (that is, a device type different from any of the supported IBM devices) and must be defined with RDEVICE TYPE

UNSUPPORTED statement in the system configuration file. By knowing the class of devices (DASD, tape, and so on) to which the unsupported device belongs, z/VM knows what kinds of unsupported devices are similar.

Note: DASD which is defined as being unsupported cannot be IPLed by CP. Unsupported DASD can only be used by a virtual machine which is IPLed either from a supported device or from a named saved system.

Device Sensing

Most IBM devices can be queried (sensed) to determine what type of device they are. This is done at IPL time and when a device is varied on. For example, assume you have an IBM DASD device at address 300. CP issues a sense ID command to device 300. The device returns information indicating what type of device it is.

CP may need more information than that returned by some devices when they are queried. An example of this type of device is the IBM 4248 printer. When queried, the printer returns only the information that it is a 4248. Necessary information such as the printer class and the forms control buffer to be used is not provided via sensing; if CP is not provided this information on an RDEVICE statement, CP uses default values for that type of printer.

If a device does not respond to the device sense ID command, then the device must be defined in the system configuration file using the RDEVICE statement.

For a complete description of the RDEVICE statement, see "RDEVICE Statement" on page 227.

DASD

Table 2 on page 28 indicates for DASD devices whether or not a statement is required in the system configuration file. For some devices, a statement is needed only if the device is shared, in which case the following statement can be used:

```
RDEVICE rdev TYPE DASD SHARED YES
```

Table 2. Configuration Guide for DASD

Device	Statement Needed in Configuration File
3380 Models A04, AA4, B04, AD4, BD4, AE4, BE4, AJ4, BJ4, AK4, BK4	• When not shared, not required • When shared, TYPE DASD YES
3380 Model CJ2	• When not shared, not required • When shared, TYPE DASD YES
3390 Models A14, A18, B14, B18, B1C, A24, A28, B24, B28, B2C, A34, B34, A38, B38, B3C	• When not shared, not required • When shared, TYPE DASD YES
3390 Models A98, B9C	• When not shared, not required • When shared, TYPE DASD YES
9336 Model 20	Not required

See "RDEVICE Statement (DASD)" on page 236 for a complete description of the RDEVICE statement.

Tapes

Table 3 on page 29 indicates for tape devices whether or not a statement is required in the system configuration file. The TYPE operand of the RDEVICE statement is required for those devices whose characteristics cannot be dynamically determined.

Table 3. Configuration Guide for Tape Drives

Device	**Statement Needed in Configuration File**
3480 All Models	Not required*
3490 All Models	Not required*
3495 Tape Library Dataserver	**
3590 All Models	Not required*

Notes:

*The 'TYPE TAPE' operand can be specified to provide query capability for tapes that are intentionally left offline at IPL.

**See the *DFSMS/VM Installation and Customization Guide* for information on defining the 3495 Tape Library Dataserver for VM use.

See "RDEVICE Statement (Tape Units)" on page 249 for a complete description of the RDEVICE statement.

Printers

Table 4 on page 29 indicates for printer devices whether or not a statement is required in the system configuration file. The TYPE operand of the RDEVICE statement is required for those devices whose characteristics cannot be dynamically determined.

Table 4. Configuration Guide for Printers

Device	**Statement Needed in Configuration File**
3203 Model 5	TYPE 3203
3211 Models 1, 5	TYPE 3211 (For devices, such as a 4248, that emulate a 3211)
3262	TYPE IMPACT_PRINTER
3268 Models 2, 2C	Not required
3287 Models 1, 1C, 2, 2C, 4	Not required
3289 Models 1, 3, 4, 8	TYPE 3289
3800 Model 1	TYPE 3800
3800 Models 3, 6, 8	TYPE Advance Function Printing (AFP) or TYPE 3800
3812	• When coax attached, not required • When BSC attached, TYPE BSC_ADAPTER • When SDLC (VTAM attached), not required

Table 4. Configuration Guide for Printers (continued)

Device	Statement Needed in Configuration File
3816	• When coax attached, not required • When BSC attached, TYPE BSC_ADAPTER • When SDLC (VTAM attached), not required
3820	• When channel attached for use as a PSF printer, TYPE AFP • When VTAM attached, TYPE 3705
3825	TYPE AFP
3827	TYPE AFP
3828	TYPE AFP
3835	TYPE AFP
3900	TYPE AFP
4245 Models 1, 12, 20	TYPE IMPACT_PRINTER
4248 Models 1, 2	TYPE IMPACT_PRINTER
6262 Models 14, 22	TYPE IMPACT_PRINTER

For BSC attached 3812 or 3816 printers, see "RDEVICE Statement (Communication Controllers)" on page 234 for a complete description of the appropriate RDEVICE statement. For the other printers, see "RDEVICE Statement (Advanced Function Printers)" on page 228 and "RDEVICE Statement (Impact Printers)" on page 243 for a complete description.

Unit Record Devices

Table 5 on page 30 indicates for reader and punch devices whether or not a statement is required in the system configuration file. The TYPE operand of the RDEVICE statement is required for those devices whose characteristics cannot be dynamically determined.

Table 5. Configuration Guide for Reader and Punch Devices

Device	Statement Needed in Configuration File
3505 Models B1, B2	TYPE READER
3525 Models P1, P2, P3	TYPE PUNCH

See "RDEVICE Statement (Card Readers)" on page 232 and "RDEVICE Statement (Card Punches)" on page 230 for a complete description of the appropriate RDEVICE statement.

Displays

Display devices can be defined in the system configuration file. IBM 3277 displays have 3277 defined on the TYPE operand of the RDEVICE statement. All other non-3270 displays can be sensed and no configuration file statement is needed unless the device has special features, such as the Operator Identification Card. If a display has a special feature, an RDEVICE statement for the display is needed. For example, if a display has an Operator Identification Card, the following RDEVICE statement should be included in the system configuration file:

```
RDEVICE rdev TYPE 3270_DISPLAY OPER_IDENT_READER YES
```

The operand *rdev* is the real device number. For a complete description this statement, see "RDEVICE Statement (Graphic Display Devices)" on page 239.

Communication Controllers

Table 6 on page 31 indicates for communication controller devices whether or not a statement is required in the system configuration file. The TYPE operand of the RDEVICE statement is required for those devices whose characteristics cannot be dynamically determined.

Table 6. Configuration Guide for Communication Controllers

Device	Statement Needed in Configuration File
3745	• When running NCP, not required • When running PEP for ASCII lines, TYPE TELE2_ADAPTER

See "RDEVICE Statement (Communication Controllers)" on page 234 for a complete description of the RDEVICE statement.

ESCON Devices

These ESCON devices can be sensed by CP and do not require a system configuration file statement:

- 9032 ESCON Director Model 2
- 9033 ESCON Director Model 1
- 9034 ESCON Converter Model 1
- 9035 ESCON Converter Model 2
- ESCON CTCA

For information on defining ESCON devices that cannot be sensed by CP, see "RDEVICE Statement (Unsupported Devices)" on page 252 in the description of the RDEVICE statement in Chapter 6, "The System Configuration File," on page 47. For information on defining ESCON-attached devices, see the appropriate device tables in this chapter.

Other Devices

Table 7 on page 31 indicates for various devices whether or not a statement is required in the system configuration file. The TYPE operand of the RDEVICE statement is required for those devices whose characteristics cannot be dynamically determined.

Table 7. Configuration Guide for Other Devices

Device	Statement Needed in Configuration File
3088	Not required
3423 Optical Media Attachment	TYPE UNSUPPORTED
3737 Remote CTCA	TYPE CTCA
3890 Document Processor	Not required
4753 Network Security Processor	Not required (looks like a 3088)
7171	TYPE 3270_DISPLAY
CTCA	TYPE CTCA

Table 7. Configuration Guide for Other Devices (continued)

Device	**Statement Needed in Configuration File**
OSA	Not required

Chapter 5. Crypto Planning and Management

IBM Z servers provide two facilities that enable application programs and operating systems to access the cryptographic capabilities of the server:

1. Central Processor Assist for Cryptographic Function (CPACF)
2. IBM Crypto Express Features

CPACF

CPACF is a set of z/Architecture instructions provided by the Message Security Assist (MSA) facility and its extensions. It provides symmetric cryptographic functions using clear and protected keys. No additional hardware is required, though the CPACF Enablement feature (code 3863) must be installed for full function.

Crypto Express Features

Crypto Express features are tamper sensing and responding hardware security modules (HSMs) that perform advanced symmetric and asymmetric cryptographic operations, and provide the ability to secure the encryption keys.

Use of the Crypto Express feature requires the CPACF Enablement feature (code 3863) to be installed as a pre-requisite.

Each Crypto Express feature has one or more crypto *adapters*. The Hardware Management Console (HMC) refers to these adapters as "Cryptos". CP refers to them as "APs". Each crypto adapter has multiple *domains*. CP refers to a single domain on an adapter as a crypto "resource". Thus, a crypto resource is identified by its AP number and domain number.

Configuring Crypto Express Adapters

Crypto Express adapters can be configured in one of the following modes:

- Accelerator - Clear key functions only
- IBM Common Cryptographic Architecture (CCA) Coprocessor - Clear key and secure key functions
- IBM Enterprise Public-Key Cryptographic Standards (PKCS) #11 Coprocessor - Secure key functions through a PKCS#11 API only. This configuration mode is also known as EP11 Coprocessor or XCP Coprocessor.

Crypto resources on an adapter that is configured in accelerator or CCA coprocessor mode can be used by z/VM virtual machines as dedicated or shared crypto resources.

Crypto resources on an adapter that is configured in EP11 coprocessor mode can be used by z/VM virtual machines only as dedicated crypto resources.

By default, Crypto Express adapters are configured as CCA coprocessors. The HMC can be used to reconfigure an adapter as an accelerator or EP11 coprocessor.

Crypto Domain Access

There are two types of crypto domain access:

- Usage domains - For usage of cryptographic functions.
- Control domains - For management of the domains, which includes the management of master keys.

z/VM recognizes only those crypto resources for which the z/VM LPAR has both usage and control access.

Assigning Crypto Express Resources to z/VM

For z/VM to access Crypto Express resources, they must be assigned to the LPAR that is running the z/VM system. This assignment and some necessary configuration of the crypto adapters can be done during the process of building an activation profile for the z/VM LPAR from the HMC.

To do this, use the *Customize/Delete Activation Profiles Task* on the HMC:

- Go to the Crypto section of the Image Profile.
- Assign one or more domains with "Control and Usage" access.
- Assign one or more cryptos specifying "Candidate and Online". This causes the adapter to be brought online when the LPAR is activated.

After the LPAR is activated by using this Image Profile, the crypto resources will be available to CP when z/VM is IPLed.

Assignment of Crypto Resources on a z/VM System

Each crypto resource that is available to CP is assigned to one of three categories.

1. SHARED - This resource is assigned to the shared pool of crypto resources on the system and will service requests from multiple virtual machines. Shared crypto resources support only crypto operations that do not rely on domain state, such as clear-key encryption, random number generation, and digital signature operations.

   Only crypto resources that are configured as an accelerator or CCA coprocessor can be included in the shared pool. Crypto resources configured as an EP11 coprocessor cannot be added to the shared pool.

   All crypto resources in the shared pool must be the same type and mode. For example, if the first resource that is assigned to the shared pool is a Crypto Express6 in accelerator mode, then all additional resources that are assigned to the shared pool must also be a Crypto Express6 configured in accelerator mode.

   Virtual machines that have access to the shared pool of crypto resources are referred to as "APVIRT crypto" virtual machines.
2. DEDICATED - This resource is assigned to only one virtual machine for its exclusive use. Crypto resources that are configured as an accelerator, CCA coprocessor, or EP11 coprocessor can be dedicated to a virtual machine.

   Virtual machines that have dedicated crypto resources that are assigned are referred to as "APDED crypto" virtual machines.
3. FREE - This resource is not designated for any particular use. It is available to be assigned to the shared pool or to a virtual machine for its dedicated use.

A virtual machine can have access to the shared pool (APVIRT), or it can have dedicated resources (APDED), but not both.

CP identifies the assignment of a crypto resource by using one of the following values:

shared
: indicates that the crypto resource is attached to the system for shared use.

attached to *userid*
: indicates that the crypto resource is dedicated to a logged on virtual machine.

free
: indicates that the crypto resource is not in use.

free, dedication planned
: indicates that the crypto resource is not in use, however, it has been specified on a CRYPTO APDED statement in the online user directory.

See QUERY CRYPTO in *z/VM: CP Commands and Utilities Reference*.

Determining the Shared Pool of Crypto Resources at System IPL

A crypto resource is assigned to the shared pool if it is specified on the CRYPTO APVIRTUAL statement in the system configuration file. See the "CRYPTO APVIRTUAL Statement" on page 80, for more details.

If no resources are identified on a CRYPTO APVIRTUAL statement in the system configuration file, z/VM selects up to two crypto resources to include in the shared pool. If possible, z/VM selects resources on two different adapters of the same type and mode, which enables crypto work that is assigned to the shared pool to continue even if one of the adapters is taken offline.

Changing the Shared Pool of Crypto Resources after System IPL

After z/VM system IPL, crypto resources can be removed from the shared pool by using the DETACH CRYPTO FROM SYSTEM command. Resources can be added to the shared pool by using the ATTACH CRYPTO TO SYSTEM command. Both AP(s) and Domain(s) must be specified when issuing ATTACH CRYPTO TO SYSTEM or DETACH CRYPTO FROM SYSTEM. See *z/VM: CP Commands and Utilities Reference*, for details on these commands.

Changing the Type of Crypto Resources in the Shared Pool

Once a shared pool is established, all shared crypto resources must be detached from the system before a crypto resource of a different type or mode can be added to the shared pool.

Attention: Removing all resources from the shared crypto pool might cause application errors if APVIRT virtual machines are actively performing crypto work.

Accessing Shared Crypto Resources

A CRYPTO APVIRTUAL statement must be included in a virtual machine's directory entry to enable access to the shared crypto resources. With this statement in the user directory, the virtual machine is given access to the shared crypto resources at logon. The virtual crypto resource providing access to the shared pool appears to a virtual machine as AP 1, domain 1. When an application running in an APVIRT virtual machine performs a crypto operation, z/VM determines the best real crypto resource to queue the work on, balancing the load among all the shared resources. See "CRYPTO Directory Statement" on page 490, for more details.

While a virtual machine is logged on, access to the shared crypto resources can be removed by using the DETACH CRYPTO APVIRT command. Access can be restored by using the DEFINE CRYPTO APVIRT command. See *z/VM: CP Commands and Utilities Reference*, for details on these commands.

Assigning Dedicated Crypto Resources to a Virtual Machine

Dedicated crypto resources can be assigned to a virtual machine by including a CRYPTO APDEDICATED statement in the user directory entry. Crypto resources specified in the user directory are assigned at logon. For a virtual machine to gain dedicated access to crypto resources, the specified domains on each specified AP must be FREE as shown by QUERY CRYPTO DOMAINS. If one or more domains on an AP are not FREE, then none of the domains on that AP are attached to the virtual machine. For details, see "CRYPTO Directory Statement" on page 490.

After a virtual machine is logged on, dedicated crypto resources can be assigned by using the ATTACH CRYPTO TO *userid* command. This attachment lasts for the logon session. In order to have this attachment occur at each logon, add a CRYPTO APDEDICATED statement to the user directory.

A virtual machine's dedicated crypto resources can be removed by using the DETACH CRYPTO FROM *userid* command.

An operating system running in an APDED virtual machine generally expects to work with a single domain, across multiple APs as needed for bandwidth and redundancy. The same master key is installed in that domain across all the assigned APs.

In order for a virtual machine to have dedicated access to a crypto resource, each specified domain number must be available for dedicated use by this virtual machine on all specified AP numbers. If one or more domains on an AP are not available for dedicated use by this virtual machine, then none of the domains on the AP are assigned. Any requested domains that are not assigned to the LPAR that the z/VM system is running on are eliminated from the request and will not be reason to deny access to the remaining requested APs or domains.

For example, assume that domain 1 on APs 1, 2, and 3 are resources that are dedicated to virtual machine vmach1. You can then attach domain 2 to vmach1 only if domain 2 is available to vmach1 on all of APs 1, 2, and 3. If domain 2 is not available on one or more of APs 1, 2, or 3, then you cannot attach domain 2 to vmach1 on any of APs 1, 2, or 3.

After the first set of crypto resources have been dedicated to the virtual machine, the ATTACH CRYPTO and DETACH CRYPTO commands can specify either APs or domains, not both, to be added or removed. See ATTACH and DETACH in *z/VM: CP Commands and Utilities Reference*.

Changing the Crypto Attachment Mode of a Virtual Machine

At logon time, the CRYPTO statement in a virtual machine's user directory determines whether the virtual machine is an APVIRT or APDED virtual machine. After logon, CP commands (in combination with changes to the user directory) can be used to switch a virtual machine between APVIRT and APDED.

To transition from APVIRT to APDED, first detach the APVIRT crypto resource from a virtual machine and then attach the selected dedicated crypto resources. To make this change permanent, replace the CRYPTO APVIRT statement in the user directory with a CRYPTO APDEDICATED statement specifying the selected crypto resources.

To transition from APDED to APVIRT, first replace any CRYPTO APDEDICATED statements in the virtual machine's directory entry with a CRYPTO APVIRTUAL statement. Next, detach all dedicated crypto resources from a virtual machine. Finally, use the DEFINE CRYPTO APVIRTUAL command to gain access to the shared pool of crypto resources.

⚠ **Attention:** Changing the crypto attachment mode while the guest is running might result in errors from any program actively using crypto resources.

Configuring Crypto Adapters ONLINE and OFFLINE

The VARY ONLINE CRYPTO and VARY OFFLINE CRYPTO commands can be used to change the availability of crypto adapters that are assigned to the z/VM LPAR. Take a crypto adapter offline when necessary to service the adapter or to make z/VM's domains on it available to be reassigned to a different LPAR. Bring a crypto adapter online after service has been performed or when a new adapter has been assigned to the z/VM LPAR. See *z/VM: CP Commands and Utilities Reference*, for details on these commands.

Querying the Status of Crypto Resources on a z/VM System

QUERY CRYPTO DOMAINS displays the status of all crypto resources that are assigned to the z/VM LPAR.

```
q crypto domains
AP 000 CEX6A  Domain 010   operational    online    shared
AP 000 CEX6A  Domain 084   operational    online    free, dedication planned
AP 001 CEX6A  Domain 010   resetting      online    attached to TRACYK
AP 001 CEX6A  Domain 084   operational    online    free, dedication planned
```

QUERY VIRTUAL CRYPTO displays the status of crypto resources available to the virtual machine.

```
q v crypto (from an APVIRT user)
AP 001 CEX7A  Domain 001   shared         online
```

```
q v crypto (from an APDED user)
AP 001 CEX6A  Domain 010   dedicated      online
AP 002 CEX6C  Domain 010   dedicated      offline
```

Tracking Crypto Status Changes

Crypto resources can transition among various states which are reported as *device status* and *configuration state* by the QUERY CRYPTO DOMAINS command. Refer to the QUERY CRYPTO command in *z/VM: CP Commands and Utilities Reference* for a list and description of device statuses and configuration states.

Resources are normally in *operational* state. A resource might briefly be *busy* or *resetting*, but is expected to return to *operational* state. Resources that are *deconfigured*, *checkstopped*, or *revoked* require some form of manual intervention to address the issue.

CP updates the device status reported by the QUERY CRYPTO command every 33 seconds. Consequently, the status presented by the QUERY CRYPTO command might not reflect the current status of the resource.

Dedicated Crypto Resource Management

Dedicated resources are managed and used directly by the virtual machine to which they are attached. CP's only involvement with dedicated resources is in configuration (VARY ONLINE and VARY OFFLINE) and assignment (ATTACH and DETACH).

The guest is responsible for recovery and retry of failed crypto operations. For this reason, it is recommended that at least two crypto resources on different adapters be dedicated to the virtual machine. Use the guest system's crypto commands to interrogate the state of any resources of concern.

Shared Crypto Resource Management

APVIRT crypto virtual machines are each presented with a single simulated crypto resource, which is backed by a pool of real resources managed by CP. When there are multiple resources assigned to the shared pool, CP can perform the following management tasks:

1. Load-balance work across all resources in the shared pool.
2. Divert work to other shared resources if a shared resource enters a non-working state.
3. Add resources to and remove resources from the shared pool dynamically and non-disruptively, in response to operator commands. This allows resources to be selectively taken out of use for service and maintenance.

The FORCE option on the DETACH CRYPTO FROM SYSTEM command is required to remove the last crypto resource from the shared pool. In this case. the QUERY VIRTUAL CRYPTO command reports the configuration state of the simulated crypto resource as *unavailable*.

If all resources assigned to the shared pool go into a permanent error state, the QUERY VIRTUAL CRYPTO command reports the configuration state of the simulated resource to the user as an *error*.

When the configuration state of the shared pool transitions to *unavailable* or *error*, work that was in progress is discarded and new work is rejected. Messages are issued to the CP operator when this happens.

Live Guest Relocation of APVIRT Virtual Machines

APVIRT crypto virtual machines can be freely relocated to another member system in a relocation domain under certain conditions.

The first condition is that the shared pools of the source and destination members are the same CEX adapter mode (accelerator mode or CCA coprocessor mode).

In addition, one or both of the following conditions must be true.

- The shared pools of the source and destination members are the same CEX adapter type.
- A destination system has the Live Guest Relocation (LGR) for mixed APVIRT facility and is configured with a compatible shared crypto pool. A shared crypto pool might be incompatible in the case of a deprecated CEX adapter or an architectural level-set.

The LGR for mixed APVIRT facility is available in z/VM 7.2 (APAR VM66496) and later releases.

The LGR for mixed APVIRT facility provides the following benefits:

- Crypto Express (CEX) adapters and their capabilities can be managed by relocation domains in much the same way that CPU facilities are managed.
- APVIRT virtual machines have more relocation opportunities when an updated system with a new CEX adapter type is introduced to the SSI.

Note: Relocation of an APVIRT user from a system with LGR for mixed APVIRT to a system without LGR for mixed APVIRT is permitted only if the virtual CEX capabilities of the source and destination systems are identical.

The following topics provide examples of using LGR in environments with mixed crypto resource types.

Migration to an Updated System that Has the LGR for Mixed APVIRT Facility and a New Crypto Type

APVIRT virtual machines can be relocated from older systems without the LGR for mixed APVIRT facility to systems that have the LGR for mixed APVIRT facility. Such a relocation is permitted only if the destination system supports the user's current APVIRT capabilities. If the destination system supports the user's current APVIRT capabilities, the outbound relocation is permitted and the return relocation is also permitted. You can use the VMRELOCATE TEST command to test whether the virtual machine is eligible to be relocated to the specified system. See VMRELOCATE.

Figure 1 on page 38 illustrates a scenario for moving a workload from an older system to an upgraded system that has the LGR for mixed APVIRT facility.

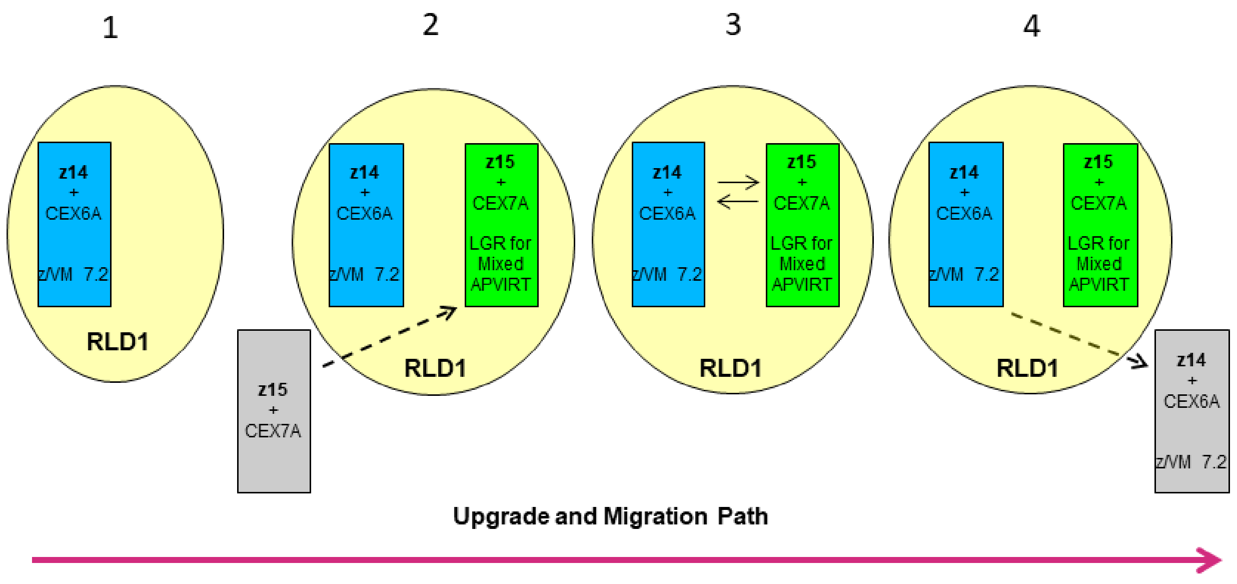


Figure 1. Using relocation to migrate a workload to an upgraded system.

In this scenario a z15 system with a CEX7A crypto adapter and the LGR for mixed APVIRT facility is added to a relocation domain that includes a z14 system with a CEX6A crypto adapter. The z14 system does not have the LGR for mixed APVIRT facility. The virtual machine can be freely relocated from the z14 system to the z15 system. Relocation of the virtual machine back to the z14 system is also permitted.

The workload migration involves the following steps. Each numbered step corresponds to a numbered image in Figure 1 on page 38:

1. The relocation domain RLD1 contains only a z14 system, which does not have the LGR for mixed APVIRT facility.

2. Prepare the relocation domain for workload migration:
   - On the z15, load an image that contains the LGR for mixed APVIRT facility.
   - Update relocation domain definitions to include the z15 system.
   - On the z14, identify one or more virtual machines that run crypto workloads that you want to migrate to the new z15 system.
3. Use LGR to migrate the selected set of virtual machines from the z14 system to the z15 system.

   LGR from the z15 system back to the z14 system is permitted. The guest retains its virtual AP capabilities from the originating system. If the originating system is not reconfigured, then the same pool capabilities apply and are identical to the guest's virtual characteristics when the decision is made to relocate back.
4. After you verify that the migration is successful you can consider the following actions:
   - Migrate more work from the z14 system to the z15 system.
   - Decommission the z14 system.

Relocation Subdomains in an Environment with Mixed Crypto Adapter Types

A relocation subdomain is the set of members of a relocation domain that have the same CEX adapter mode.

APVIRT virtual machines that are assigned to a relocation domain where the member systems are running LGR for mixed APVIRT see a virtual CEX adapter with the following capabilities:

- The virtual CEX adapter has capabilities that are common to all the members of that relocation domain that run shared pools of the same CEX mode. The CEX mode can be accelerator or CCA coprocessor mode.
- The virtual CEX adapter presents the maximal common subset of capabilities for that CEX mode.

Figure 2 on page 40 illustrates an example of relocation subdomain when all systems have the LGR for mixed APVIRT facility.

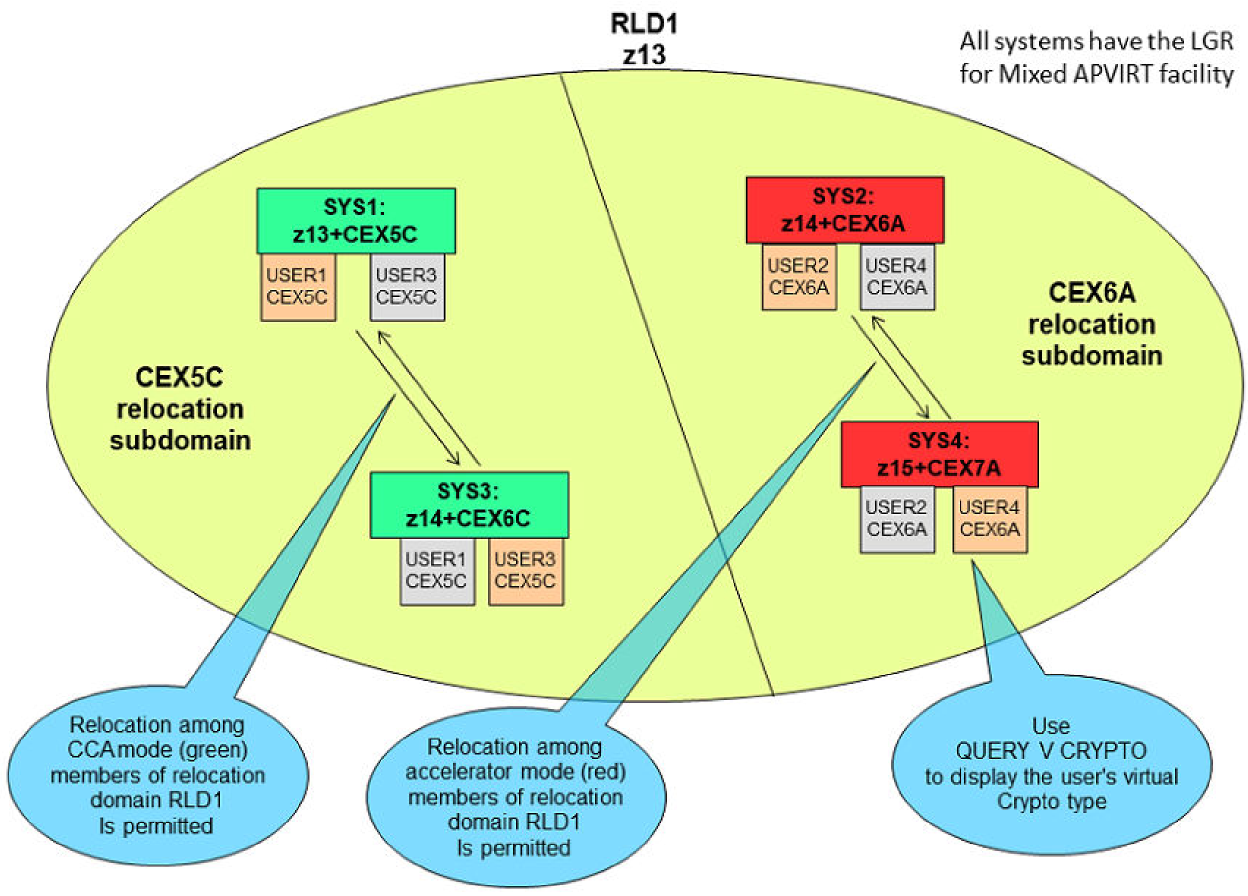

RLD1
z13
All systems have the LGR for Mixed APVIRT facility
SYS1: z13+CEX5C
USER1 CEX5C
USER3 CEX5C
CEX5C relocation subdomain
SYS3: z14+CEX6C
USER1 CEX5C
USER3 CEX5C
SYS2: z14+CEX6A
USER2 CEX6A
USER4 CEX6A
CEX6A relocation subdomain
SYS4: z15+CEX7A
USER2 CEX6A
USER4 CEX6A
Relocation among CCAmode (green) members of relocation domain RLD1 Is permitted
Relocation among accelerator mode (red) members of relocation domain RLD1 Is permitted
Use QUERY V CRYPTO to display the user's virtual Crypto type

Figure 2. A relocation domain for APVIRT virtual machines that is partitioned into relocation subdomains

This example shows a relocation domain with the following characteristics:

- All systems have the LGR for mixed APVIRT facility.
- From the perspective of an APVIRT virtual machine, the relocation domain is partitioned into an accelerator relocation subdomain and a CCA-coprocessor relocation subdomain.
- Relocation domain RLD1 is a virtual z13 system because z13 is the maximal common system level of the processors in the relocation subdomain.
- USER1 and USER3 are in the virtual CEX5C relocation subdomain and can be freely relocated between systems SYS1 and SYS3. The maximal common CEX type is CEX5. The outbound relocation is permitted and the return relocation is also permitted. Because all systems have the LGR for mixed APVIRT facility, the initial relocation is permitted from SYS1 or SYS3. USER1 and USER3 cannot be relocated to systems SYS2 or SYS4 because the shared pools don't have the same AP mode.
- USER2 and USER4 are in the virtual CEX6A relocation subdomain and can be freely relocated between systems SYS2 and SYS4. The outbound relocation is permitted and the return relocation is also permitted. Because all systems have the LGR for mixed APVIRT facility, the initial relocation is permitted from SYS2 or SYS4. USER2 and USER4 cannot be relocated to systems SYS1 or SYS3 because the shared pools don't have the same AP mode.

The CEX capabilities of the APVIRT users in a relocation domain can be affected by the use of dynamic crypto commands or by adding or removing members in a relocation domain. No APVIRT user is forcibly downgraded by the relocation domain calculations except when the downgrade is applied to the system on which the user is running. This situation is analogous to the situation for CPU facilities when relocation domains are reconfigured.

Multiple Relocation Domains in an Environment with Mixed Crypto Adapter Types

SSI permits systems to be members of more than one relocation domain. Where a system is a member of more than one relocation domain and each relocation domain supports different shared pool capabilities, the virtual machines on the common system might see different virtual CEX capabilities. A virtual machine sees capabilities that are defined by their assigned relocation domain.

Figure 3 on page 41 provides an example of a system that is a member of two relocation domains.

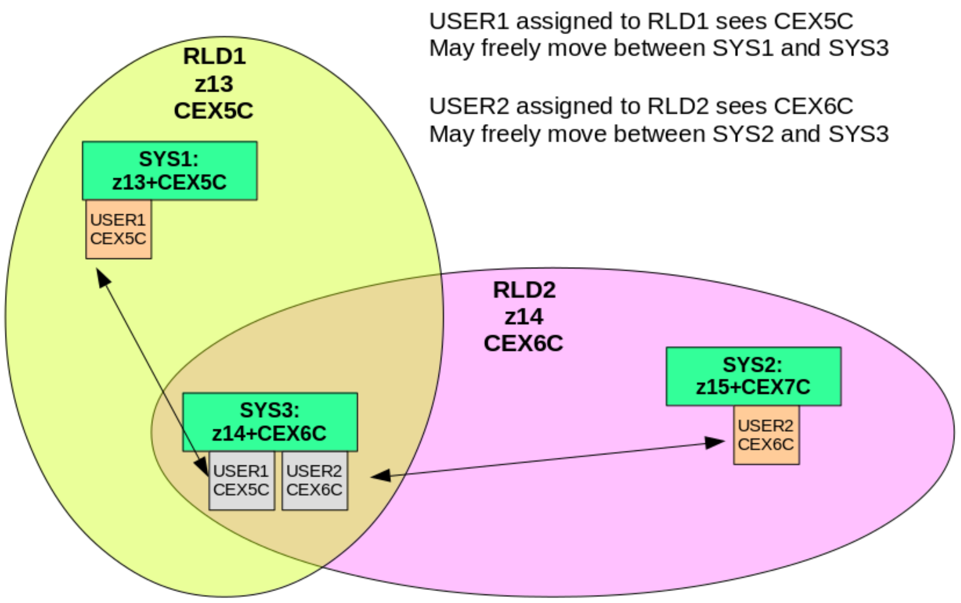

USER1 assigned to RLD1 sees CEX5C
May freely move between SYS1 and SYS3
USER2 assigned to RLD2 sees CEX6C
May freely move between SYS2 and SYS3
RLD1 z13 CEX5C
SYS1: z13+CEX5C
USER1 CEX5C
RLD2 z14 CEX6C
SYS2: z15+CEX7C
USER2 CEX6C
SYS3: z14+CEX6C
USER1 CEX5C
USER2 CEX6C

Figure 3. One system in two relocation domains

All systems in the example are assumed to have the LGR for mixed APVIRT facility.

USER1 is assigned to relocation domain RLD1 and sees virtual CEX5C adapters. USER1 can be freely relocated between systems SYS1 and SYS3.

USER2 is assigned to relocation domain RLD2 and sees virtual CEX6C adapters. USER2 can be freely relocated between systems SYS2 and SYS3.

Running Second Level z/VM Systems

Crypto Express adapters can be used by second-level z/VM systems.

z/VM supports the following second-level environments:

- APDED host with APVIRT guest
- APDED host with APDED guest
- APVIRT host with APVIRT guest

Manipulation of the physical crypto resources must be done from the first-level system. The second-level system is only able to VARY ONLINE or VARY OFFLINE crypto resources logically. This could result in a state where the resource is logically offline but still physically configured online. To prevent the resource from being used by an APDED second-level guest, it should be detached from the second-level guest.

Applying Maintenance to a Crypto Express Adapter

Follow the recommended process for the update you want to apply. Some firmware updates can be applied concurrently; others require the adapter to be put into standby state. If an update is applied concurrently, it might not become active until the AP is removed from the LPAR and reassigned back.

Refer to the documentation for applying each individual update to determine the steps that are needed to install and activate a firmware update.

If you need to place an AP into standby state or reconfigure the adapter at the HMC to activate maintenance, we recommend you first issue the VARY OFFLINE CRYPTO command to make the adapter unavailable for use by CP or its guests. There is no need to DETACH resources before using the VARY OFFLINE CRYPTO command. CP remembers the attachment settings and the resources become usable per their assignments (shared or dedicated use) when the VARY ONLINE CRYPTO command is used to make the adapter usable again.

Examples

Let’s begin with a z/VM LPAR that has the following crypto resources assigned.

```
AP 010 CEX7A      Domains 00  01  08  09  75  76
AP 011 CEX7C      Domains 00  01  08  09  75  76
AP 012 CEX7C      Domains 00  01  08  09  75  76
AP 013 CEX7P      Domains 00  01  08  09  75  76
AP 014 CEX6P      Domains 00  01  08  09  75  76
AP 015 CEX6A      Domains 00  01  08  09  75  76
AP 016 CEX6A      Domains 00  01  08  09  75  76
```

Assignment of Shared Crypto Resources

If shared crypto resources are not specified with a CRYPTO APVIRTUAL statement in the system configuration file, then after z/VM IPL the following crypto resource assignments exist:

	DOM 000	DOM 001	DOM 008	DOM 009	DOM 075	DOM 076
AP 010 CEX7A	SHARED	SHARED				
AP 011 CEX7C						
AP 012 CEX7C						
AP 013 CEX7P						
AP 014 CEX6P						
AP 015 CEX6A						
AP 016 CEX6A						

CP selected two crypto resources to include in the shared pool. This is not a recommended configuration because all shared resources are on the same adapter. This does not allow for continuous operation of work for APVIRT virtual machines if adapter 010 is taken offline or becomes unable to process work.

To ensure that shared resources are available on more than one adapter, the resources can be assigned by adding the following statement to the system configuration file prior to IPLing z/VM.

```
CRYPTO APVIRTUAL AP 015 016 DOMAIN 000
```

The shared crypto resources can also be reassigned after system IPL using CP commands. All shared resources must be on adapters with matching type and mode. There is only one CEX7A adapter on this system, so we remove all shared resources of this type and use the two CEX6A adapters for processing APVIRT crypto work.

```
detach crypto ap 010 dom 000 001 from system force
```

```
attach crypto ap 015 016 domain 000 to system
```

After reassigning shared crypto resources, the following crypto assignments will exist.

	DOM 000	DOM 001	DOM 008	DOM 009	DOM 075	DOM 076
AP 010 CEX7A						

	DOM 000	DOM 001	DOM 008	DOM 009	DOM 075	DOM 076
AP 011 CEX7C						
AP 012 CEX7C						
AP 013 CEX7P						
AP 014 CEX6P						
AP 015 CEX6A	SHARED					
AP 016 CEX6A	SHARED					

User Directory Crypto Statements

The user directory can contain CRYPTO statements to indicate which crypto resources a virtual machine can access. A CRYPTO APVIRTUAL statement in a virtual machine's directory entry allows the user to access the system's pool of shared crypto resources at logon. One or more CRYPTO APDEDICATED statements in a virtual machine's directory entry specify crypto resources that CP attempts to assign to the user for exclusive use at logon. A crypto resource must be free (not shared and not dedicated to a different user) for CP to successfully assign it for dedicated use.

To give ALAN access to shared crypto resources include the following statement in ALAN's user directory entry.

```
CRYPTO APVIRTUAL
```

To assign dedicated crypto resources to TRACY, you might include the following statement in TRACY's directory entry.

```
CRYPTO DOMAIN 000 001 008 APDEDICATED 010 011 012
```

To assign dedicated crypto resources to RICHARD, you might include the following statement in RICHARD's directory entry.

```
CRYPTO DOMAIN 008 009 076 APDEDICATED 011 012 013 014
```

Notice that some crypto resources are specified on more than one CRYPTO APDEDICATED statement in the user directory. This is not recommended. When this is the case, the first user to log on is given exclusive use of the resource. No other user can have access to the resource until the first user logs off.

After TRACY logs on the following resource assignments exist.

	DOM 000	DOM 001	DOM 008	DOM 009	DOM 075	DOM 076
AP 010 CEX7A	TRACY	TRACY	TRACY			
AP 011 CEX7C	TRACY	TRACY	TRACY			
AP 012 CEX7C	TRACY	TRACY	TRACY			
AP 013 CEX7P						
AP 014 CEX6P						
AP 015 CEX6A	SHARED					
AP 016 CEX6A	SHARED					

If RICHARD logs on next, the following resource assignments will exist. Some of the requested domains on APs 11 and 12 were not available, so no resources on these APs were dedicated to RICHARD at logon.

	DOM 000	DOM 001	DOM 008	DOM 009	DOM 075	DOM 076
AP 010 CEX7A	TRACY	TRACY	TRACY			
AP 011 CEX7C	TRACY	TRACY	TRACY			
AP 012 CEX7C	TRACY	TRACY	TRACY			

	DOM 000	DOM 001	DOM 008	DOM 009	DOM 075	DOM 076
AP 013 CEX7P			RICHARD	RICHARD		RICHARD
AP 014 CEX6P			RICHARD	RICHARD		RICHARD
AP 015 CEX6A	SHARED					
AP 016 CEX6A	SHARED					

When ALAN logs on, he is given access to shared crypto resources. The status and assignment of all crypto resources along with a list of APVIRT users are reported by QUERY CRYPTO DOMAINS USERS.

```
q crypto domains users
AP 010 CEX7A  Domain 000   operational    online    attached to TRACY
AP 010 CEX7A  Domain 001   operational    online    attached to TRACY
AP 010 CEX7A  Domain 008   operational    online    attached to TRACY
AP 010 CEX7A  Domain 009   operational    online    free
AP 010 CEX7A  Domain 075   operational    online    free
AP 010 CEX7A  Domain 076   operational    online    free
AP 011 CEX7C  Domain 000   operational    online    attached to TRACY
AP 011 CEX7C  Domain 001   operational    online    attached to TRACY
AP 011 CEX7C  Domain 008   operational    online    attached to TRACY
AP 011 CEX7C  Domain 009   operational    online    free, dedication planned
AP 011 CEX7C  Domain 075   operational    online    free
AP 011 CEX7C  Domain 076   operational    online    free, dedication planned
AP 012 CEX7A  Domain 000   operational    online    attached to TRACY
AP 012 CEX7A  Domain 001   operational    online    attached to TRACY
AP 012 CEX7A  Domain 008   operational    online    attached to TRACY
AP 012 CEX7A  Domain 009   operational    online    free, dedication planned
AP 012 CEX7A  Domain 075   operational    online    free
AP 012 CEX7A  Domain 076   operational    online    free, dedication planned
AP 013 CEX7P  Domain 000   operational    online    free
AP 013 CEX7P  Domain 001   operational    online    free
AP 013 CEX7P  Domain 008   operational    online    attached to RICHARD
AP 013 CEX7P  Domain 009   operational    online    attached to RICHARD
AP 013 CEX7P  Domain 075   operational    online    free
AP 013 CEX7P  Domain 076   operational    online    attached to RICHARD
AP 014 CEX6P  Domain 000   operational    online    free
AP 014 CEX6P  Domain 001   operational    online    free
AP 014 CEX6P  Domain 008   operational    online    attached to RICHARD
AP 014 CEX6P  Domain 009   operational    online    attached to RICHARD
AP 014 CEX6P  Domain 075   operational    online    free
AP 014 CEX6P  Domain 076   operational    online    attached to RICHARD
AP 015 CEX6A  Domain 000   operational    online    shared
AP 015 CEX6A  Domain 001   operational    online    free
AP 015 CEX6A  Domain 008   operational    online    free
AP 015 CEX6A  Domain 009   operational    online    free
AP 015 CEX6A  Domain 075   operational    online    free
AP 015 CEX6A  Domain 076   operational    online    free
AP 016 CEX6A  Domain 000   operational    online    shared
AP 016 CEX6A  Domain 001   operational    online    free
AP 016 CEX6A  Domain 008   operational    online    free
AP 016 CEX6A  Domain 009   operational    online    free
AP 016 CEX6A  Domain 075   operational    online    free
AP 016 CEX6A  Domain 076   operational    online    free

Shared-Crypto Users:
ALAN
```

If both TRACY and RICHARD log off, then RICHARD logs on, all resources that are specified on the CRYPTO APDEDICATED statement in RICHARD’s directory entry will be assigned to RICHARD.

	DOM 000	DOM 001	DOM 008	DOM 009	DOM 075	DOM 076
AP 010 CEX7A						
AP 011 CEX7C			RICHARD	RICHARD		RICHARD
AP 012 CEX7C			RICHARD	RICHARD		RICHARD
AP 013 CEX7P			RICHARD	RICHARD		RICHARD
AP 014 CEX6P			RICHARD	RICHARD		RICHARD
AP 015 CEX6A	SHARED					
AP 016 CEX6A	SHARED					

When TRACY logs on next, the following configuration results.

	DOM 000	DOM 001	DOM 008	DOM 009	DOM 075	DOM 076
AP 010 CEX7A	TRACY	TRACY	TRACY			
AP 011 CEX7C			RICHARD	RICHARD		RICHARD
AP 012 CEX7C			RICHARD	RICHARD		RICHARD
AP 013 CEX7P			RICHARD	RICHARD		RICHARD
AP 014 CEX6P			RICHARD	RICHARD		RICHARD
AP 015 CEX6A	SHARED					
AP 016 CEX6A	SHARED					

Attaching and Detaching Dedicated Crypto Resources

RICHARD is logged on as an APDED virtual machine. We can remove some crypto resources from him without requiring a logoff and logon. When detaching resources from an APDED virtual machine, only APs or Domains can be specified on the DETACH CRYPTO command.

```
detach crypto dom 8 from richard
```

Now attach additional resources to TRACY. When attaching resources to an APDED virtual machine, only APs or Domains can be specified on the ATTACH CRYPTO command.

```
attach crypto ap 11 12 to tracy
```

DAMIAN is logged on but does not have access to any crypto resources. To attach dedicated crypto resources to DAMIAN, both AP(s) and Domain(s) must be specified on the ATTACH command.

```
attach crypto ap 14 domain 8 75 to damian
```

	DOM 000	DOM 001	DOM 008	DOM 009	DOM 075	DOM 076
AP 010 CEX7A	TRACY	TRACY	TRACY			
AP 011 CEX7C	TRACY	TRACY	TRACY	RICHARD		RICHARD
AP 012 CEX7C	TRACY	TRACY	TRACY	RICHARD		RICHARD
AP 013 CEX7P				RICHARD		RICHARD
AP 014 CEX6P			DAMIAN	RICHARD	DAMIAN	RICHARD
AP 015 CEX6A	SHARED					
AP 016 CEX6A	SHARED					

These changes are for the current logon session. To make these changes last through logoff and logon, update the user directory.

Changing Access to Shared Crypto Resources

ALAN can remove his access to shared crypto resources.

```
detach crypto apvirtual
```

As long as the CRYPTO APVIRTUAL statement remains in ALAN’s directory entry, he can restore access to shared crypto resources by issuing:

```
define crypto apvirtual
```

A system administrator can make these changes to ALAN's configuration by using the FOR command. For example:

```
for alan cmd detach crypto apvirtual
```

```
for alan cmd define crypto apvirtual
```

Making a Crypto Adapter Temporarily Unavailable to CP and Virtual Machines Running on CP

If maintenance or repair needs to be done on an adapter, it can be taken offline using the VARY OFFLINE CRYPTO command. When the adapter is brought back online, the shared and dedicated assignments are reinstated.

```
vary offline crypto ap 12 15
```

```
q crypto domain
AP 012 CEX7A  Domain 000   disconfigured    offline    attached to TRACY
AP 012 CEX7A  Domain 001   disconfigured    offline    attached to TRACY
AP 012 CEX7A  Domain 008   disconfigured    offline    attached to TRACY
AP 012 CEX7A  Domain 009   disconfigured    offline    free, dedication planned
AP 012 CEX7A  Domain 075   disconfigured    offline    free
AP 012 CEX7A  Domain 076   disconfigured    offline    free, dedication planned
…….
AP 015 CEX7C  Domain 000   disconfigured    offline    shared
AP 015 CEX7C  Domain 001   disconfigured    offline    free
AP 015 CEX7C  Domain 008   disconfigured    offline    free
AP 015 CEX7C  Domain 009   disconfigured    offline    free
AP 015 CEX7C  Domain 075   disconfigured    offline    free
AP 015 CEX7C  Domain 076   disconfigured    offline    free
```

```
vary online crypto ap 12 15
```

```
q crypto domain
AP 012 CEX7A  Domain 000   operational    online     attached to TRACY
AP 012 CEX7A  Domain 001   operational    online     attached to TRACY
AP 012 CEX7A  Domain 008   operational    online     attached to TRACY
AP 012 CEX7A  Domain 009   operational    online     free, dedication planned
AP 012 CEX7A  Domain 075   operational    online     free
AP 012 CEX7A  Domain 076   operational    online     free, dedication planned
…….
AP 015 CEX7C  Domain 000   operational    online     shared
AP 015 CEX7C  Domain 001   operational    online     free
AP 015 CEX7C  Domain 008   operational    online     free
AP 015 CEX7C  Domain 009   operational    online     free
AP 015 CEX7C  Domain 075   operational    online     free
AP 015 CEX7C  Domain 076   operational    online     free
```

Chapter 6. The System Configuration File

This chapter:

- Discusses the parm disk
- Provides a summary of system configuration file statements
- Defines general rules for coding the system configuration file
- Describes each system configuration file statement in detail

The Parm Disk

The system definition information required at IPL resides in files on the *parm disk*, a CMS-formatted minidisk that CP can read. There are multiple parm disks, and this layout is the same for both non-SSI and SSI installations, to help ease the migration to a later SSI environment.

- PMAINT CF0 is the parm disk where the main system configuration file and the logo configuration file are located. If you are using ECKD DASD, this disk is located on the common volume (default label VMCOM1).
  - The main system configuration file, usually called SYSTEM CONFIG, contains operating characteristics such as the layout of the CP system residence disk, lists of DASD volumes that CP uses, your real storage configuration, and information CP requires to determine the correct offset from Coordinated Universal Time (UTC). It also contains real device definitions for I/O devices that do not respond to a sense ID request and for I/O devices that need more information defined than a sense ID request returns (for example, printers and communications controllers). This file is described in detail in this chapter.
  - The logo configuration file, usually called LOGO CONFIG, contains information about the creation and configuration of logos, including the file names and file types of the different logo files. For more information, see Chapter 7, "The Logo Configuration File," on page 337.
- MAINT CF1 is the parm disk where the CPLOAD MODULE is located. This is the CP kernel file that is loaded at IPL. If you are using ECKD DASD, this disk is located on the system residence volume (default label M01RES).
- MAINT*vrm* CF2 (where *vrm* is the z/VM version, release, and modification level) is the parm disk that serves as a staging area for updates applied by the SERVICE command, before you use PUT2PROD to copy them to MAINT CF1. If you are using ECKD DASD, this disk is located on the release volume (default label *vrm*RL1).

If you are using SCSI disks for your installation, all of the parm disks are located on the IPL volume.

Note:

1. The parm disk must be CMS-formatted. If it is not CMS-formatted, the system enters a disabled wait state, code 6758. If you get this wait code, you can use the alternate parm disk at IPL time. To do this, use the Stand-Alone Program Loader (SAPL) in console mode and change the parm disk extent to the alternate parm disk extent on the menu screen. For more information, see Using the Stand-Alone Program Loader in *z/VM: System Operation*.

   On an Extended Address Volume (EAV), a parm disk can reside anywhere, but is limited to 65520 cylinders in size, since it must be CMS-formatted.
2. During SHUTDOWN REIPL or system restart processing, CP uses the parm disk location from the previous IPL. If the parm disk was moved by use of CPFMTXA or ICKDSF, CP might fail to find the CPLOAD module and message HCP6739E is issued, or CP might find the old CPLOAD module on the previous parm disk. Use the EXTENT option of the SHUTDOWN command to force CP find the parm disk at its new location, or use the OFFSET option to specify the new location of the parm disk. It is recommended to only move the parm disk shortly before a planned REIPL.

Summary of System Configuration File Statements

Table 8 on page 48 lists all the system configuration file statements, gives you a brief description of each statement, and points you to where you can get more information about each statement.

Table 8. System Configuration File Statements (SYSTEM CONFIG)

Statement	Description	Location
ACTIVATE ISLINK	Identifies a communication link to ISFC.	"ACTIVATE ISLINK Statement" on page 55
ALTERNATE_OPERATORS	Identifies up to eight user IDs that have the potential to become the system operator automatically when the primary system operator logs off.	"ALTERNATE_OPERATORS Statement" on page 56
ASSOCIATE EXIT	Assigns one or more entry points or external symbols to an exit point during initialization.	"ASSOCIATE EXIT Statement" on page 58
ASSOCIATE MESSAGES/MSGS	Assigns an external symbol to a local message repository and gives CP information about how to select messages in that repository during initialization.	"ASSOCIATE MESSAGES / MSGS Statement" on page 62
BEGIN / END	Identifies blocks of system configuration file statements that apply to particular systems.	"BEGIN / END Statements" on page 65
CHARACTER_DEFAULTS	Establishes system-wide defaults for the logical character delete, escape, line delete, line end, and tab symbols.	"CHARACTER_DEFAULTS Statement" on page 67
CP_ACCESS	Defines the list of disks that CP accesses immediately when it brings the user directory on line.	"CP_ACCESS Statement" on page 70
CP_ADDON_INITIALIZE_ROUTINES	Lists the installation-added entry points that will be called during system initialization.	"CP_ADDON_INITIALIZE_ROUTINES Statement" on page 72
CP_OWNED	Defines and generates a list of up to 255 CP-owned volumes.	"CP_OWNED Statement" on page 73
CPXLOAD	Loads a file containing customer-written CP routines from the parm disk or a CP-accessed disk into the system execution space during initialization.	"CPXLOAD Statement" on page 76
CRYPTO APVIRTUAL	Specifies the APs and domains for shared crypto use.	"CRYPTO APVIRTUAL Statement" on page 80
CU	Defines the way CP initializes specific control units.	"CU Statement" on page 84
DEFINE ALIAS	Defines a new alias for an existing CP command on the system during initialization.	"DEFINE ALIAS Statement" on page 87
DEFINE COMMAND/CMD	Defines a new CP command or a new version of an existing CP command on the system during initialization.	"DEFINE COMMAND / CMD Statement" on page 90
DEFINE DIAGNOSE	Defines a new DIAGNOSE code on the system during initialization.	"DEFINE DIAGNOSE Statement" on page 96
DEFINE EXIT	Defines a new exit point in CP during initialization.	"DEFINE EXIT Statement" on page 100
DEFINE LAN	Defines a guest LAN.	"DEFINE LAN Statement" on page 103
DEFINE VSWITCH	Creates a CP system-owned switch (a virtual switch) to which virtual machines can connect.	"DEFINE VSWITCH Statement" on page 107
DEVICES	Tells CP what to do with various devices at IPL.	"DEVICES Statement" on page 123
DISABLE COMMAND/CMD	Prevents CP from processing requests for the specified CP command during and after initialization.	"DISABLE COMMAND / CMD Statement" on page 128

Table 8. System Configuration File Statements (SYSTEM CONFIG) (continued)

Statement	Description	Location
DISABLE DIAGNOSE	Prevents CP from processing requests for the specified locally-developed DIAGNOSE codes during and after initialization.	"DISABLE DIAGNOSE Statement" on page 130
DISABLE EXITS	Prevents CP from calling all entry points and external symbols associated with one or more exit points during and after initialization.	"DISABLE EXITS Statement" on page 132
DISTRIBUTE	Specifies the distribution features for the local system.	"DISTRIBUTE Statement" on page 134
DRAIN	Stops new operations on specified real devices.	"DRAIN (Disk) Statement" on page 136
EDEVICE	Defines an emulated device that represents a real device.	"EDEVICE Statement" on page 139
EMERGENCY_MESSAGE_CONSOLES	Defines consoles for system emergency messages.	"EMERGENCY_MESSAGE_CONSOLES Statement" on page 144
ENABLE COMMAND/CMD	Permits CP to process requests for the specified CP command during and after initialization.	"ENABLE COMMAND / CMD Statement" on page 146
ENABLE DIAGNOSE	Permits CP to process requests for the specified locally-developed DIAGNOSE codes during and after initialization.	"ENABLE DIAGNOSE Statement" on page 148
ENABLE EXITS	Permits CP to call all entry points and external symbols associated with one or more exit points during and after initialization.	"ENABLE EXITS Statement" on page 150
ENCRYPT	Specifies settings for your system's host level encryption.	"ENCRYPT Statement" on page 152
ENFORCE_BY_VOLID	Enforces attachment of DASD devices by their VOLIDs on the ATTACH command.	"ENFORCE_BY_VOLID Statement" on page 154
EQUATE	Groups names of systems that all have something in common and should thus be treated similarly.	"EQUATE Statement" on page 155
EXTERNAL_SYNTAX	Adds locally-developed system configuration file statements to the system without modifying the system configuration file processor, HCPZSC ASSEMBLE.	"EXTERNAL_SYNTAX Statement" on page 157
FEATURES	Establishes certain attributes of the system at system initialization.	"FEATURES Statement" on page 158
FORM_DEFAULT	Generates default user form names.	"FORM_DEFAULT Statement" on page 173
HOT_IO_RATE	Specifies the number of contiguous unsolicited interrupts that CP will accept from faulty I/O devices before it stops accepting input from those devices.	"HOT_IO_RATE Statement" on page 175
IMBED	Specifies files to imbed into the SYSTEM CONFIG file at IPL.	"IMBED Statement" on page 178
INIT_MITIME	Specifies the MITIME (the time interval at which a device is checked for missing interrupts) that is in effect during device initialization at system IPL time.	"INIT_MITIME Statement" on page 180
IODF	Specifies the IODF that HCD will use in its control of the I/O configuration.	"IODF Statement" on page 181
JOURNALING	Specifies whether CP should include the journaling facility in the system being generated; establishes attributes of the facility.	"JOURNALING Statement" on page 183
LOGO_CONFIG	Specifies the name and type of a logo configuration file.	"LOGO_CONFIG Statement" on page 187

Table 8. System Configuration File Statements (SYSTEM CONFIG) (continued)

Statement	Description	Location
MODIFY COMMAND/CMD	Redefines an existing CP command on the system during initialization.	"MODIFY COMMAND / CMD Statement" on page 188
MODIFY DIAGNOSE	Redefines an existing DIAGNOSE code on the system during initialization.	"MODIFY DIAGNOSE Statement" on page 192
MODIFY EXIT	Redefines or removes an existing dynamic CP exit point during initialization.	"MODIFY EXIT Statement" on page 195
MODIFY LAN	Modifies the properties of a guest LAN.	"MODIFY LAN Statement" on page 198
MODIFY PORT	Defines or changes the OSA devices that make up a link aggregation group and sets the attributes of a link aggregation group.	"MODIFY PORT Statement" on page 200
MODIFY PRIV_CLASSES	Changes the privilege classes and establishes initial values.	"MODIFY PRIV_CLASSES Statement" on page 205
MODIFY VSWITCH	Modifies the properties of an existing virtual switch.	"MODIFY VSWITCH Statement" on page 206
MULTITHREADING	Defines the multithreading characteristics of the system.	"MULTITHREADING Statement" on page 215
OPERATOR_CONSOLES	Defines a list of consoles from which CP can choose an operator console.	"OPERATOR_CONSOLES Statement" on page 217
PRINTER_TITLE	Specifies the printed output classes that are to contain classification titles.	"PRINTER_TITLE Statement" on page 221
PRIV_CLASSES	Changes the privilege class that authorizes certain internal CP functions.	"PRIV_CLASSES Statement" on page 223
PRODUCT	Defines a product or feature to the system.	"PRODUCT Statement" on page 225
RDEVICE	Adds to the system's definition of a set of real devices.	"RDEVICE Statement" on page 227
RELOCATION_DOMAIN	Defines subsets of SSI membership as domains.	"RELOCATION_DOMAIN Statement" on page 260
SAY	Writes a line of text to the operator's console during initialization.	"SAY Statement" on page 261
SET SHUTDOWNTIME	Defines the amount of time reserved for a CP shutdown to be performed.	"SET SHUTDOWNTIME Statement" on page 263
SET SIGNAL	Defines the duration of the system default shutdown signal timeout interval.	"SET SIGNAL Statement" on page 265
SET VARIABLE	Defines and sets an environment variable that is accessible to every class G user on the system.	"SET VARIABLE Statement" on page 267
SRM	Changes system resource manager settings related to HiperDispatch.	"SRM Statement" on page 269
SSI	Defines information about the name of the SSI cluster, location of the persistent data record (PDR), and systems that are members of the SSI cluster.	"SSI Statement" on page 273
START	Restarts devices after they have been drained; changes the processing options in effect for devices.	"START (Disk) Statement" on page 276
STORAGE	Configures the use of real storage.	"STORAGE Statement" on page 278
SYSTEM_ALIAS	Specifies HyperPAV alias devices to be automatically attached to the system during system initialization.	"SYSTEM_ALIAS Statement" on page 285
SYSTEM_DATEFORMAT	Sets the system-wide default date format for commands that provide multiple date formats.	"SYSTEM_DATEFORMAT Statement" on page 286

Table 8. System Configuration File Statements (SYSTEM CONFIG) (continued)

Statement	Description	Location
SYSTEM_IDENTIFIER	Creates a system name for the processor on which you run z/VM.	"SYSTEM_IDENTIFIER Statement" on page 287
SYSTEM_IDENTIFIER_DEFAULT	Provides CP with a default system name.	"SYSTEM_IDENTIFIER_DEFAULT Statement" on page 290
SYSTEM_RESIDENCE	Describes the layout of the system residence disk.	"SYSTEM_RESIDENCE Statement" on page 292
SYSTEM_USERIDS	Specifies user IDs that will perform special functions during and after IPL.	"SYSTEM_USERIDS Statement" on page 294
THROTTLE	Limits the number of I/O operations that guest operating systems can initiate to a specific real device.	"THROTTLE Statement" on page 297
TIMEZONE_BOUNDARY	Tells CP which time zone to choose at IPL.	"TIMEZONE_BOUNDARY Statement" on page 298
TIMEZONE_DEFINITION	Defines system time zones according to their distance from UTC.	"TIMEZONE_DEFINITION Statement" on page 300
TOLERATE_CONFIG_ERRORS	Marks sections of the system configuration file in which CP is not to tolerate errors.	"TOLERATE_CONFIG_ERRORS Statement" on page 302
TRANSLATE_TABLE	Specifies replacements for standard translation tables.	"TRANSLATE_TABLE Statement" on page 304
USERFORM	Creates a list of user form names and their corresponding operator form numbers; specifies forms as NARROW so that a narrow separator form is printed.	"USERFORM Statement" on page 308
USER_DEFAULTS	Defines defaults to be used for • global lines per page values for virtual printers and consoles defined on the system, and • querying other users' POSIX database information and having their POSIX security values changed.	"USER_DEFAULTS Statement" on page 309
USER_VOLUME_EXCLUDE	Defines volumes to be excluded from the user volume list.	"USER_VOLUME_EXCLUDE Statement" on page 312
USER_VOLUME_INCLUDE	Defines user volumes by a generic volume identifier.	"USER_VOLUME_INCLUDE Statement" on page 314
USER_VOLUME_RDEV	Specifies a user DASD volume at a specific real device number.	"USER_VOLUME_INCLUDE Statement" on page 314
USER_VOLUME_LIST	Generates a list of user DASD volumes not to be used for paging, spooling, directory, or temporary disk space.	"USER_VOLUME_LIST Statement" on page 316
VMLAN	Controls global attributes for guest LAN and virtual switch support.	"VMLAN Statement" on page 320
XLINK_DEVICE_DEFAULTS	Changes the defaults for the CSE area location and format for particular DASD types.	"XLINK_DEVICE_DEFAULTS Statement" on page 324
XLINK_SYSTEM_EXCLUDE	Specifies the systems that CP is to exclude from cross-system link.	"XLINK_SYSTEM_EXCLUDE Statement" on page 328
XLINK_SYSTEM_INCLUDE	Specifies the systems that CP is to include in cross-system link.	"XLINK_SYSTEM_INCLUDE Statement" on page 329
XLINK_VOLUME_EXCLUDE	Specifies the DASD volumes that CP is exclude from cross-system link.	"XLINK_VOLUME_EXCLUDE Statement" on page 331
XLINK_VOLUME_INCLUDE	Specifies the DASD volumes that CP is include in cross-system link.	"XLINK_VOLUME_INCLUDE Statement" on page 333

General Rules for Coding a System Configuration File

When creating or updating a system configuration file, you must follow some general syntax rules.

Format

The system configuration file can be a fixed-length or variable-length record file. After all continuations of a statement have been resolved, no individual statement in the file should be longer than 4000 characters, and no individual record in the file should be longer than 4000 characters.

Comments

You can add comments to the file by delimiting them with a beginning character sequence of /* and an ending sequence of */. A single comment may span multiple records in the file. Thus,

```
/*-------------------------------------------------------*
 * The following section defines the CP owned volume list *
 *-------------------------------------------------------*/
```

is a valid comment. As many comments as necessary may be entered on a single record in the file. For example,

```
 Features Retrieve Max 100 /* Max for all */ Default 7 /* 7 to start */
```

is allowed.

Continuations

To put a statement in more than one record of the configuration file, place a comma at the ends of all but the last line of the statement. For example, you can code the FEATURES statement as follows:

```
 Features                 ,
    Retrieve Max 100 ,    /* Max for all */
    Default 7             /* 7 to start  */
```

A comma at the end of a line indicates that the statement is not yet complete. You should not code a comma if the statement information is complete but a comment continues to the next record.

```
 Features                 ,
    Retrieve Max 100 ,    /* Maximum number of buffers a non
                             ... privileged user may ask for      */
    Default 7             /* Seven to start.  This matches the
                             ... previous release's default       */
```

is valid, but a comma after DEFAULT 7 would make the statement invalid, as you have specified no subsequent options. The comma can be placed directly after the last entry on the line. Thus, the example above would also be valid if specified as

```
 Features,
    Retrieve Max 100,     /* Maximum number of buffers a non
                             ... privileged user may ask for      */
    Default 7             /* Seven to start.  This matches the
                             ... previous release's default       */
```

Finally, a blank line does not cause a continuation to be terminated. You could specify

```
 Features,

    Retrieve Max 100
```

because CP ignores any blank lines in the configuration file.

Case

In general, it does not matter whether information in the system configuration file is entered in upper case or mixed case. Most entries in the configuration file are converted to upper case before they are processed. Thus,

```
System_Identifier_Default  THATVM
```

and

```
System_Identifier_Default  thatvm
```

would have the same results. An exception to this rule occurs when CP encounters a quoted string in the file. A quoted string is any string enclosed within single quotation marks; the string may contain blanks and special character sequences such as /* and */ that are not usually permitted in a single token. Where quoted strings are allowed and a quoted string is specified in the configuration file, the text inside the quotation marks is not converted to upper case. In the following example, the specified printer title would remain in mixed case:

```
Printer_title  I   'Internal Use Only'
```

while

```
Printer_title  J   Confidential
```

would create a print classification of CONFIDENTIAL.

Record Qualifiers

You can instruct CP to process certain statements in the system configuration file only if the file is being used on certain systems. To do this, add one or more record qualifiers onto the front of a statement. A record qualifier consists of a parameter followed by a colon. The parameter can be the system name to which you wish the statement to apply, or it can be one of the following:

- a pattern that contains the special characters * (used in place of one or more arbitrary characters) and % (used in place of exactly one arbitrary character) and that matches one or more systems to which you wish the statement to apply
- a symbol defined with an EQUATE statement that lists the system or systems to which you wish the statement to apply. An EQUATE statement can list such systems either explicitly or describe them using the pattern matching discussed above.

For example, specifying

```
BOBVM1: BOBVM2:  Operator_Consoles 00F2
```

will cause the above OPERATOR_CONSOLES statement to be processed only if the system being IPLed is called BOBVM1 or BOBVM2. CP finds out the name of the system from SYSTEM_IDENTIFIER or SYSTEM_IDENTIFIER_DEFAULT statements.

The Order of Statements in the File

In general, statements may be specified in the system configuration file in any order. There are, however, a few exceptions to this rule:

- TIMEZONE_BOUNDARY statements can refer only to time zones that have been previously defined by TIMEZONE_DEFINITION statements.
- IMBED statements and LOGO_CONFIG statements can only use the special filename or filetype of `-SYSTEM-` if the system name has been previously resolved by a SYSTEM_IDENTIFIER or SYSTEM_IDENTIFIER_DEFAULT statement.
- Record qualifiers can only be used if the system name has been previously resolved by a SYSTEM_IDENTIFIER or SYSTEM_IDENTIFIER_DEFAULT statement.

Therefore, we recommend that you code SYSTEM_IDENTIFIER, SYSTEM_IDENTIFIER_DEFAULT, and EQUATE statements at the top of the configuration file.

For statements that do not specify lists of items, statements that occur later in the configuration file override equivalent statements specified earlier. For example,

```
Features  Disable Clear-TDisk
.
.
.
Features  Enable  Clear_TDisk
```

would cause TDISK clearing to be enabled. Exceptions to this rule are the CP_OWNED statement and the IODF statement. If you specify more than one CP-OWNED statement with the same slot number, CP uses only the first occurrence of these statements and ignores the subsequent statements. Likewise, if you specify more than one IODF statement, CP uses only the first occurrence of the IODF statement and ignores the subsequent statements.

Statements that specify lists of items are usually processed cumulatively.

```
User_Volume_List ESAUS1 ESAUS2 ESAUS3
.
.
.
User_Volume_List ESAUS4 ESAUS5 ESAUS6
```

would cause the user volume list to consist of all six specified volumes. Two exceptions to this rule are the OPERATOR_CONSOLES statement and the EMERGENCY_MESSAGE_CONSOLES statement. Each invocation of one of these statements defines an entire list of consoles to be used for a specific purpose; any subsequent invocation causes the entire list for that purpose to be replaced.

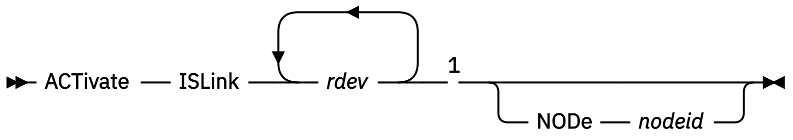
Checking the Syntax of Statements in the File

You can check the syntax of the statements in a system configuration file by using the CPSYNTAX command. This command checks the specified system configuration file and any files imbedded in that file. For details, see *z/VM: CP Commands and Utilities Reference*.

Adding New Operands to Existing Statements

Warning: Care should be taken when adding new operands to existing statements in the system configuration file. When a statement is processed by a CPLOAD module that does not include support for the new operands on the statement, an error message is displayed and the entire statement will be ignored. This will cause features and settings that were previously in effect on your system to no longer be in effect. Backing out to a previous CPLOAD module or running a mixed-release / mixed-service level SSI cluster are situations where an older CPLOAD module may be used. To avoid issues caused by an existing statement being ignored, code each new operand on a separate statement until it is supported by all releases of CP that will use this system configuration file. Also, ensure the new statement is within a "TOLERATE_CONFIG_ERRORS YES" section of the file if you wish to avoid being prompted during CP initialization.

ACTIVATE ISLINK Statement



Notes:

1 You can specify a maximum of 16 real device numbers.

Purpose

Use the ACTIVATE ISLINK statement to identify a communication link to ISFC.

How to Specify

Include as many statements as needed; they are optional.

Note that when an SSI statement is included in the configuration file, ACTIVATE ISLINK statements must also be specified to define direct ISFC connections from every member to each of the other members of the SSI cluster.

Operands

rdev

identifies one to sixteen real device numbers you want to use as a link.

When the first device is initialized for a new link, the ISLINK for that node is created automatically.

NODe *nodeid*

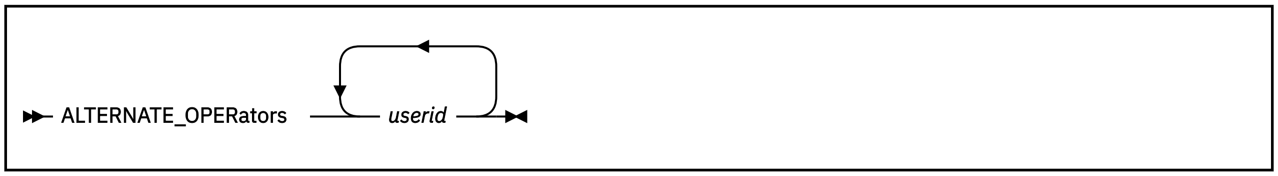
is the optional node (gateway) identifier for the ISFC link expected for each *rdev*. If not specified, it is determined by device initialization.

If a conflicting *nodeid* is specified for an already active device, an error message is issued.

Usage Notes

1. The system operator receives all informational and error messages.
2. You can improve throughput across your ISFC link by activating more than one device on the same link. Note, however, that FICON® subchannels on the same channel path share some hardware resources. In general you will observe better data transfer rates if subchannels on a given link are configured to use different channel paths.
3. When an SSI statement is specified, ACTIVATE ISLINK statements must also be specified to define direct ISFC connections to each of the other members of the SSI cluster. If other member(s) are joined to the cluster during IPL of an SSI member, ISFC connections to the joined members must be established before the system operator is logged on and IPL completes.
   - If there is not an ISFC connection to every joined member, message HCP1669I is displayed and the system waits until there are connections to all joined members.
   - If there are not enough ACTIVATE ISLINK statements to define direct connections to every joined member, message HCP1670E is displayed, followed by disabled wait state 1670.
4. Avoid using different CHPID speeds in any one logical link, as that will cause the ISFC logical link to operate less efficiently.

ALTERNATE_OPERATORS Statement



Purpose

Use the ALTERNATE_OPERATORS statement to specify a maximum of eight user IDs that have the potential to become the system operator automatically when the primary system operator logs off.

How to Specify

The ALTERNATE_OPERATORS statement is optional. You can place the ALTERNATE_OPERATORS statement anywhere in the system configuration file. If you specify more than one ALTERNATE_OPERATORS statement, the last statement overrides any previous specifications.

Operands

userid

specifies up to eight alternate operator user IDs. If the current primary system operator logs off and the default primary system operator is not available, one of these user IDs automatically becomes the primary system operator if the following conditions are met:

- the user ID is logged on or disconnected
- the user ID has at least one of the privilege classes required for the system operator at the time the operator logs off.

The variable *userid* is an alphanumeric character string of as many as eight characters. There is no default alternate operator user ID.

Usage Notes

1. If alternate operator user IDs are specified and the primary system operator logs off, CP will select the default operator ID before the alternates if the default operator is logged on or disconnected.
2. If no alternate operator user IDs are specified or no alternate can be selected when the primary system operator logs off, there will be no operator until one the following events occurs:
   - The default operator ID logs on
   - Any user ID logs on with at least one of the privilege classes required for the system operator
   - The SET SYSOPER command successfully selects a new operator
   - The system is IPLed.
3. For more information, see "SYSTEM_USERIDS Statement" on page 294.
4. For more information, see "PRIV_CLASSES Statement" on page 223.

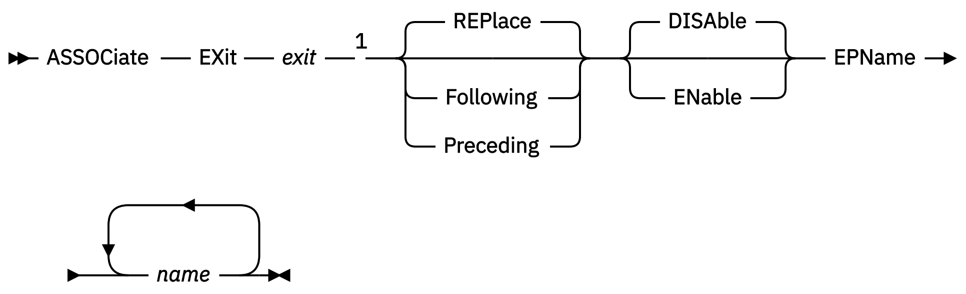
Examples

To specify user IDs ALTOP, OTHEROP and OPBACKUP as alternate operators, use the following ALTERNATE_OPERATORS statement:

```
Alternate_Operators,
         ALTOP,   /* This ID is tried after the default operator */
```

```
            OTHEROP, /* This ID will be tried next                  */
            OPBACKUP /* This ID is tried last                       */
```

ASSOCIATE EXIT Statement



Notes:

[1] You can specify the following operands in any order, as long as EPNAME is the last operand specified.

Purpose

Use the ASSOCIATE EXIT statement to assign one or more entry points or external symbols to an exit point during initialization.

You can also assign entry points and external symbols to an exit point after initialization by using the ASSOCIATE EXIT command. For more information, see ASSOCIATE EXIT in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place ASSOCIATE EXIT statements anywhere in the system configuration file.

If you specify more than one statement with the same exit point number, CP keeps a cumulative list of operations. For example, if you have one statement indicating that you want to replace the list of entry point names and a later statement indicating that you want to add an entry point to the end of the list, CP replaces the list **and** adds to the end. The second statement does not overrule the first statement.

Operands

exit
is the number of the exit point to which you want to assign an entry point or external symbol. The variable *exit* must be a hexadecimal number between X'0000' and X'FFFF'.

REPlace
tells CP to replace the current list of entry point names and external symbols that are already associated with this exit point with the list specified after the EPNAME operand.

Note: The order that you specify the entry points and external symbols is the order in which CP will call them.

Following
tells CP to add the specified entry point names or external symbols to the end of the list of existing entry point names and external symbols that are already associated with the specified exit point number.

Note: The order that you specify the entry points and external symbols is the order in which CP will call them.

Preceding
tells CP to add the specified entry point names or external symbols to the beginning of the list of existing entry point names and external symbols that are already associated with the specified exit point number.

Note: The order that you specify the entry points and external symbols is the order in which CP will call them.

DISAble
tells CP not to call the entry points and external symbols associated with this exit point until you enable it. (For more information about enabling exit points, see Usage Note "4" on page 59.) If omitted, DISABLE is the default.

ENable
tells CP to immediately start calling the entry points and external symbols associated with this exit point.

EPName *name*
is the name (or names) of the entry point or external symbol that CP calls when encountering this exit point number. Each *name* must be a 1-character to 8-character string. The first character must be alphabetic or one of the following special characters: dollar sign ($), number sign (#), underscore (_), or at sign (@). The rest of the string can be alphanumeric characters, the four special characters ($, #, _, and @), or any combination thereof.

Note: The order that you specify the entry points and external symbols is the order in which CP will call them.

Usage Notes

1. If you specify the ASSOCIATE EXIT statement in your system configuration file, you should also specify the CPXLOAD statement to load the customer-written CP routines for the exit point into the system execution space. These customer-written CP routines should contain the entry point names and external symbols that you will specify on the ASSOCIATE EXIT statement.

   If CP cannot locate one or more of the entry point names or external symbols on your ASSOCIATE EXIT statement after initialization, CP will ignore the unknown entry point name or external symbol that it does not recognize and continues normal processing. If the unknown entry point or external symbol is part of a list associated with an exit point, CP continues processing the other members of the list. CP does not ignore an exit point because it cannot find one entry point or external symbol in the list. CP only ignores an exit point if it cannot find all the entry points and external symbols in the list.

   For more information, see "CPXLOAD Statement" on page 76. See also CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
2. To display whether there are any unknown entry points or external symbols associated with an exit point, use the QUERY UNRESOLVED command. For more information, see QUERY UNRESOLVED in *z/VM: CP Commands and Utilities Reference*.
3. If you specify the ASSOCIATE EXIT statement and do not specify the ENABLE operand, CP will redefine the exit point with the information from your ASSOCIATE EXIT statement, but will not call any of the entry points or external symbols associated with that exit point until you later enable it.
4. There are 4 ways to enable an exit point:
   - Specify another ASSOCIATE EXIT statement further on in your system configuration file and specify the ENABLE operand,
   - Specify an ENABLE EXITS statement further on in your system configuration file. For more information, see "ENABLE EXITS Statement" on page 150.
   - Enter an ASSOCIATE EXIT command after initialization and specify the ENABLE operand, or
   - Enter an ENABLE EXITS command after initialization.

For more information, see ASSOCIATE EXIT and ENABLE EXITS in *z/VM: CP Commands and Utilities Reference*.

By default, exit points are disabled. Thus, in general, you should follow any ASSOCIATE EXIT commands or statements with ENABLE EXITS commands or statements.

5. CP calls the entry points and external symbols for an exit point in the order that you specify them on the ASSOCIATE EXIT statement or command, unless an entry point overrides this action. Any entry point can tell CP to change the normal processing flow by skipping one or all subsequent entry points or external symbols.
6. To display status and usage statistics information about a specific exit point after initialization, use the QUERY EXITS command. For more information, see QUERY EXITS in *z/VM: CP Commands and Utilities Reference*.

   Note: While processing the ASSOCIATE EXIT statement (or command), CP creates a CP exit block for the specified exit point. For a static exit point, CP does not create CP exit control blocks until you associate one or more entry points or external symbols with that exit point. If you try to issue a QUERY EXITS command against such an exit point, CP issues message HCP2752E as the response to your QUERY EXITS command. For a dynamic exit point, QUERY EXITS responds with the definition of the exit, even if there are no entry points associated with it.
7. To display the address of the CP exit block for a specific exit point after initialization, use the LOCATE XITBK command. For more information, see LOCATE XITBK in *z/VM: CP Commands and Utilities Reference*. Again, if you have not associated one or more entry points or external symbols with the specified exit point, there is no CP exit block for CP to locate and display. Instead, CP issues error message HCP2752E.
8. To display the address of the CP indirect call locator block for a specific exit point, use the LOCATE ICLBK command. For more information, see LOCATE ICLBK in *z/VM: CP Commands and Utilities Reference*.
9. To change the definition of an existing dynamic exit point, or remove the exit point from the system, use the MODIFY EXIT statement or command. For more information, see "MODIFY EXIT Statement" on page 195. See also MODIFY EXIT in *z/VM: CP Commands and Utilities Reference*.
10. To stop CP from calling the entry points and external symbols associated with one or more exit points after defining those exit points, use the DISABLE EXITS command. For more information, see DISABLE EXITS in *z/VM: CP Commands and Utilities Reference*.
11. To remove the customer-written CP routines from the system execution space:
    a. Use the DISASSOCIATE command to revoke all entry point and external symbol assignments that were made with the ASSOCIATE EXIT statement or command. For more information, see DISASSOCIATE in *z/VM: CP Commands and Utilities Reference*.
    b. Use the CPXUNLOAD command to unload the customer-written CP routines. For more information, see CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.
12. After creating a CP exit block, CP will not erase that CP exit block until another IPL. Disabling the exit point affects certain fields in the CP exit block, but does not erase it. Disassociating the entry point names and external symbols erases those fields in the CP exit block, but does not erase the CP exit block itself.
13. For more information about user-defined exit points, see Benefits of using CP exits in *z/VM: CP Exit Customization*.

Examples

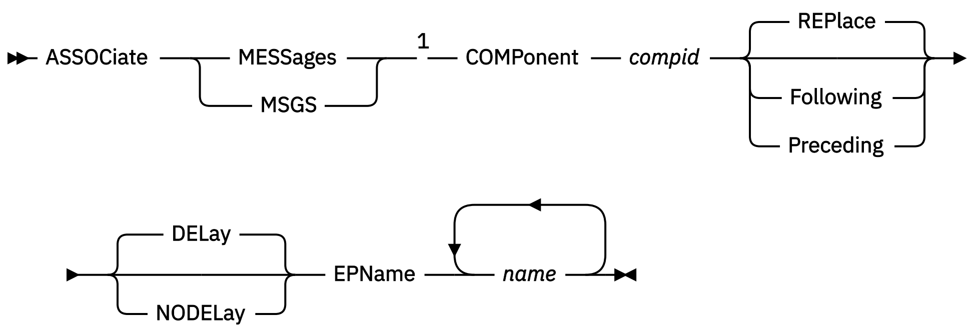
1. To have CP associate entry point HCPSRC00 with exit number F and to replace any existing entry point associations, use the following:

```
Associate Exit f EPname hcpsrc00
```

2. To have CP add entry point HCPSRC04 at the end of the current list for exit point 9C, use the following:

```
Associate Exit 9c Following EPname hcpsrc04
```

ASSOCIATE MESSAGES / MSGS Statement



Notes:

1 You can specify the following operands in any order, as long as EPNAME is the last operand specified.

Purpose

Use the ASSOCIATE MESSAGES / MSGS statement to assign an external symbol to a local message repository and to give CP information about how to select the messages in that repository during initialization.

You can also assign external symbols to local message repositories after initialization using the ASSOCIATE MESSAGES or MSGS commands. For more information, see ASSOCIATE MESSAGES / MSGS in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place ASSOCIATE MESSAGES or MSGS statements anywhere in the system configuration file.

If you specify more than one statement with the same component name, CP keeps a cumulative list of operations. For example, if you have one statement indicating that you want to replace the list of entry point names and a later statement indicating that you want to add an entry point to the end of the list, CP replaces the list **and** adds to the end. The second statement does not overrule the first statement.

Operands

COMPonent *compid*
tells CP the component identifier to use when issuing one of the messages in the local message repository. The variable *compid* is a 1-character to three-character alphanumeric string. For example, the component ID for the system message repository (z/VM) is HCP, which is, by default, the last message repository in the search list. For more information, see Usage Note “1” on page 63.

REPlace
tells CP to replace the current list of entry point names and external symbols that are already associated with this local message repository with the list specified after the EPNAME operand.

Note: The order that you specify the entry points and external symbols is the order in which CP will call them.

Following
tells CP to add the specified entry point names or external symbols to the end of the list of existing entry point names and external symbols that are already associated with the specified local message repository.

Note: The order that you specify the entry points and external symbols is the order in which CP will call them.

Preceding
tells CP to add the specified entry point names or external symbols to the beginning of the list of existing entry point names and external symbols that are already associated with the specified local message repository.

Note: The order that you specify the entry points and external symbols is the order in which CP will call them.

DELay
tells CP to process this ASSOCIATE statement after all of the CP_ACCESS statements have been processed and after all of the delayed CPXLOAD statements have been processed. If omitted, DELAY is the default.

NODELay
tells CP to process this ASSOCIATE statement immediately.

EPName *name*
is the name (or names) of the entry point or external symbol that points to the data in the system execution space where the local message repository can be found. Each *name* must be a 1-character to 8-character string. The first character must be alphabetic or one of the following special characters: dollar sign ($), number sign (#), underscore (_), or at sign (@). The rest of the string can be alphanumeric characters, the four special characters ($, #, _, and @), or any combination thereof.

Note: The order that you specify the entry points and external symbols is the order in which CP will call them.

Usage Notes

1. HCP is the standard component ID for z/VM messages. If you do not specify any ASSOCIATE MESSAGES or MSGS statements (or commands) for an entry point, HCPMES is, by default, the only message repository in the search list for that entry point.

   If you do assign one or more local message repositories to an entry point, those repositories are added to the search list in the order that you specify (using the REPLACE, FOLLOWING, or PRECEDING operands) and, by default, HCPMES is the last message repository in the search list.

   When the routines in that entry point issue a message, CP searches the first message repository in the search list. If CP finds the message in that repository, it issues the message and does not search any more repositories. If CP does not find the message, it continues searching through each repository until it finds the first occurrence of that message.

   Using ASSOCIATE MESSAGES or MSGS statements (or commands), you can assign local message repositories which override existing z/VM messages in the HCPMES repository. Or, you can specify HCP as the component ID and move the z/VM message repository to a place in the search list other than last place.
2. Before CP can begin using your message repository, you must:
   a. Generate your messages using the CMS GENMSG command. For more information, see CMS GENMSG in *z/VM: CP Commands and Utilities Reference*.
   b. Load the message repository file by using the CPXLOAD statement or command. For more information, see “CPXLOAD Statement” on page 76. See also CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
3. If CP cannot find the entry point you specified for EPNAME when the message is being displayed, CP displays message substitution data, if any.
4. To display information about the local message repositories available on your system, use the QUERY CPLANGLIST command and specify the ASSOCIATED operand. For more information, see QUERY CPLANGLIST in *z/VM: CP Commands and Utilities Reference*.
5. To remove the message repository file from the system execution space:

   a. Use the DISASSOCIATE command to revoke the external symbol assignment made with the ASSOCIATE MESSAGES or MSGS statement (or command.)
   b. Use the CPXUNLOAD command to unload the repository.

   For more information, see DISASSOCIATE and CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.

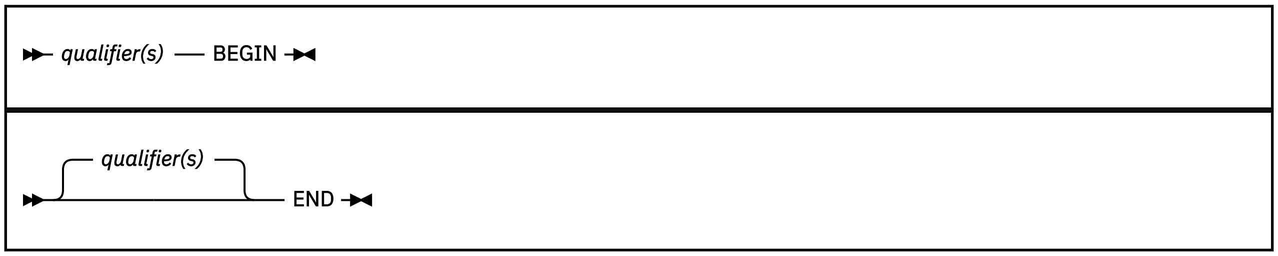
6. To have CP use the message repositories associated with a specific component, you must specify that component ID on the COMPID keyword of the HCPCONSL macroinstruction. For more information, see *z/VM: CP Exit Customization*

Examples

1. To have CP assign an uppercase English message repository containing messages starting with OUR to entry point OURMESS1, use the following:

```
Associate Messages component our,          /* Set it up so we can use */
                   preceding,              /* messages we wrote.      */
                   epname    ourmess1
```

BEGIN / END Statements



Purpose

Use BEGIN and END statements to identify blocks of system configuration file statements that apply to particular systems. Blocks make it easy to define CP-owned lists and other statements that are unique to specific systems in a common configuration file.

How to Specify

BEGIN is an optional statement. If a BEGIN statement is specified, then a corresponding END statement must be specified in the same file. A BEGIN block cannot end in a different file.

Statement blocks bound by BEGIN and END cannot be nested in other BEGIN blocks.

Operands

qualifier(s)

are the record qualifiers used to identify the systems that are associated with a BEGIN block. For the definition of a record qualifier, see "Record Qualifiers" on page 53. Up to 64 qualifiers can be specified on a BEGIN or END statement.

The END statement applies to all qualifiers specified on the previous BEGIN statement. Therefore, qualifiers are optional on END, but if specified, must match those specified on the previous BEGIN.

Usage Notes

1. No record qualifiers are allowed on individual statements within a BEGIN block.
2. If an error related to a BEGIN or END statement is found, a wait state 1689 occurs. A BEGIN or END error could result in statements being included or excluded erroneously. Depending on which statements process incorrectly, the system might initialize in a manner that results in loss of data. The wait state enables you to fix the problem without risking the loss of data.

Examples

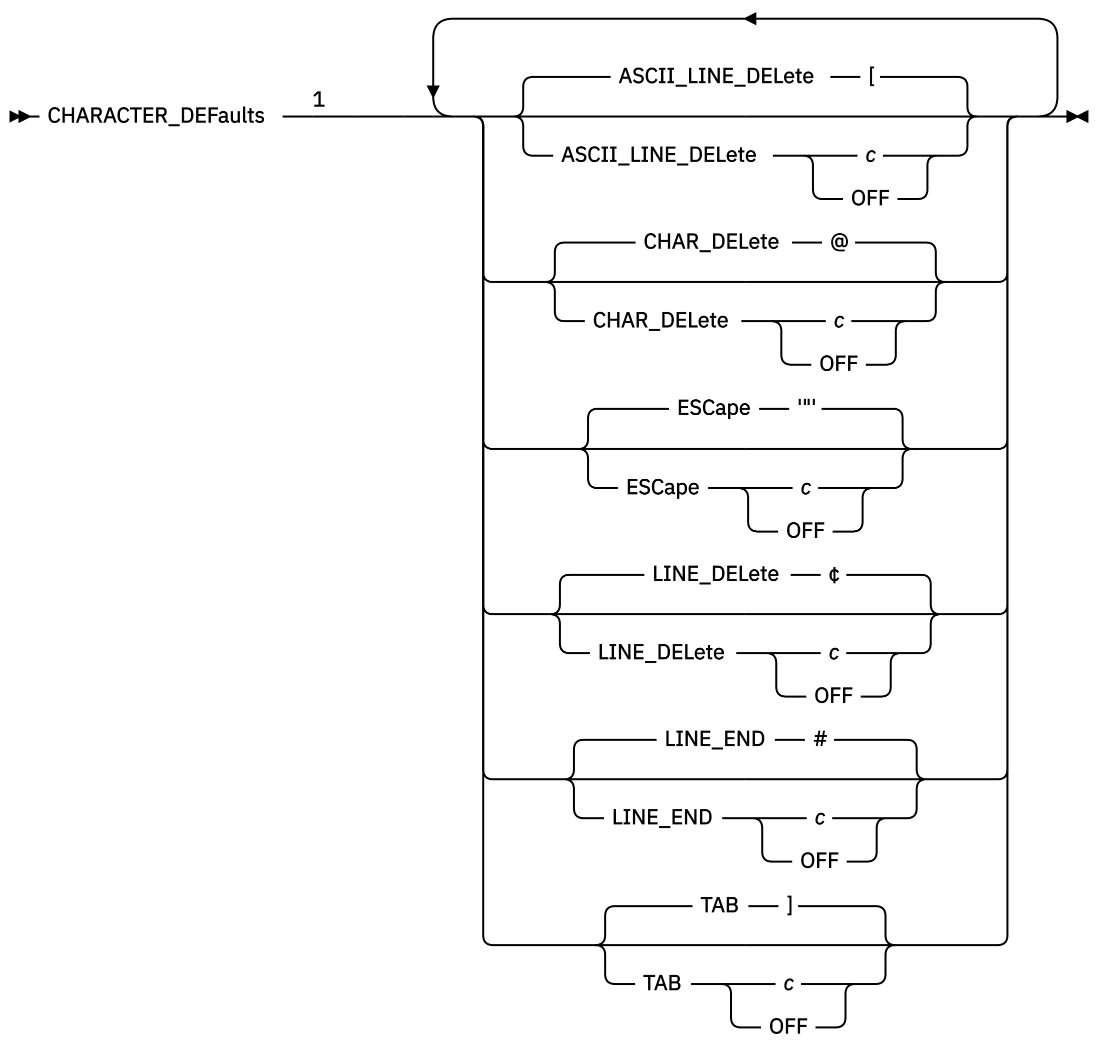
1. To have one set of CP-OWNED statements apply to one system and another set of CP-OWNED statements apply to another system, use the following statements:

```
VM1:  BEGIN
  CP_Owned   Slot   1  VM1PG1
  CP_Owned   Slot   2  SPVOL1 Owned
  CP_Owned   Slot   3  SPVOL2 Shared
  CP_Owned   Slot   4  VM1PG2
VM1:  END

VM2:  BEGIN
  CP_Owned   Slot   1 VM2PG1
  CP_Owned   Slot   2 SPVOL1 Shared
  CP_Owned   Slot   3 SPVOL2 Owned
  CP_Owned   Slot   4 VM2PG2
VM2:  END
```

2. The following shows an OPERATOR_CONSOLES statement that applies to two systems:

```
VM1: VM2: BEGIN
   Operator_Consoles 0009 ,
                     001F 0500 0520 0530 0540 0550 0560 0570 0580 ,
                     001D 001E
          END
```

CHARACTER_DEFAULTS Statement



Notes:

[1] You must specify at least one of the following operands.

Purpose

Use the CHARACTER_DEFAULTS statement to set the default characters that will represent the logical character delete, escape, line delete, line end, and tab symbols on your system.

Operands

ASCII_LINE_DELete *c*
: defines the logical line delete symbol for `ASCII` devices on your system. Choose a character that is not commonly specified by the users on your system. If you specify `OFF`, your system will not have a logical line delete function for ASCII devices. If omitted, the default is a left bracket ([).

CHAR_DELete *c*
: defines the default logical character delete symbol on your system. Choose a character that is not commonly specified by the users on your system. If you specify `OFF`, your system will not have a logical character delete symbol. If omitted, the default is an at sign (@).

ESCape *c*
: defines the default logical escape symbol on your system. Choose a character that is not commonly specified by the users on your system. If you specify OFF, your system will not have a logical escape character. If omitted, the default is a double quotation mark (").

LINE_DELete *c*
: defines the logical line delete symbol for all non-ASCII devices on your system. Choose a character that is not commonly specified by the users on your system. If you specify OFF, your system will not have a logical line delete function. If omitted, the default is a cent sign (¢).

LINE_END *c*
: defines the logical line end symbol on your system. Choose a character that is not commonly specified by the users on your system. If you specify OFF, your system will not have a logical line end symbol. If omitted, the default is a pound sign (#).

TAB *c*
: defines the logical tab symbol on your system. If you specify OFF, your system will not have a logical tab character. If omitted, the default is a right bracket (]).

For each of the Operands, you can specify *c* by typing the character (*c*), by typing the character enclosed in single or double quotation marks ('*c*') or ("*c*") or by typing the hexadecimal equivalent of the character (X'*hh*') or (X"*hh*"). If you need to specify a single quotation mark, enclose it within double quotation marks. If you need to specify a double quotation mark, enclose it within single quotation marks.

Usage Notes

1. The CHARACTER_DEFAULTS statement defines default symbols for your entire system. If you want to override the system defaults for a specific user, you can specify defaults for that user:
   - On the USER or IDENTITY directory statement in the user directory. For more information, see "USER Directory Statement" on page 616 and "IDENTITY Directory Statement" on page 510.
   - By having the user issue the CP TERMINAL command after logging on. For more information, see TERMINAL in *z/VM: CP Commands and Utilities Reference*.
2. You cannot use any of the letters A through Z, the numbers 0 through 9, or the bytes X'0E' (shift out) or X'0F' (shift in) for the line-end, line-delete, character-delete, escape, or tab characters.
3. With the widespread use of "@" in internet addresses, it is recommended that you specify OFF for the default character-delete symbol, or define a symbol other than @ as the character-delete symbol, to avoid problems in the data stream.
4. Specifying a blank character (X'40') for any of the default symbols is not recommended.
5. The character display depends on the code page used by the terminal emulator. These characters are from code page 037 United States:

   # X'7B'
   : SYSLEND LINEND

   ¢ X'4A'
   : SYSLDEL LINDEL

   @ X'7C'
   : SYSLCEL CHARDEL

   " X'7F'
   : SYSLESCP ESCAPE

   ⎵ X'6A'
   : SYSTAB TABCHAR

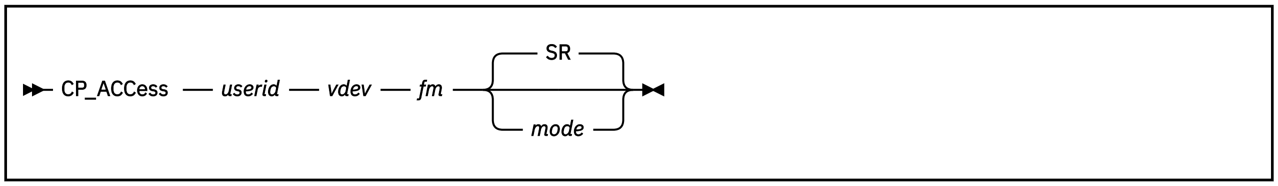
   [X'AD'
   : SYSALDEL LINDEL

Examples

1. To turn off the logical line delete symbol and define the logical tab symbol on your system as a backslash (\), use the following CHARACTER_DEFAULTS statement:

   ```
   Character_Defaults Line_Delete off,     /* Turn off Line_Delete    */
                      Tab \                /* Override Tab default    */
   ```

CP_ACCESS Statement



Purpose

Use the CP_ACCESS statement to specify a CMS-formatted minidisk that CP should access when it brings the user directory online. This CP-accessed minidisk may contain system, logo, or device information to use at run-time. This information may include log messages and logos. CP searches for this information on any CP-accessed disks you specify in the order that you specify.

How to Specify

Include as many statements as needed; they are optional. You can place CP_ACCESS statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

userid
specifies the user ID of the owner of the minidisk that you want to make available to CP.

vdev
is the virtual device number of the specified user's minidisk, as defined in the virtual machine definition in the user directory. The number can be any hexadecimal number between X'0000' and X'FFFF'.

fm
is the file mode letter that you want assigned to all files on the specified minidisk. You can specify any letter from A to Z.

mode
is the access mode. The following is a list of valid modes listed in order of increasing control:

R
Read-only access. CP establishes read access.

RR
Read-only access. CP establishes read access.

W
Write access. CP establishes write access.

WR
Write access. CP establishes write access. If write access is denied, CP establishes read access.

M
Multiple-write access. CP establishes write access. If a previous write, stable, or exclusive mode access exists, CP denies access.

MR
Multiple-write access. CP establishes write access. If a previous write or stable access exists, CP establishes read-only access.

MW
Multiple-write access. CP establishes write access in all cases.

SR
(The default) Stable read-only access. CP establishes read access. CP denies all requests for write access to a disk with an existing SR mode access. A stable access means that the user holding the SR access can be assured that the disk remains stable, unchanged by others, until CP releases the access.

SW
Stable write access. CP establishes write access. CP denies all requests for write access to a disk with an existing SW mode access. A stable access means that the user holding the SW access can be assured that the disk remains stable, unchanged by others, until CP releases the access.

SM
Stable multiple access. CP establishes write access. CP denies all requests for write access to a disk with an existing SM mode access. A stable access means that the user holding the SM access can be assured that the disk remains stable, unchanged by others, until CP releases the access.

ER
Exclusive read-only access. CP establishes read access. CP denies all requests for access to a disk with an existing exclusive mode. An exclusive access means that the user holding the ER access has stable access with the added restriction that no one else has, or can get access to, the specified minidisk until CP releases the access.

EW
Exclusive write access. CP establishes write access. CP denies all requests for access to a disk with an existing exclusive mode. An exclusive access means that the user holding the EW access has stable access with the added restriction that no one else has, or can get access to, the specified minidisk until CP releases the access.

Usage Notes

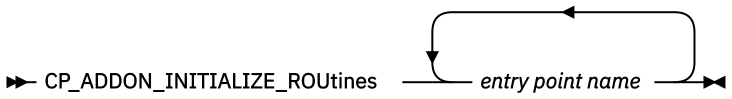
1. Because CP does not bring the user directory online until after all CP_ACCESS statements are parsed, CP cannot tell you whether you specified an invalid user ID and virtual device number combination. To find out which CP_ACCESS statements were valid, you must wait until initialization completes and check to see what minidisks CP accessed. To display all the CP-accessed minidisks, use the QUERY CPDISKS command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
2. If you specify the NODIRECT option during the IPL, CP ignores all CP_ACCESS statements in the system configuration file.
3. After initializing CP, use the CPACCESS command to access minidisks for CP and use the CPRELEASE command to release minidisks. For more information, see CPACCESS and CPRELEASE in *z/VM: CP Commands and Utilities Reference*.
4. The use of the stable and exclusive link modes (SR, SW, SM, ER, EW) is controlled by the LNKSTABL and LNKEXCLU options on the OPTION directory statement. For more information, see "OPTION Directory Statement" on page 572.

Examples

1. To access three minidisks owned by the MAINT and PICTURE user IDs, use the following CP_ACCESS statements:

```
CP_Access maint    0300  a  SR    /* Give CP access to LOGMSG files */
CP_Access maint    0301  b  SR    /* Give CP access to backup
                                      LOGMSG files                */
CP_Access picture  0193  c  SR    /* Give CP access to LOGO files   */
```

CP_ADDON_INITIALIZE_ROUTINES Statement



Purpose

Code the CP_ADDON_INITIALIZE_ROUTINES statement to generate a list of installation-added entry points in CP which are to be called during system initialization.

How to Specify

The CP_ADDON_INITIALIZE_ROUTINES statement is optional; if specified, you can include as many statements as you need as long as the total number of entry points does not exceed 500.

You can place CP_ADDON_INITIALIZE_ROUTINES statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

entry point name
is the installation-defined entry point which is to be called during system initialization. The variable *entry point name* is a 1-character to 8-character identifier. This must be a valid CP entry point, which can be located with the CP LOCATE command. The entry point must be MP (multi-processor)-capable and use a dynamic savearea.

Usage Notes

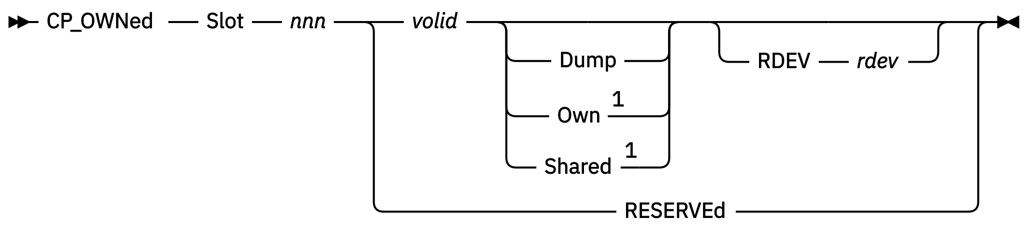
1. Entry points listed must be included in the CP module.
2. CP will schedule each of the specified entry points for execution during the system initialization process. However, the entry points may not run until after initialization is complete.

Examples

To specify LOCALMOD, USERMOD, and XMOD as entry points to be started during system initialization, code the following CP_ADDON_INITIALIZE_ROUTINES statement:

```
CP_ADDON_INITIALIZE_ROUTINES  LOCALMOD,USERMOD,XMOD
```

CP_OWNED Statement



Notes:

[1] The OWN and SHARED operands are ignored and are included for compatibility only.

Purpose

Use the CP_OWNED statement to define a list of up to 255 CP-owned DASD volumes. CP-owned DASD volumes are the CP system residence volume and any volumes containing real system paging, spooling, dump, directory, and temporary disk space.

How to Specify

Include as many statements as needed; they are optional. You can place CP_OWNED statements anywhere in the system configuration file. If you specify more than one statement with the same slot number, CP uses only the first statement. Subsequent CP_OWNED statements do not redefine the slot.

Operands

Slot *nnn*
tells CP the number of the slot in the CP-owned volume list. *nnn* must be a decimal number from 1 to 255.

volid
is the 1- to 6-character volume serial number of the volume you want to include in the CP-owned volume list.

Dump
tells CP to reserve the spool space on the specified volume exclusively for dumps.

Own
is ignored and is included for compatibility only. In an SSI cluster, volume ownership is determined by the ownership information recorded on the volume.

Shared
is ignored and is included for compatibility only. In an SSI cluster, volume ownership is determined by the ownership information recorded on the volume.

RDEV *rdev*
specifies the real device address of the DASD volume you want to include in the CP-owned volume list. The variable *rdev* is a 1- to 4-character hexadecimal device address.

RESERVEd
tells CP to reserve the slot for future use.

Usage Notes

1. Before CP can use a DASD, you must format, label, and allocate space on the DASD. You can do this using the Device Support Facilities program (ICKDSF) or the CPFMTXA utility. However, we recommend that you use the ICKDSF method of CP volume maintenance because it is required for certain DASD types. You should format ECKD devices without filler records using ICKDSF. For more information

about ICKDSF, see the *ICKDSF User's Guide and Reference*. For more information about CPFMTXA, see CPFMTXA in *z/VM: CP Commands and Utilities Reference*.

2. If you specify a volume that is not mounted on a CP_OWNED statement when CP is loaded, CP considers that volume unavailable. If possible, CP continues processing and creates an empty slot for the specified volume in the CP-owned volume list. By creating empty slots in the CP-owned volume list, you can provide space for future growth. When you need to attach a volume to the system, the system operator can mount and attach the volume (using the CP ATTACH command) without having to re-IPL the system. For more information, see ATTACH in *z/VM: CP Commands and Utilities Reference*.
3. You can add new volumes to a system that has already been IPLed in RESERVED slots in the CP-owned volume list. For more information, see "Adding DASD Space to a Running System" on page 660.

   To delete a CP_OWNED volume:

   - Remove the CP_OWNED statement from the system configuration file. In this case, you must IPL with a cold start.
   - Change the volid to RESERVED in the system configuration file. In this case, you must IPL, but a cold start may not be necessary.

   Attention: If you delete any volume that contains spool space from the CP-owned volume list, or move any volume that contains spool space to a different slot in the CP-owned volume list, a clean start is required. A clean start causes the deletion of spool files and system data files, including named saved systems and saved segments. Files that need to be preserved over such a change should be dumped to tape using the SPXTAPE DUMP command.

   After the clean start, SPXTAPE LOAD should be used to restore the spool files and system data files from tape. This ensures that spool files and system data files that existed on the system before deletion or movement of volumes in the CP-owned volume list are restored correctly.
4. If you specify the DUMP option for a volume with spool space that is already in use when you IPL the system, those spool files remain on that volume until you move them with a CP SPXTAPE DUMP command with the PURGE option. For more information, see SPXTAPE in *z/VM: CP Commands and Utilities Reference*.
5. If the DASD volume at the specified RDEV address has a different volume ID than specified on the CP_OWNED statement, or if no DASD volume exists at the specified RDEV address, no volume with the specified volume ID is attached.
6. At system initialization (IPL), if more than one DASD volume has the same volume serial number (*volid*) and that *volid* is in the CP or user volume list, and no *rdev* has been specified for that *volid*, the volume with the lowest device number is attached to the system. This rule does not apply to duplicates of the system residence volume (the volume containing the minidisk from which the CP nucleus module was read).
7. Specifying an *rdev* on the CP_OWNED statement provides a convenient method for managing a case where there might be multiple DASD defined to the system that have the same volume ID, and the device with the higher device number is the one that should be attached. This eliminates the need to set the devices OFFLINE_AT_IPL and then attach the devices after the IPL process has completed.
8. For IBM recommendations on dump space allocation, see "Allocating Space for CP Hard Abend Dumps" on page 655.

Examples

1. To define a CP-owned volume list that contains:

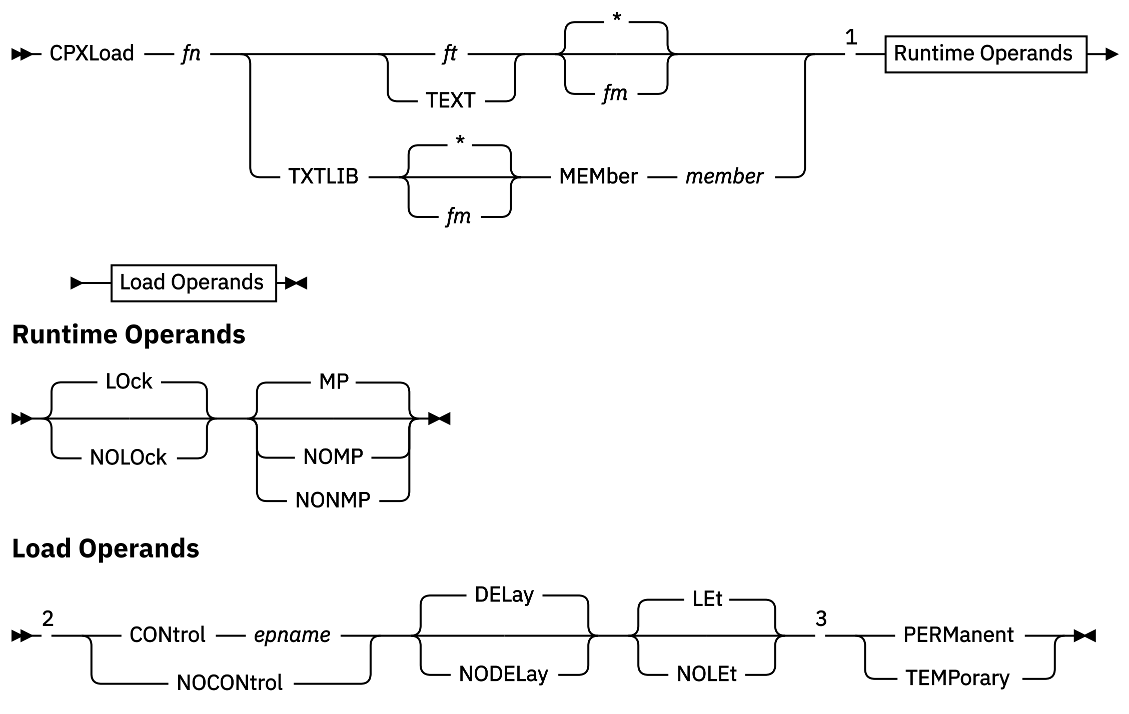
   - A system residence volume
   - A volume for spool and temporary disk space
   - Two volumes for spool space, one residing at a particular RDEV number
   - A volume for dump space

- Three empty slots for volumes you will be getting shortly

use the following CP_OWNED statements:

```
CP_Owned  Slot 001  esares             /* System Residence volume   */
CP_Owned  Slot 002  essys1             /* Spool and T-disk space    */
CP_Owned  Slot 003  essys2             /* Spool space               */
CP_Owned  Slot 004  sysdmp   Dump      /* Dump space                */
CP_Owned  Slot 005  spool1 RDEV 3BC0   /* More spool space          */

CP_Owned  Slot 006  Reserved           /* Leave some slots open in  */
CP_Owned  Slot 007  Reserved           /* case we need to add some  */
CP_Owned  Slot 008  Reserved           /* extra spool space later.  */
```

CPXLOAD Statement



Notes:

[1] You can specify the Runtime and Load operands in any order.

[2] If you specify CONTROL or NOCONTROL on an OPTIONS directive, you can omit them on this CPXLOAD command.

[3] If you specify PERMANENT or TEMPORARY on an OPTIONS directive, you can omit them on this CPXLOAD command.

Purpose

Use the CPXLOAD statement to load a file containing customer-written CP routines from the parm disk or a CP-accessed disk into the system execution space during initialization.

You can also load customer-written CP routines into the system execution space after initialization using the CPXLOAD command. For more information, see CPXLOAD in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place CPXLOAD statements anywhere in the system configuration file. For more information about where you can place the customer-written CP routines, see Usage Note "3" on page 78.

Operands

fn

is the name of the file that you want loaded. Because you are loading during system initialization, you must make sure the CP routines are located in a place where CP can find them. See Usage Note "3" on page 78 for more information.

ft

is the file type (other than TEXT or TXTLIB) of the file that you want loaded.

This file may contain CPXLOAD directives, text records, or a combination of both. If the file contains text records, it must have a fixed record format (RECFM F) and a logical record length of 80 (LRECL 80). If the file does not contain text records, it can be any record format and must have a logical record length less than 4,000 (LRECL < 4000).

TEXT

tells CP that this is a text file that contains 1 or more CSECTs and can contain 1 or more CPXLOAD directives. Text files must have a fixed record format (RECFM F) and a logical record length of 80 (LRECL 80).

TXTLIB

tells CP that the file is a text library that contains 1 or more members. A TXTLIB member can contain 1 or more control sections (CSECTs) and can contain CPXLOAD directives. TXTLIB files must have a fixed record format (RECFM F) and a logical record length of 80 (LRECL 80).

*****

tells CP to search the list of CP-accessed minidisks until it finds the first occurrence of the specified file that you want loaded. If you do not specify a file mode, * is the default.

fm

is the file mode of the CP-accessed minidisk containing the file that you want loaded.

MEMber *member*

is the name of the member in the TXTLIB that you want loaded.

You can use generic member names to request a specific subset of files. A generic member name is a 1-character to 8-character string with asterisks (*) in place of 1 or more characters and percent signs (%) in place of exactly 1 character. For example:

```
hc%p* ...
```

lists all members that start with HC and have P as their fourth character.

LOck
NOLOck

has no effect and is retained only for compatibility. All symbols are considered resident, which means they cannot be locked or unlocked.

MP

tells CP that the entry point is multiprocessor (MP) capable. This means that the entry point can be dispatched on any of the machine's processors. If omitted, MP is the default.

NOMP
NONMP

tells CP that the entry point is dispatched only on the master processor, because (in general) the entry point assumes that competitive routines are also not multiprocessor (MP) capable. Use NOMP or NONMP to prevent entry points from overlaying each other's chains of control blocks when you do not take the precaution of getting a system lock. For example, SPOOL routines are NOMP.

CONtrol *epname*

tells CP to call the specified entry point after loading the customer-written CP routines and before processing a CPXUNLOAD request. You can load the customer-written CP routines containing the specified entry point either before or within this CPXLOAD request. The variable *epname* must be a 1-character to 8-character string. The first character must be alphabetic or one of the following special characters: dollar sign ($), number sign (#), underscore (_), or at sign (@). The rest of the string can be alphanumeric characters, the 4 special characters ($, #, _, and @), or any combination thereof.

Note: Normally, if CP cannot find an entry point when processing an exit point routine, it ignores the unknown entry point and continues normal processing. This is not true when you specify the CONTROL *epname* operand. If CP cannot find the entry point you specify on CONTROL, CP will terminate processing your CPXLOAD statement and will not load the customer-written CP routines into the system execution space.

NOCONtrol
tells CP not to call an entry point after loading the customer-written CP routines and before processing a CPXUNLOAD request.

DELay
tells CP to process all CP_ACCESS statements before processing this CPXLOAD statement. The customer-written CP routines that you are loading must be located on one of the disks specified on one of the CP_ACCESS statements. If omitted, DELAY is the default.

NODELay
tells CP to process this CPXLOAD statement immediately and not to wait until after processing the CP_ACCESS statements. This means that the customer-written CP routines must be a file on the parm disk, which is the only disk that CP has access to at this stage of the initialization process. If you specify a file mode letter and NODELAY, CP ignores your file mode, issues message HCP2777I, and looks for your customer-written CP routines on the parm disk.

LEt
tells CP to load the specified file and to ignore any records that are completely blank or that contain an unexpected value in column 1. This is meant to accommodate the noncommented information that can be left in a TEXT file by an assembler utility such as VMHASM.

NOLEt
tells CP to stop loading the specified file when it encounters an unexpected value in column 1. Column 1 is expected to contain '*' (to denote a comment), X'02' (to denote a TEXT record), or blank (to denote a possible CPXLOAD directive).

PERManent
tells CP that the customer-written CP routines being loaded are to remain a part of CP until a CP SHUTDOWN command is issued or a software-initiated restart (bounce) occurs. This means you cannot use the CPXUNLOAD command to remove these CP routines.

TEMPorary
tells CP that the customer-written CP routines being loaded can be unloaded in the future with a CPXUNLOAD command.

Usage Notes

1. When loading your files into storage, CP treats each control section (CSECT) independently for storage allocation. Also during loading, CP allocates 1 page of storage to each CSECT. There is 1 exception: if CP encounters a CSECT of zero length during CPXLOAD processing, that CSECT is deleted. If you need to load a zero-length CSECT, add an EXPAND directive to your input file. For example, if you had zero-length CSECT XXXDOG to load, you would add "EXPAND XXXDOG(8)" to your input file.
2. When invoking your files, CP treats each entry point in a CSECT independently for the MP attribute.
3. The customer-written CP routines that you are loading must be on a disk that CP has access to when the load operation is done. If you have a CPXLOAD statement in your system configuration file that specifies the NODELAY operand, the customer-written CP routines must be a file on the parm disk because the minidisks specified on CP_ACCESS statements are not available until the end of the initialization process. If your CPXLOAD statement specifies the DELAY operand, the customer-written CP routines can be on any disk, as long as you have specified a CP_ACCESS statement for that disk in your configuration file. For more information, see "CP_ACCESS Statement" on page 70.
4. You can specify runtime and load operands on either the CPXLOAD statement or the OPTIONS directive, or both. However, if you specify options on both and those options conflict, CP uses the options from the CPXLOAD statement. For example, suppose you specify PERMANENT on the OPTIONS directive and TEMPORARY on the CPXLOAD statement, CP will load the CP routines as temporary. For more information, see OPTIONS Directive in *z/VM: CP Exit Customization*.
5. To assign entry points and external symbols to an exit point and to enable or disable that exit point, use the ASSOCIATE EXIT statement or command. For more information, see "ASSOCIATE EXIT Statement" on page 58. See also ASSOCIATE EXIT in *z/VM: CP Commands and Utilities Reference*.

6. To assign an external symbol to a local message repository, use the ASSOCIATE MESSAGES or MSGS statement or command. For more information, see "ASSOCIATE MESSAGES / MSGS Statement" on page 62. See also ASSOCIATE MESSAGES / MSGS in *z/VM: CP Commands and Utilities Reference*.
7. To display information about customer-written CP routines loaded by the CPXLOAD statement, use the QUERY CPXLOAD command. For more information, see CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
8. To display information about external symbols you may have loaded, use the LOCATE SYMBOL command. For more information, see LOCATE SYMBOL in *z/VM: CP Commands and Utilities Reference*.
9. To remove the customer-written CP routines from the system execution space:
   a. Use the DISASSOCIATE command to revoke all entry point and external symbol assignments that were made with the ASSOCIATE EXIT statement or command. For more information, see DISASSOCIATE in *z/VM: CP Commands and Utilities Reference*.
   b. Use the CPXUNLOAD command to unload the customer-written CP routines. For more information, see CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.
10. To remove CPXLOADed files from the system execution space:
   a. Use the DISASSOCIATE command to revoke all entry point and external symbol assignments that were made by using the ASSOCIATE EXIT or ASSOCIATE MESSAGES statements or commands. For more information, see DISASSOCIATE in *z/VM: CP Commands and Utilities Reference*.
   b. Use the CPXUNLOAD command to unload the customer-written CP routines. For more information, see CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.
11. For more information about loading customer-written CP routines into the system execution space, about runtime and load operands, and about CPXLOAD directives, see *z/VM: CP Exit Customization*.
12. CPXLOAD provides the ability to load executable code, message repositories, and data modules dynamically. Only compiled files may be loaded. These compiled files would be the TEXT file output from the assembler or from the CMS GENMSG command.

Examples

1. To have CP load a multiprocessor capable abend text deck, use the following:

```
CPXload abend text *  MP,         /* Load the abend text deck       */
                     Control kaos,
                     Permanent
```

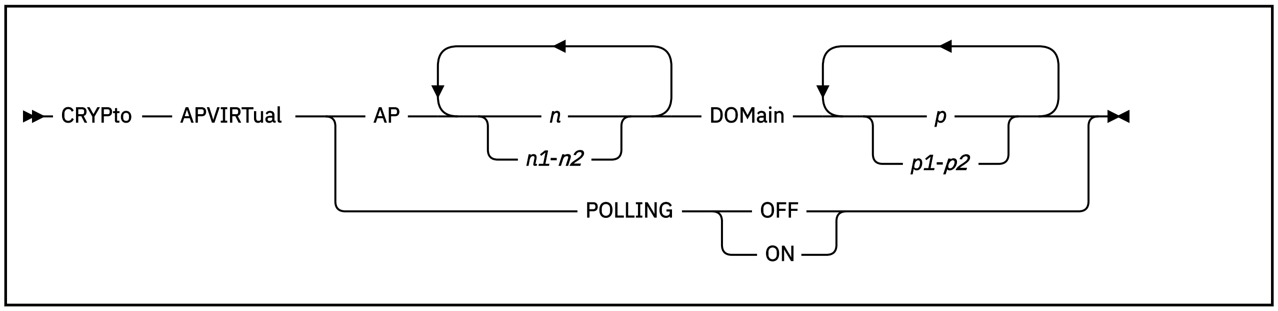
   CP assigns a load identifier (load ID) to the loaded CP routines. If you ever want to unload these CP routines (using the CPXUNLOAD command), you will need to specify the load ID that CP assigned. If you do not know the load ID, use the ALL operand of the QUERY CPXLOAD command to display (among other things) the load IDs of all the CP routines loaded onto (and not yet unloaded from) the system.

   The load ID that CP assigns is a 1-digit to 10-digit decimal number between 0 and 2,147,483,647. The first time you use the CPXLOAD statement (or command), CP assigns that set of customer-written CP routines a load ID of 0. CP increases the load ID by 1 for each subsequent CPXLOAD request. If all the CPXLOAD requests are successful, you will have a sequential list of loaded CP routines.

   If one or more of the CPXLOAD requests are unsuccessful or you issue the CPXULOAD command to unload some customer-written CP routines, there will be one or more gaps in your sequential list of loaded CP routines. The next time you load some customer-written CP routines, CP ignores these gaps in the list and assigns the newly-loaded CP routines a load ID that is one more than the last assigned load ID.

   For example, suppose you had customer-written CP routines loaded at IDs 0 through 7 and you unloaded the CP routines at IDs 2, 3, and 5. This means you still have CP routines loaded at IDs 0, 1, 4, 6, and 7. The next time you use the CPXLOAD command, CP will assign 8 as the load ID for those CP routines. CP will not try to fill in the gaps at 2, 3, or 5.

CRYPTO APVIRTUAL Statement



Purpose

Use the CRYPTO APVIRTUAL statement to specify the following settings:

- The exact adapters (APs) and domains for shared crypto use.
- The polling setting that is in effect at system IPL.

How to Specify

CRYPTO APVIRTUAL statements are optional. Include as many statements as you require. You can place CRYPTO APVIRTUAL statements anywhere in the system configuration file.

If you include more than one CRYPTO APVIRTUAL statement with a POLLING setting, the last POLLING setting overrides any previous settings.

If facilities to support POLLING OFF are not available, then the default setting is POLLING ON. If facilities to support POLLING OFF are available, and if there are no CRYPTO APVIRTUAL statements with a POLLING setting, then the default setting is POLLING OFF.

Operands

n
n1-n2
: is a crypto adapter number (or numbers). Each adapter number must be a decimal value in the range 0 - 255. You can specify a single adapter number, a nonwrapping range of adapter numbers, or any combination thereof.

p
p1-p2
: is a crypto domain number (or numbers). Each domain number must be a decimal value in the range 0 - 255. You can specify a single domain number, a nonwrapping range of domain numbers, or any combination thereof.

POLLING ON
: CP uses polling to drive the processing of APVIRT crypto work.

POLLING OFF
: CP uses interruptions to drive the processing of APVIRT crypto work. CP does not use polling to drive the processing of APVIRT crypto work

Usage Notes

1. A specific crypto resource is identified with an adapter number (AP number) and a domain number. The CRYPTO APVIRTUAL statements in the system configuration file specify crypto resources to be assigned for shared use during system initialization. A maximum of 512 crypto resources can be assigned from the adapters and domains that are specified in the system configuration file. Duplicate

assignments of crypto resources count toward the limit of 512 resources. Any crypto resources beyond the limit of 512 are ignored.

2. The order of CRYPTO APVIRTUAL statements that specify crypto resources determines the order in which crypto resources are selected for sharing.
3. CP uses only one crypto type and mode for sharing. A resource must be usable to be added to the shared pool. A resource is usable only if it has a device assignment of FREE and a device status of OPERATIONAL or RESETTING. The specified crypto resources are checked for usability in the order that they appear on the statements in the system configuration file. The first usable crypto resource that is found determines the type and mode of crypto that CP uses for sharing. Crypto resources of different types and modes are not included in the shared use pool and remain free.
4. If none of the crypto resources that are specified by CRYPTO APVIRTUAL statements are available, then no crypto resources are defined for shared use at system initialization time.
5. If the system configuration file contains no CRYPTO APVIRTUAL statement that specifies crypto resources, and if multiple types and modes of crypto resources are available, then the shared use type and mode is chosen in the following order: CEX8A, CEX7A, CEX6A, CEX5A, CEX4A, CEX3A, CEX8C, CEX7C, CEX6C, CEX5C, CEX4C, CEX3C. The types and modes of crypto resources that are supported might be different for different processors.
6. Each Crypto resource that is specified by a CRYPTO APVIRTUAL statement in the system configuration file is assigned to the shared use pool if both the following conditions are true:

   - The resource is available for shared use.
   - The resource matches the type of the first resource selected.

   When no CRYPTO APVIRTUAL statement that specifies APs and domains is included in the system configuration file, crypto initialization sets aside no more than two crypto resources to be used for sharing.
7. A crypto resource can be specified on both a CRYPTO APVIRTUAL statement in the system configuration file and a CRYPTO DOMAIN statement in the user directory file. The definition in the system configuration file overrides the definition in the user directory file.
8. It is possible to specify the same crypto resources more than once on a single CRYPTO APVIRTUAL statement or on several CRYPTO APVIRTUAL statements. Duplicate pairs of APs and domains are not filtered and are ignored by CP. Duplicate pairs of APs and domains count toward the limit of 512 crypto resources.
9. Resources that are specified in the CRYPTO statement must be authorized for use by the z/VM LPAR to be usable. Resources that are not authorized for use by zVM's LPAR are ignored until authorized.
10. Crypto Express Adapters can be configured in three modes: Accelerator mode, IBM Common Cryptographic Architecture (CCA) coprocessor mode, or IBM Enterprise Public_Key Cryptography Standards (PKCS) #11 (EP11) coprocessor mode. The following types and modes are used in output from z/VM:

    CEX3A - Crypto Express3 configured in accelerator mode
    CEX3C - Crypto Express3 configured in CCA coprocessor mode.
    CEX4P - Crypto Express4 configured in EP11 coprocessor mode
    CEX4A - Crypto Express4 configured in accelerator mode
    CEX4C - Crypto Express4 configured in CCA coprocessor mode.
    CEX5P - Crypto Express5 configured in EP11 coprocessor mode
    CEX5A - Crypto Express5 configured in accelerator mode
    CEX5C - Crypto Express5 configured in CCA coprocessor mode.
    CEX6P - Crypto Express6 configured in EP11 coprocessor mode
    CEX6A - Crypto Express6 configured in accelerator mode
    CEX6C - Crypto Express6 configured in CCA coprocessor mode.
    CEX7P - Crypto Express7 configured in EP11 coprocessor mode
    CEX7A - Crypto Express7 configured in accelerator mode
    CEX7C - Crypto Express7 configured in CCA coprocessor mode.

CEX8A - Crypto Express8 configured in accelerator mode
CEX8C - Crypto Express8 configured in CCA coprocessor mode.
CEX8P - Crypto Express8 configured in EP11 coprocessor mode.

A crypto adapter must be configured in accelerator or CCA coprocessor mode to be included in the shared pool.

11. After system initialization, crypto resources can be added to the system's shared pool by using the ATTACH command with the CRYPTO operand. Crypto resources can be removed from the system's shared pool by using the DETACH CRYPTO command. For more information, see ATTACH and DETACH CRYPTO in *z/VM: CP Commands and Utilities Reference*.
12. After system initialization, the polling setting can be changed by using the SET CRYPTO APVIRTUAL command with the POLLING operand. For more information, see SET CRYPTO in *z/VM: CP Commands and Utilities Reference*.
13. For more information, see Chapter 5, "Crypto Planning and Management," on page 33.

Examples

1.
```
CRYPTO APVIRT AP 1 2 DOMAIN 7 8
```

This statement causes CP to check the crypto resources in the following order:

AP 1 DOMAIN 7
AP 1 DOMAIN 8
AP 2 DOMAIN 7
AP 2 DOMAIN 8

If AP 1 DOMAIN 7 is available at system initialization time, then the type of this crypto determines the type of crypto that is used for sharing. The remaining crypto resources in the list are checked for availability. If they are available and they match the type of the first crypto resources, then they are added to the shared pool. If they are available but do not match the type that is used for sharing, then they are not added to the shared pool and remain free.

2.
```
CRYPTO APVIRT AP 1 DOMAIN 7 8
CRYPTO APVIRT AP 2 DOMAIN 7 8
```

These statements have the same result as example "1" on page 82.

3.
```
CRYPTO APVIRT AP 1-4 5 DOMAIN 10-11 6
```

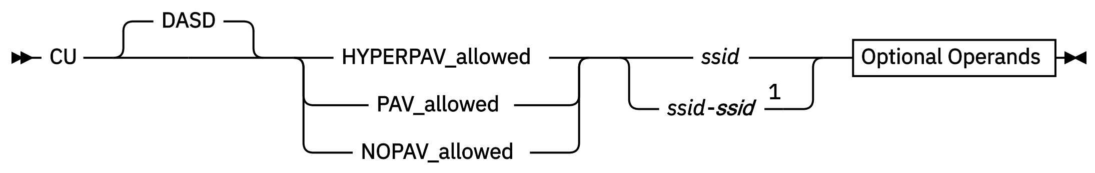
These statements cause CP to check the crypto resources in the following order:

AP 1 DOMAIN 10
AP 1 DOMAIN 11
AP 1 DOMAIN 6
AP 2 DOMAIN 10
AP 2 DOMAIN 11
AP 2 DOMAIN 6
AP 3 DOMAIN 10
AP 3 DOMAIN 11
AP 3 DOMAIN 6
AP 4 DOMAIN 10
AP 4 DOMAIN 11
AP 4 DOMAIN 6
AP 5 DOMAIN 10
AP 5 DOMAIN 11
AP 5 DOMAIN 6

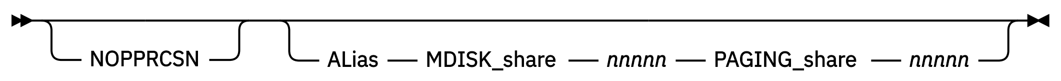
4.
```
CRYPTO APVIRT POLLING OFF
CRYPTO APVIRT AP 1 DOMAIN 7
CRYPTO APVIRT POLLING ON
```

   Because POLLING is specified on two statements, the last setting overrides previous settings. At system IPL, the setting is POLLING ON.

CU Statement



Optional Operands



Notes:

[1] If the same *ssid* is specified more than once, the last occurrence is used.

Purpose

Use the CU statement to define the way CP initializes specific control units.

Operands

DASD
: tells CP that the specified control units are DASD control units.

HYPERPAV_allowed
: tells CP to allow a specified control unit to operate with HyperParallel Access Volume (HyperPAV) devices in your z/VM system.

PAV_allowed
: tells CP to allow a specified control unit to operate with Parallel Access Volume (PAV) devices in your z/VM system.

NOPAV_allowed
: tells CP to not allow a specified control unit to operate with PAV or HyperPAV devices in your z/VM system.

ssid
ssid-ssid
: is the subsystem identifier (as established in a control unit during its installation) of a control unit on which this statement is to operate. The variable *ssid* must be a hexadecimal number between X'0000' and X'FFFF'. You can specify a single subsystem identifier, a list, a range, or any combination thereof.

NOPPRCSN
: tells CP to disable Peer-to-Peer Remote Copy (PPRC) suspend summary notifications for a specified control unit in your z/VM system.

ALias
: tells CP the system configuration statement applies to aliases that are associated with the control unit. When this operand is specified, MDISK_share and PAGING_share are also required in the order shown in the command syntax diagram.

MDISK_share *nnnnn*
: specifies the relative share that minidisk I/O will use when it competes for SYSTEM-attached HyperPAV aliases in the control unit. The fraction of the total number of SYSTEM-attached HyperPAV aliases that CP will attempt to make available for use by minidisk I/O is *nnnnn* divided by the sum of the values of the MDISK_share and PAGING_share operands. The value of *nnnnn* ranges from 0 to 10000. When this operand is not specified, the value of MDISK_share is 0. When the value of MDISK_share is 0, minidisk I/O is not promised any share of the control unit's SYSTEM-attached HyperPAV aliases.

PAGING_share *nnnnn*
specifies the relative share that paging I/O will use when it competes for SYSTEM-attached HyperPAV aliases in the control unit. The fraction of the total number of SYSTEM-attached HyperPAV aliases that CP will attempt to make available for use by paging I/O is *nnnnn* divided by the sum of the values of the MDISK_share and PAGING_share operands. The value of *nnnnn* ranges from 0 to 10000. When this operand is not specified, the value of PAGING_share is 0. When the value of PAGING_share is 0, paging I/O is not promised any share of the control unit's SYSTEM-attached HyperPAV aliases.

Usage Notes

1. For any control units not specified in a CU statement, CP will attempt to establish the highest level of PAV mode that is supported by the functional capability of each control unit.
2. For each capability (HYPERPAV_allowed, PAV_allowed, NOPAV_allowed), the information on the CU statement is processed sequentially. If you specify more than one CU statement or you overlap ranges of subsystem identifiers, CP uses the last specification for each control unit. For example, if you specify:

   ```
   CU         HYPERPAV          1000-1fff,
              PAV               1800-18ff,
              HYPERPAV          1820-182f
   ```

   CP will allow PAV mode for control units 1800-181f and 1830-18ff, and will allow HyperPAV mode for control units 1000-17ff, 1820-182f, and 1900-1fff.
3. The MDISK_share and PAGING_share operands influence CP in choosing an I/O operation to be run on a SYSTEM-attached HyperPAV alias device. The settings of these operands have meaning to CP only when all SYSTEM-attached HyperPAV aliases that are associated with the control unit are currently busy running I/O and an alias becomes available.

   The default behavior for CP when an alias becomes available is to scan for I/O that is queued on the HyperPAV base devices on the control unit. The first I/O operation found that can be run on a SYSTEM-attached HyperPAV alias device is chosen for execution. This is the behavior when the values of MDISK_share and PAGING_share are 0. By specifying a non-zero value for MDISK_share, PAGING_share, or both, you specify an entitlement for the respective use. CP makes sure an entitlement is met before resorting to its default behavior.

   The entitlement computation rounds the value to the nearest whole number, specifically:

   - A value with a decimal component of less than .5 is rounded down.
   - A value with a decimal component of .5 or greater is rounded up.

   However, if the sum of the resulting entitlements for minidisk I/O and paging I/O exceeds the number of SYSTEM-attached HyperPAV aliases, both entitlements are rounded down and the default algorithm applies to the residual alias.

   To determine the alias activity for a HyperPAV base device, examine the device monitor data. To see minidisk I/O's and paging I/O's current shares and entitlements for a control unit, use the QUERY CU command with the ALIAS_Share operand. To obtain a complete list of alias devices that are associated with a control unit, use the QUERY CU command with the ALIASES operand.
4. The z/VM system programmer and the DASD management programmer must work together to make sure each logical subsystem (LSS) that is being used on the system has a unique SSID.

Examples

1. To have CP allow HyperPAV mode for control units 0000-0100, allow PAV mode for control units 2000-2100, and disallow aliases for control units 4100-4200, specify the following:

   ```
   CU         HYPERPAV          0000-1000,
              PAV               2000-2100,
              NOPAV             4100-4200
   ```

2. To have CP set a minidisk alias share of 100 and a paging alias share of 200 for control unit 4343, specify the following:

   ```
   CU HYPERPAV 4343 ALIAS MDISK_SHARE 100 PAGING_SHARE 200
   ```

   In this example, if the number of HyperPAV aliases is 17, the minidisk entitlement is 6 and the paging entitlement is 11.

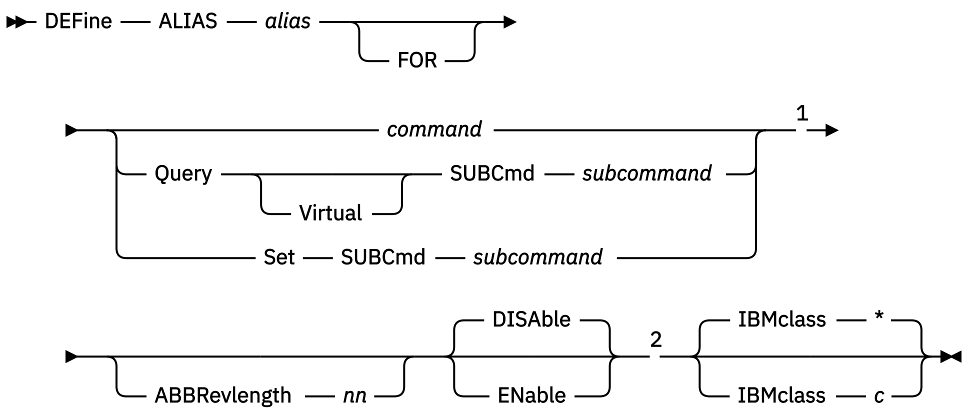
   The minidisk entitlement calculation is:

   ```
   (100 / (100 + 200)) * 17 = 17 / 3 = 5.67 (rounded up to 6)
   ```

   The paging entitlement calculation is:

   ```
   (200 / (100 + 200)) * 17 = 34 / 3 = 11.33 (rounded down to 11)
   ```

DEFINE ALIAS Statement



Notes:

[1] You can specify the following operands in any order.

[2] If the existing CP command has multiple versions, you must specify the IBMCLASS operand for CP to locate the correct version.

Purpose

Use the DEFINE ALIAS statement to define a new alias for an existing CP command on the system during initialization.

You can also define a new alias after initialization by using the DEFINE ALIAS CP command. For more information, see DEFINE ALIAS in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place DEFINE ALIAS statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

alias
: is the name of the alias that you are defining. The variable *alias* must be a 1-character to 12-character string.

command
: is the name of the existing CP command for which you are creating an alias. The variable *command* is a 1-character to 12-character string.

Query SUBCmd *subcommand*
: tells CP the name of the existing CP QUERY subcommand for which you are creating an alias. The variable *subcommand* is a 1-character to 12-character string.

Query Virtual SUBCmd *subcommand*
: tells CP the name of the existing CP QUERY VIRTUAL subcommand for which you are creating an alias. The variable *subcommand* is a 1-character to 12-character string.

Set SUBCmd *subcommand*
: tells CP the name of the existing CP SET subcommand for which you are creating an alias. The variable *subcommand* is a 1-character to 12-character string.

ABBRevlength *nn*
: is the length of the smallest acceptable abbreviation of the alias that you are defining. The variable *nn* is a decimal number between 1 and the length of the full alias name.

DISAble
: tells CP not activate this alias until you enable it. (For more information about enabling aliases, see Usage Note "5" on page 89.) If omitted, DISABLE is the default.

ENable
: tells CP to immediately activate this alias.

IBMclass *
: tells CP to define aliases for all versions of the specified command or subcommand. If omitted, IBMCLASS * is the default.

IBMclass *c*
: tells CP to define an alias for a specific version of the specified command or subcommand. The variable *c* can be any 1 of the following:

  A
  : this is a system-control command to be used by the primary system operator.

  B
  : this is a command for operational control of real devices.

  C
  : this is a command to alter host storage.

  D
  : this is a command for system-wide control of spool files.

  E
  : this is a command to examine host storage.

  F
  : this is a command for service control of real devices.

  G
  : this is a general-use command used to control the functions of a virtual machine.

  0
  : (zero) this command has no specific IBM class assigned.

Usage Notes

1. You can create many aliases for one command, but you cannot create an alias for an alias. For example, the CP MSG command is actually an alias for the CP MESSAGE command.

   ```
               +----------+
    MSG ----------->  | MESSSAGE |
               +----------+
   ```

   You can create another alias for the MESSAGE command.

   ```
               +----------+
    TELL ---------->  | MESSSAGE |
               +----------+
   ```

   But you cannot create an alias for MSG.

   ```
               +----------+
    TELL --> MSG -->  | MESSSAGE |
               +----------+
   ```

2. If you specify the QUERY, QUERY VIRTUAL, or SET operands, you are creating an alias for a subcommand, not a command. For example, suppose you created alias TUBE for the CP QUERY VIRTUAL GRAF command. To invoke your new alias, you would enter QUERY VIRTUAL TUBE, not QUERY TUBE or just TUBE.
3. When specifying an alias name, you can use special characters in the name. However, we do not recommend that you use the pattern matching characters (* and %). You can use these characters to define aliases, but they may seem confusing when you issue other commands that allow you to use the pattern matching characters because CP will interpret the % or * in your alias name as a pattern matching character.
4. If you try to define a minimum abbreviation that matches the abbreviation for an existing command, subcommand, or alias CP rejects your DEFINE statement. For example, if you created a "QUEUE" alias with a minimum abbreviation of 2, CP would reject your QUEUE alias because "QU" is an abbreviation for the QUERY command. In this case, you would need to specify a minimum abbreviation greater than or equal to 4 because the first 3 characters of QUERY and QUEUE are identical.
5. If you do not specify the ENABLE operand, the new alias is initially in a disabled state. To activate an alias after you define it, use the ENABLE COMMAND system configuration statement or the ENABLE COMMAND CP command. For information about the ENABLE COMMAND system configuration statement, see "ENABLE COMMAND / CMD Statement" on page 146. For information about the ENABLE COMMAND CP command, see ENABLE COMMAND / CMD in *z/VM: CP Commands and Utilities Reference*.
6. To deactivate an alias after you define it, use the DISABLE COMMAND system configuration statement or DISABLE COMMAND CP command. For information about the DISABLE COMMAND system configuration statement, see "DISABLE COMMAND / CMD Statement" on page 128. For information about the DISABLE COMMAND CP command, see DISABLE COMMAND / CMD in *z/VM: CP Commands and Utilities Reference*.
7. After an ALIAS NAME is defined, an ALIAS NAME cannot be used again. An ALIAS cannot be modified or eliminated. It can only be disabled. Only a SHUTDOWN or RESTART IPL will eliminate an ALIAS.
8. For information about creating an alias, see Defining an Alias Command in *z/VM: CP Exit Customization*.

Examples

1. To have CP define GOODNIGHT as an alias for the CP SHUTDOWN command and make it available immediately after initialization, use the following:

```
Define Alias goodnight For shutdown,      /* Goodnight, Gracie! */
                       AbbrevLength 5,
                       Enable
```

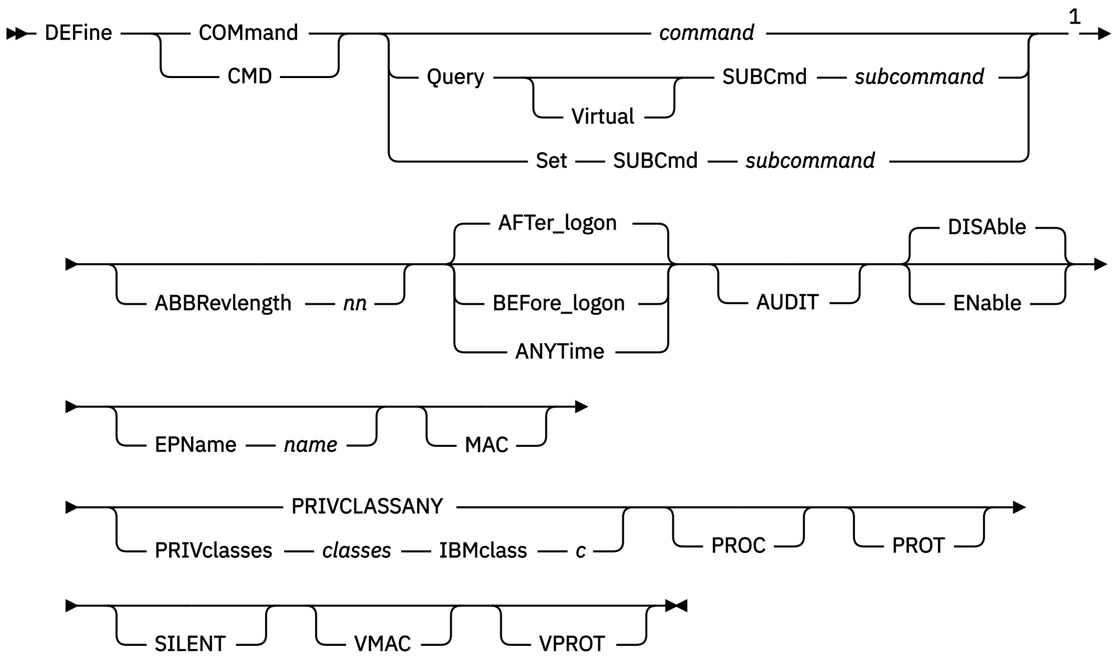
2. To have CP define TELL as an alias for the IBM class <ANY> version of the CP MESSAGE command and make it available immediately after initialization, use the following:

```
Define Alias tell For message, /* Create CP TELL command so users do not */
                  IBMclass 0,  /* get confused when they drop out of CMS */
                  Enable
```

3. To have CP define NUKE as an alias for all of the IBM class versions of the CP PURGE command and make it available immediately after initialization, use the following:

```
Define Alias nuke For purge,   /* Set it up so that users can “nuke”   */
                  IBMclass *,  /* instead of “purge”                   */
                  Enable
```

DEFINE COMMAND / CMD Statement



Notes:

[1] You must specify at least one of these operands. If you specify more than one operand, you can specify them in any order.

Purpose

Use the DEFINE COMMAND or CMD statement to define a new CP command or a new version (by IBM class) of an existing CP command on the system during initialization.

You can also define a new CP command after initialization by using the DEFINE COMMAND or CMD commands. For more information, see DEFINE COMMAND / CMD command in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place DEFINE COMMAND or CMD statements anywhere in the system configuration file. If you specify more than 1 statement with the same command or subcommand name, CP uses only the first statement. Subsequent DEFINE COMMAND or CMD statements do not redefine the command.

Operands

command
is the name of the command that you are defining. The variable *command* is a 1-character to 12-character string.

Query SUBCmd *subcommand*
tells CP the name of the CP QUERY subcommand that you are defining. The variable *subcommand* is a 1-character to 12-character string.

Query Virtual SUBCmd *subcommand*
tells CP the name of the CP QUERY VIRTUAL subcommand that you are defining. The variable *subcommand* is a 1-character to 12-character string.

Set SUBCmd *subcommand*
: tells CP the name of the CP SET subcommand that you are defining. The variable *subcommand* is a 1-character to 12-character string.

ABBRevlength *nn*
: is the length of the smallest acceptable abbreviation of the command or subcommand that you are defining. The variable *nn* is a decimal number between 1 and the length of the full command or subcommand name.

AFTer_logon
: tells CP that the command version you are defining will only be issued by users after they log onto the system. Most CP commands fall into this category. The default is AFTER_LOGON.

BEFore_logon
: tells CP that the command you are defining will only be issued by users before they log onto the system. For example, the CP DIAL command can only be used before logon.

  Note: If you specify BEFORE_LOGON, you must specify PRIVCLASSANY because CP cannot check privilege classes before a user logs on. Thus, you cannot specify BEFORE_LOGON with the PRIVCLASSES or IBMCLASS operands.

ANYTime
: tells CP that the command version you are defining can be issued by users both before and after they log onto the system. For example, many systems let users issue the CP MESSAGE or MSG commands before and after logon.

  Note: If you specify ANYTIME, you must specify PRIVCLASSANY because CP cannot check privilege classes before a user logs on. Thus, you cannot specify ANYTIME with the PRIVCLASSES or IBMCLASS operands.

AUDIT
: tells the external security manager (ESM) to audit the command that you are defining. When you audit a command, the ESM logs each attempt by users to issue this command.

DISAble
: tells CP not to call the entry points and external symbols associated with this CP command until you enable it. (For more information about enabling CP commands, see Usage Note "4" on page 93.) If omitted, DISABLE is the default.

ENable
: tells CP to immediately start calling the entry points and external symbols associated with this CP command.

EPName *name*
: tells CP the name of the entry point that contains the code to process the command. The variable *name* must be a 1-character to 8-character string. The first character must be alphabetic or one of the following special characters: dollar sign ($), number sign (#), underscore (_), or at sign (@). The rest of the string can be alphanumeric characters, the four special characters ($, #, _, and @), or any combination thereof.

  Note: If you are defining a new command, you must specify EPNAME. If you are defining a new version of an existing command, specifying EPNAME is optional. This is because CP only allows one entry point per command, regardless of how many versions that command has. So, when you define a new version of an existing command, CP already knows the entry point name.

  The QUERY and SET commands are the only exceptions, because they have subcommands. Note that CP only allows one entry point per subcommand, regardless of how many versions that subcommand has.

MAC
: tells CP to enable mandatory access control (MAC) for the command that you are defining. When MAC is enabled for your command, the external security manager (ESM) compares the security label of the user who issued your command to the security label of the resource or user that your command will affect. If you want the ESM to dynamically turn MAC on or off for this command, specify the VMAC operand.

PRIVCLASSANY
tells CP that users with any privilege class can issue the command that you are defining.

***PRIVclasses** classes*
tells CP that only users with 1 or more of the specified privilege classes can issue the command that you are defining. The variable *classes* is 1 or more privilege classes in the range A through Z, 1 through 6, or an asterisk (*). Privilege class * indicates all privilege classes (A-Z and 1-6).

Note:

1. If you want more than one privilege class, specify your classes in one string of characters. Do not separate the classes with blank spaces. For example, specify "`privclasses abc123`", not "`privclasses a b c 1 2 3`".
2. If you specify PRIVCLASSES, you must also specify IBMCLASS. You can specify these 2 operands in any order.

IBMclass *c*
tells CP what type of command you are defining. The variable *c* can be any 1 of the following:

A
this is a system-control command to be used by the primary system operator.

B
this is a command for operational control of real devices.

C
this is a command to alter host storage.

D
this is a command for system-wide control of spool files.

E
this is a command to examine host storage.

F
this is a command for service control of real devices.

G
this is a general-use command used to control the functions of a virtual machine.

Note: If you specify IBMCLASS, you must also specify PRIVCLASSES. You can specify these 2 operands in any order.

PROC
tells CP that, after performing the initial privilege class checks, your command processor will be responsible for any further calls to the external security manager (ESM) for the command that you are defining.

PROT
tells the external security manager (ESM) to protect the command that you are defining. When protection is enabled for your command, the ESM checks an access list to ensure that the user who issued your command is authorized to do so. If you want the ESM to dynamically turn protection on or off for your command, specify the VPROT operand.

SILENT
tells CP that the responses from the command you are defining can be suppressed by invoking it using the SILENTLY command. For more information, see SILENTLY in *z/VM: CP Commands and Utilities Reference*.

Note: Response suppression is supported only for the ATTACH, DETACH, and GIVE commands.

VMAC
gives the external security manager (ESM) the power to dynamically turn mandatory access control (MAC) on or off. If you specify MAC and VMAC for this command, you are enabling MAC and allowing the external security manager (ESM) to dynamically disable MAC. If you specify VMAC and you do not specify MAC for this command, you are disabling MAC and allowing the ESM to dynamically enable MAC.

VPROT

gives the external security manager (ESM the power to dynamically turn command access protection on or off. If you specify PROT and VPROT for this command, you are enabling protection and allowing the external security manager (ESM) to dynamically disable that protection. If you specify VPROT and you do not specify PROT for this command, you are disabling protection and allowing the ESM to dynamically enable that protection.

Usage Notes

1. For each existing CP command, CP has at least one command table entry block. If the command has more than one privilege class, CP has one command table entry block for each version of the command. The only exceptions to this are the QUERY and SET commands. CP has at least one command table entry block for each QUERY and SET subcommand.
2. When you define a new CP command or a new version of an existing CP command, you must supply CP with certain information about that command. The amount of information you must supply varies depending on what you are defining. You must always supply the command (or subcommand) information. You must also supply the following information:
   - When you add a new command, you must specify `EPNAME PRIVCLASSANY` or `EPNAME PRIVCLASSES IBMCLASS`.
   - When you add a new command version, you specify `PRIVCLASSANY` or `PRIVCLASSES IBMCLASS`.
3. To load the command processing code into the system execution space, use the CPXLOAD statement or CP command. For more information about the CPXLOAD statement, see "CPXLOAD Statement" on page 76. For more information about the CPXLOAD CP command, see CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
4. If you do not specify the ENABLE operand, the new CP command is initially in a disabled state. To activate a CP command during system initialization, use the ENABLE COMMAND or CMD statement. To activate a CP command after system initialization, use the ENABLE COMMAND or CMD CP command. For more information about the ENABLE COMMAND statement, see "ENABLE COMMAND / CMD Statement" on page 146. For more information about the ENABLE COMMAND CP command, see *ENABLE COMMAND / CMD* in *z/VM: CP Commands and Utilities Reference*.
5. To display the address of the CP command table entry block, the current IBM class, and the current privilege class for a specified CP command, use the LOCATE CMDBK command. For more information about the LOCATE CMDBK CP command, see *z/VM: CP Commands and Utilities Reference* .
6. To change the definition of an existing CP command during system initialization, use the MODIFY COMMAND or CMD statement. To change the definition of an existing CP command after system initialization, use the MODIFY COMMAND or CMD CP command. For more information about the MODIFY COMMAND statement, see "MODIFY COMMAND / CMD Statement" on page 188. For more information about the MODIFY COMMAND CP command, see *z/VM: CP Commands and Utilities Reference* .
7. Once defined, COMMANDS, SUBCOMMANDS, ALIASES, and DIAGNOSE codes cannot be DELETED. They can be altered in various appropriate ways but they remain in existence until a SHUTDOWN or RESTART IPL is done.
8. To deactivate a CP command during system initialization, use the DISABLE COMMAND statement. After system initialization, you can deactivate a CP command by using the DISABLE COMMAND CP command. For more information about the DISABLE COMMAND statement, see "DISABLE COMMAND / CMD Statement" on page 128. For more information about the DISABLE COMMAND CP command, see *z/VM: CP Commands and Utilities Reference* .
9. To remove the command processing code from the system execution space, use the CPXUNLOAD command. For more information about the CPXUNLOAD CP command, see *z/VM: CP Commands and Utilities Reference* .
10. When specifying a command or subcommand name, you can use special characters in the name. However, we do not recommend that you use the pattern matching characters (* and %). You can use these characters to define commands, but they may seem confusing when you issue the LOCATE CMDBK command. For example, suppose you define 2 new commands: CPU% — to calculate recent

CPU busy as a percentage, and CPU1 — to cause some action on real processor number 1. If you issue LOCATE CMDBK CPU%, CP displays information on both CPU% and CPU1, because CP interprets the % in your LOCATE command to be a pattern matching character.

11. If you try to define a minimum abbreviation that matches the abbreviation for an existing command or subcommand, CP rejects your DEFINE statement. For example, if you created a "QUEUE" command with a minimum abbreviation of 2, CP would reject your QUEUE command because "QU" is an abbreviation for the CP QUERY command. In this case, you would need to specify a minimum abbreviation greater than or equal to 4 because the first 3 characters of QUERY and QUEUE are identical.
12. For more information, see Defining an Alias Command in *z/VM: CP Exit Customization*.

Examples

1. To define a new command, TELL, with:
   - 3 versions:
     - Privilege class A, IBM class A, issued after logon
     - Privilege class B, IBM class B, issued after logon
     - Any privilege class, no IBM class, issued before and after logon,
   - No minimum abbreviation, and
   - The same entry point and syntax as the CP MESSAGE command,

   use the following statements:

```
Define Cmd tell EPName hcpxmgms,    /* Create class A "CP TELL"     */
                IBMClass a,         /*    command.                  */
                PrivClasses a

Define Cmd tell IBMClass b,         /* Create class B "CP TELL"     */
                PrivClasses b       /*    command.                  */

Define Cmd tell AnyTime,            /* Create "CP TELL" command for */
                PrivClassAny        /*    any class user, to use    */
                                    /*    before or after logon.    */
```

   Attention: Normally, it is not a good practice to define a CP command that has the same name as a command, exec, or function belonging to another z/VM subsystem. (This newly-defined command will interfere with the CMS TELL command.) However, in this case, we thought we would show you an example of how to create a "synonym" for a common CMS command that could be used by novice z/VM users when they accidentally drop out of their CMS session into CP.

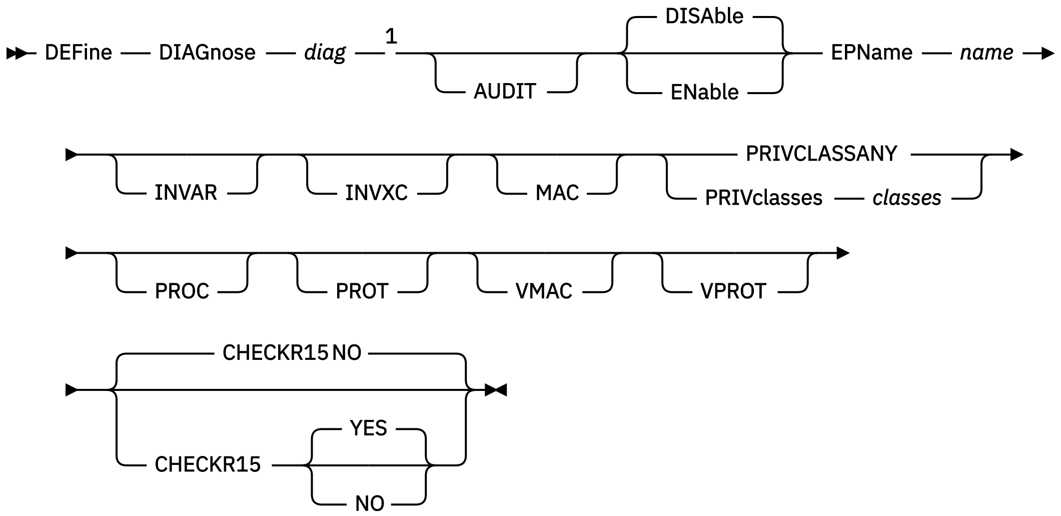
2. To define 2 new commands, CONTINUE and RESUME, which behave like the CP BEGIN command, use the following statements:

```
Define Cmd continue AbbrevLength 4,  /* Create "CP CONTINUE"        */
                    EPName hcpcmcbe, /*    command to use instead   */
                    IBMClass g,      /*    of CP BEGIN.             */
                    PrivClasses g

Define Cmd resume EPName hcpcmcbe,   /* Create "CP RESUME" command  */
                  IBMClass g,        /*    to use instead of CP     */
                  PrivClasses g      /*    BEGIN.                   */
```

   In this example, the CONTINUE and RESUME commands:
   - Have the same entry point and syntax as the CP BEGIN command,
   - Are IBM class G,
   - Are privilege class G, and
   - Can only be issued after logon.

Additionally, the CONTINUE command has a minimum abbreviation of 4 (CONT) and the RESUME command has no minimum abbreviation.

DEFINE DIAGNOSE Statement



Notes:

[1] You must specify at least one of these operands. If you specify more than one operand, you can specify them in any order.

Purpose

Use the DEFINE DIAGNOSE statement to define a new DIAGNOSE code on the system during initialization.

You can also define a new DIAGNOSE code on the system after initialization using the DEFINE DIAGNOSE command. For more information, see DEFINE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place DEFINE DIAGNOSE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

diag
: is the number of the DIAGNOSE code that you are defining. The variable *diag* is a hexadecimal number between X'0' and X'03FC' and must be a multiple of 4. The recommended variable range is a value between X'100' and X'1FC' because that is the range of diagnose codes reserved for customer use. All other DIAGNOSE code numbers are reserved for IBM use.

AUDIT
: tells the external security manager (ESM) to audit the DIAGNOSE code that you are defining. When you audit a DIAGNOSE code, the ESM logs each attempt by users to issue this DIAGNOSE code.

DISAble
: tells CP not to call the entry points and external symbols associated with this DIAGNOSE code until you enable it. (For more information about enabling DIAGNOSE codes, see Usage Note "3" on page 98.) If omitted, DISABLE is the default.

ENable
: tells CP to immediately start calling the entry points and external symbols associated with this DIAGNOSE code.

EPName *name*
: tells CP the name of the entry point that contains the code to process the DIAGNOSE code that you are defining. The variable *name* must be a 1-character to 8-character string. The first character must be alphabetic or one of the following special characters: dollar sign ($), number sign (#), underscore (_), or at sign (@). The rest of the string can be alphanumeric characters, the four special characters ($, #, _, and @), or any combination thereof.

INVAR
: tells CP not to process the DIAGNOSE code if the virtual machine that issues the DIAGNOSE code is in host access register mode.

INVXC
: tells CP not to process the DIAGNOSE code if the virtual machine that issues the DIAGNOSE code is in XC architecture mode.

MAC
: tells CP to enable mandatory access control (MAC) for the DIAGNOSE code that you are defining. When MAC is enabled for your DIAGNOSE code, the external security manager (ESM) compares the security label of the virtual machine that issued your DIAGNOSE code to the security label of the resource or user that your DIAGNOSE code will affect. If you want the ESM to dynamically turn MAC on or off for this DIAGNOSE code, specify the VMAC operand.

PRIVCLASSANY
: tells CP that users with any privilege class can issue the DIAGNOSE code that you are defining.

PRIVclasses *classes*
: tells CP that only users with 1 or more of the specified privilege classes can issue the DIAGNOSE code that you are defining. The variable *classes* is 1 or more privilege classes in the range A through Z, 1 through 6, or an asterisk (*). Privilege class * indicates all privilege classes (A-Z and 1-6).

  Note: If you want more than one privilege class, specify your classes in one string of characters. Do not separate the classes with blank spaces. For example, specify "`privclasses abc123`", not "`privclasses a b c 1 2 3`".

PROC
: tells CP that, after performing the initial privilege class checks, your DIAGNOSE code processor will be responsible for any further calls to the external security manager (ESM) for the DIAGNOSE code that you are defining.

PROT
: tells the external security manager (ESM) to protect the DIAGNOSE code that you are defining. When protection is enabled for your DIAGNOSE code, the ESM checks an access list to ensure that the user who issued your DIAGNOSE code is authorized to do so. If you want the ESM to dynamically turn protection on or off for your DIAGNOSE code, specify the VPROT operand.

VMAC
: gives the external security manager (ESM) the power to dynamically turn mandatory access control (MAC) on or off.

VPROT
: gives the external security manager (ESM) the power to dynamically turn DIAGNOSE code access protection on or off. If you specify PROT and VPROT for this command, you are enabling protection and allowing the external security manager (ESM) to dynamically disable that protection. If you specify VPROT and you do not specify PROT for this command, you are disabling protection and allowing the ESM to dynamically enable that protection.

CHECKR15 YES
CHECKR15 NO
: indicates whether the diagnose router should check register 15 upon return from the diagnose handler.

Usage Notes

1. If you do not specify the ENABLE operand, a new DIAGNOSE code is initially in a disabled state after being defined. CP treats disabled DIAGNOSE codes as if they were never defined. If you try to use a disabled DIAGNOSE code in a program, CP will give you a program check specification exception.
2. To load the DIAGNOSE processing code into the system execution space, use the CPXLOAD statement or CPXLOAD CP command. For more information, see "CPXLOAD Statement" on page 76. See also CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
3. To activate a new DIAGNOSE code after defining it, use the ENABLE DIAGNOSE statement or ENABLE DIAGNOSE CP command. For more information, see "ENABLE DIAGNOSE Statement" on page 148. See also ENABLE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
4. To change the definition of an existing DIAGNOSE code, use the MODIFY DIAGNOSE statement or command. For more information, see "MODIFY DIAGNOSE Statement" on page 192. See also MODIFY DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
5. To display information about a DIAGNOSE code (status, entry point name, and privilege class) after initialization, use the QUERY DIAGNOSE command. For more information, see QUERY DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
6. To display the address of the CP DIAGNOSE code table block for a DIAGNOSE code after initialization, use the LOCATE DGNBK command. For more information, see LOCATE DGNBK in *z/VM: CP Commands and Utilities Reference*.
7. To deactivate a DIAGNOSE code after defining it, use the DISABLE DIAGNOSE statement or command. For more information, see "DISABLE DIAGNOSE Statement" on page 130. See also DISABLE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
8. Once defined, DIAGNOSE codes cannot be deleted. They can be altered in various appropriate ways, but they remain in existence until a SHUTDOWN or RESTART IPL is done.
9. To remove the DIAGNOSE processing code from the system execution space, use the CPXUNLOAD command. For more information, see CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.
10. Many external security managers (ESMs) do not support DIAGNOSE codes above X'03FC'. For this reason, CP does not support DIAGNOSE codes above X'03FC'. The DIAGNOSE codes between X'0000' and X'03FC' are divided as follows:

    X'0000' to X'00FC'
    Reserved for IBM use

    X'0100' to X'01FC'
    Reserved for customer use

    X'0200' to X'03FC'
    Reserved for IBM use.
11. When CHECKR15 YES is specified, the diagnose router will check register 15 on return from the diagnose handler. If register 15 contains:

    RC = 0
    Processing was successful. Complete the guest instruction.

    RC = 4
    Processing failed due to a condition which would cause a guest program check. Simulate guest program interruption passed in R0.

    RC = 8
    Nullify the instruction.

    RC = 12
    Present the machine check then nullify the instruction. R2 will contain the address of the MCRBK which will contain the machine check information.

    RC = 16
    Generate machine check for processing damage, then go to HCPENDOP to terminate the instruction.

RC = 20
Present the machine check, then go to HCPENDOP to terminate the instruction. R2 will contain the address of the MCRBK, which contains machine check information.

RC = 24
Issue error message or soft abend for paging I/O error, then nullify the instruction. R1 has the message or abend number.

If a return code is invalid (negative, not a multiple of 4 or too big (RC > 24)), then a soft abend will occur.

12. For more information about user-defined DIAGNOSE codes, see Defining and Modifying Commands and Diagnose Codes in *z/VM: CP Exit Customization*.

Examples

1. To have CP define a general purpose DIAGNOSE code (X'10C') that will be processed by entry point HCPSRC00, use the following:

```
Define Diagnose 10c EPname hcpsrc00,
                    PrivClasses g
```

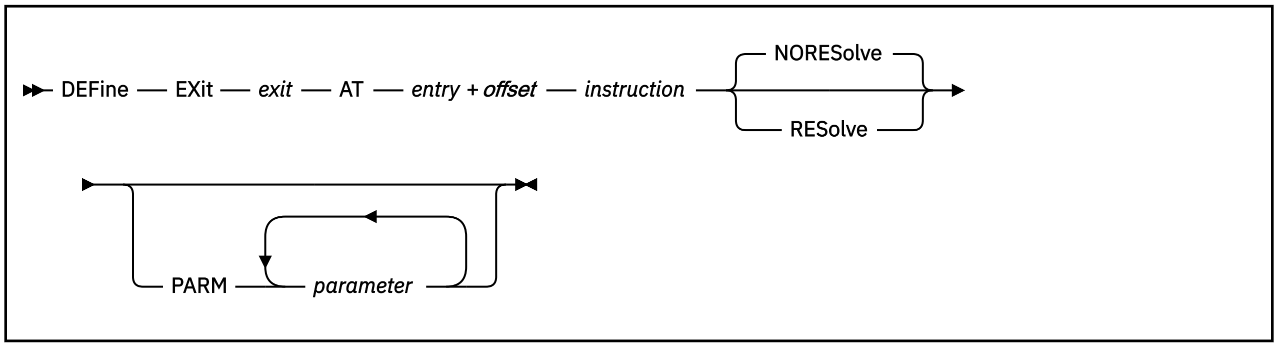
2. To have CP define DIAGNOSE code X'100' and make it available to users with privilege classes C or E, use the following:

```
Define Diagnose 100 EPname qwerty,
                    PrivClasses ce
```

3. To have CP define DIAGNOSE code X'18C' and make it available to all users, use the following:

```
Define Diagnose 18c EPname hcpsrc00,
                    PrivClassAny
```

DEFINE EXIT Statement



Purpose

Use DEFINE EXIT to define a new exit point in CP during initialization.

You can also define a new CP exit point after initialization by using the DEFINE EXIT command. For more information, see DEFINE EXIT in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place DEFINE EXIT statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

exit
: is the number of the exit point you are defining. This value must be a hexadecimal number between X'0' and X'FFFF'. The recommended value is between X'F000' and X'FFFF', because that range is reserved for customer use. All other exit numbers are reserved for IBM, vendor, or general use.

AT *entry + offset instruction*
: identifies the location of the exit point you are defining and the instruction that is located at the exit point. The variable *entry* must be a 1-character to 8-character string. The first character must be alphabetic or one of the following special characters: $ (dollar sign), # (number sign), _ (underscore), or @ (at sign). The rest of the string can be alphabetic or numeric characters, the four special characters ($, #, _, or @), or any combination. The variable *offset* must be a 1-character to 4-character even hexadecimal number between X'0' and X'FFFE'. The variable *instruction* must be a 2-, 4-, or 6-character hexadecimal number.

NORESolve
: tells CP to resolve the entry points associated with this exit number the first time they are called. This is the default.

RESolve
: tells CP to resolve the entry points associated with this exit number when the association is first established. Any existing associated entry points are resolved immediately.

PARM *parameter*
: is a list of one or more parameters to be supplied to the exit. Five kinds of tokens can be used to define a parameter:

1. Addresses: strings up to eight characters long, consisting of the hexadecimal digits 0 through 9 and A through F.
2. General Registers: strings beginning with G or R, followed by a decimal number between 0 and 15 or a hexadecimal digit, designating the contents of a general register.

3. Indirection: a percent sign (%), which causes the contents of an address or the contents of an address in a register to be used instead of the address or register contents itself.
4. Arithmetic: a plus sign (+) or minus sign (-).
5. Displacement: strings of up to four hexadecimal digits.

Each parameter string specifies how to combine these tokens to generate a parameter value to be passed to an exit routine. The following is a Backus-Naur definition of the syntax of a parameter:

```
<parameter> ::= <anchor> | <anchor><vector>
   <anchor> ::= <reg> | 0...FFFFFFFF | <anchor>%
   <vector> ::= <modifier> | <vector>% | <vector><modifier>
 <modifier> ::= +<disp> | -<disp> | +<reg> | <-<reg>
      <reg> ::= G<digit> | R<digit>
    <digit> ::= 0...15 | 0...9, A...F
     <disp> ::= 0...7FFF
```

Usage Notes

1. To load the exit point code into the system execution space, use the CPXLOAD statement or command. For more information, see "CPXLOAD Statement" on page 76. See also CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
2. To associate one or more entry points or external symbols with a specific exit point and to enable or disable that exit point, use the ASSOCIATE EXIT statement or command. For more information, see "ASSOCIATE EXIT Statement" on page 58. See also ASSOCIATE EXIT in *z/VM: CP Commands and Utilities Reference*. You can also use the ASSOCIATE EXIT statement to change the entry points and external symbols that are associated with a specific entry point.
3. To activate a new exit after defining it, use the ENABLE EXITS statement or command. For more information, see "ENABLE EXITS Statement" on page 150. See also DEFINE EXIT in *z/VM: CP Commands and Utilities Reference*.
4. To change the definition of an existing dynamic exit point, or remove the exit point from the system, use the MODIFY EXIT statement or command. For more information, see "MODIFY EXIT Statement" on page 195. See also MODIFY EXIT in *z/VM: CP Commands and Utilities Reference*.
5. To display status and usage statistics information about a specific exit point after initialization, use the QUERY EXITS command. For more information, see QUERY EXITS in *z/VM: CP Commands and Utilities Reference*.
6. Exit numbers are allocated as follows:
   - X'0000' to X'7FFF' are reserved for IBM use.
   - X'8000' to X'EFFF' are reserved for vendor and general use.
   - X'F000' to X'FFFF' are reserved for private customer use.
7. The RESOLVE option ensures that the entry names associated with an exit point are defined.
8. Each exit is passed a parameter list that begins with three standard parameters, as described in *z/VM: CP Exit Customization*. Additional parameters, specified by the PARM operand, are optional and follow the first three in the order in which they are specified.
9. Errors (for example, addressing exceptions) during evaluation of user-defined parameters when an exit is being invoked cause CP to abend.
10. For more information about user-defined exit points, see Benefits of using CP exits in *z/VM: CP Exit Customization*.

Examples

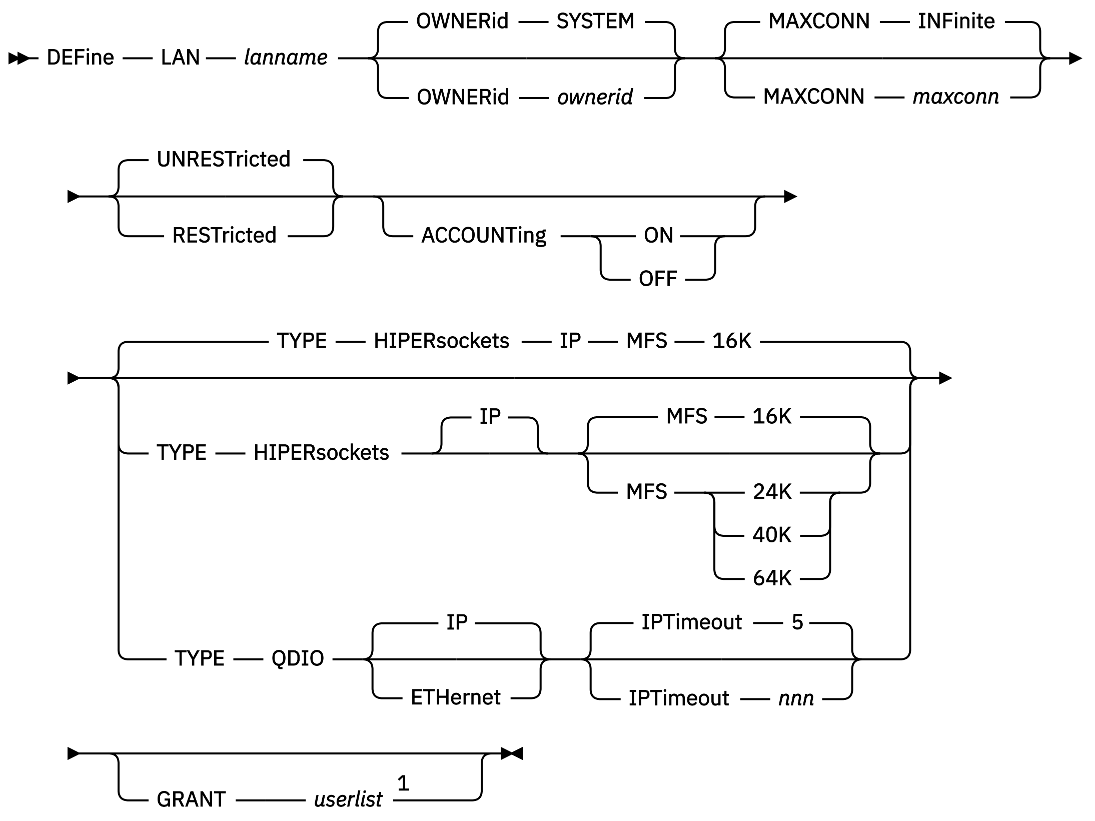
1. To define an exit that will be entered every time a MESSAGE command is issued, use the following:

```
Define Exit f422 At hcpxmgms + 4 41700000   /* Exit from MESSAGE cmd */

Enable Exits f422                           /*                       */
```

2. To define exit X'F800', pass it two additional parameters, and associate it with entry point QWERTY, use the following:

```
Define Exit f800 At hcplog + 7ce 41204028,  /*                       */
                 Parm gl+8% g4              /*                       */

Associate Exit f800 EPname qwerty           /*                       */

Enable Exits f800                           /*                       */
```

DEFINE LAN Statement



Notes:

[1] You can specify up to 20 *UserIDs* in the *userlist*.

Purpose

Use the DEFINE LAN statement to create a guest LAN which can be shared among virtual machines on the same VM system. Each guest LAN segment is identified by a unique combination of *ownerid* and *lanname*. A VM user can create a simulated network interface card (NIC) and connect it to this LAN segment. Alternatively, a VM user can create a virtual network adapter (with the CP DEFINE NIC command) and connect it to this LAN (with the COUPLE command). For more information, see "NICDEF Directory Statement" on page 565. See also DEFINE NIC and COUPLE in *z/VM: CP Commands and Utilities Reference*. For more information about virtual networking options in VM, see *z/VM: Connectivity*.

The MODIFY LAN statement can be used to add additional users to be included in the initial access list of a RESTRICTED LAN. For more information, see "MODIFY LAN Statement" on page 198.

You can also define a LAN after system initialization by using the DEFINE LAN command. For more information, see DEFINE LAN in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place DEFINE LAN statements anywhere in the system configuration file. If you specify more than one statement with the same *lanname* operand, the first definition will be accepted and subsequent statements will receive an error.

Operands

lanname
: is the name of the new emulated LAN (Local Area Network). The *lanname* is a single token (1-8 alphanumeric characters). The combination of *ownerid* and *lanname* will identify this LAN for subsequent commands.

OWNERid *ownerid*
OWNERid SYSTEM
establishes the owner of the new LAN.

MAXCONN INFinite
MAXCONN *maxconn*
sets the maximum number of simultaneous adapter connections permitted at any given time. When MAXCONN is specified as INFinite, there is no limit on the number of connections. Any other value, *maxconn*, limits the number of simultaneous connections to a decimal number in the range of 1–1024 (inclusive).

If MAXCONN is omitted, the default is MAXCONN INFinite.

RESTricted|UNRESTricted
sets the type of authorization required to connect to this LAN. Options are:

RESTricted
Defines a LAN with an access list to restrict connections. The LAN owner will use the **SET LAN** command to GRANT or REVOKE access to specific VM users (by userid). The **COUPLE** command will only allow authorized users (those on the access list) to connect a simulated adapter to a RESTRICTED network.

UNRESTricted
Defines a LAN with no access list. When CP is in control of the LAN, connections to this LAN are not restricted by user ID. All users connected to this LAN will be able to use PROMISCUOUS Mode. If an External Security Manager (ESM) is in control of the LAN, the ESM may restrict access.

When neither option is specified, the default is to define an **UNRESTricted** LAN.

ACCOUNTing ON
ACCOUNTing OFF
Allows a system administrator to control whether accounting records are created for the LAN being defined. The default setting may be changed by the VMLAN statement in the system configuration file, and queried by QUERY VMLAN.

TYPE HIPERsockets
defines a guest LAN for use by simulated HiperSockets adapters. A HiperSockets LAN can only accept connections from a simulated HiperSockets adapter.

If **TYPE** is omitted, the default is **TYPE HIPERsockets**.

MFS 16K
MFS 24K
MFS 40K
MFS 64K
sets the Maximum Frame Size (MFS) for adapters on this network. When an adapter is connected to this LAN, it will adopt the network MFS. The MFS value determines the amount of storage to be allocated for internal structures, and limits the effective MTU size (Maximum Transmission Unit) for the coupled adapters. For general internet communications, a lower MFS is probably better. However, a high MFS may provide higher data transfer rates for applications that are capable of using larger packet sizes.

If MFS is omitted, the default for HiperSockets is 16 KB. The MFS operand is not valid for QDIO but the effective MFS is 8992 for a QDIO adapter.

TYPE QDIO
defines a guest LAN for use by simulated QDIO adapters. A QDIO LAN can only accept connections from a simulated QDIO adapter.

IP | ETHernet
: indicates whether the transport for the LAN is ETHERNET or IP. An ETHERNET LAN operates at the Layer 2 level of the OSI model.

IPTimeout *nnn*
: indicates the length of time in minutes that a transient IP address table entry remains in the IP address table for the virtual network. A transient entry is an entry added to the virtual network by one guest on behalf of another. This value has no meaning for ETHERNET networks.

  nnn is a number from 1 to 240. 5 minutes is the default value.

GRANT *userlist*
: defines the initial list of users to be included in the Initial Access list of a RESTRICTED LAN. If the GRANT operand is omitted, the default is to GRANT the LAN owner. You can specify up to 20 users in this list. If selected, the GRANT operand must be the last operand on this statement.

Usage Notes

1. A Class B user can invoke the SET LAN and DETACH LAN commands to operate on a SYSTEM LAN (or any LAN regardless of ownership).
2. When a RESTRICTED LAN is defined, the owner is automatically added to the access list.
3. A LAN segment created by the DEFINE LAN statement will be "persistent" (that is, it will survive LOGOFF of the owner). It can only be removed by an explicit DETACH LAN command.
4. MODIFY LAN cannot be used to change the type of transport. The LAN will need to be redefined.
5. The TYPE QDIO guest LAN supports two modes of operation for data transport in support of both TCP/IP and non-IP based applications. In deciding which mode to deploy for your network, some things to consider about deploying an ETHERNET virtualized LAN segment are:

   - Do your servers or applications need to have their own unique MAC addresses (load balancers)?
   - Do you plan to deploy non-IP based applications on your network (SNA or NetBIOS, for example)?
   - Do you want to build a virtual LAN segment that operates closely to its physical counterpart?

   The key attributes of each transport mode with their operational characteristics are as follows:

   - ETHERNET (Layer 2)
     - Supports all applications that deploy ETHERNET (IEEE 802).
     - ETHERNET frames are transported on the LAN segment.
     - All destinations are identified by MAC address.
     - MAC addresses are locally administered by the LAN administrator through z/VM CP commands or configuration statements.
     - Each host connection is identified by a single MAC address.
     - This is a Link Layer transport, in which all hosts maintain their respective ARP caches.
     - VLAN tagging resides within the ETHERNET frames per IEEE 802.1Q specifications.
   - IP (Layer 3)
     - Supports IP for TCP/IP applications only.
     - IP packets are transported on the LAN segment.
     - All destinations are identified by IP addresses.
     - IP address assignments are set by the host running in the guest virtual machine.
     - Each host may have more than one IP address (multi-homed).
     - This is Link Layer independent (that is, no MAC addresses).
     - VLAN tagging resides in internal QDIO headers.

A guest LAN operates in only one mode for a given instance. For example, if the LAN is configured as IP, then all communications on the LAN segment must be IP based. The same is also true when the LAN is configured in ETHERNET mode.

Examples

1. To create a user LAN named QDIO that will allow up to 16 connections, specify the following:

   ```
      define lan type qdio maxconn 16
   ```

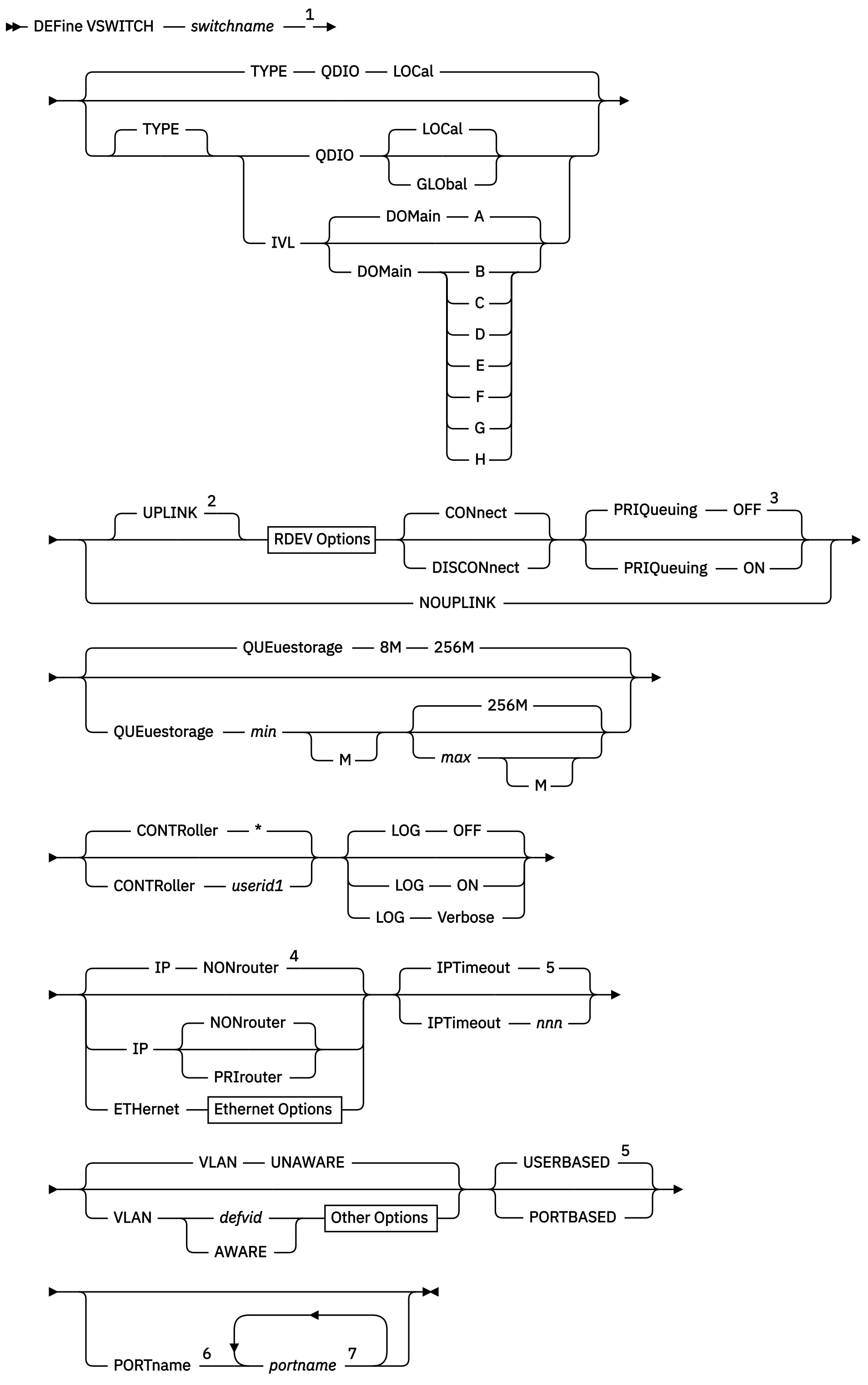
2. To create a system LAN named INEWS that allows up to 100 connections, specify the following:

   ```
      define lan inews ownerid system maxconn 100
   ```

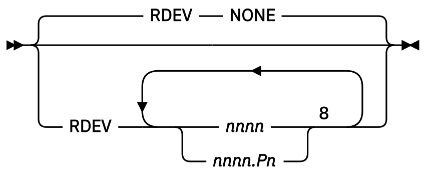
3. To create a SYSTEM LAN named INEWS that will allow up to 100 connections and will not have accounting records created, specify the following:

   ```
   cp define lan inews ownerid system maxconn 100 accounting off
   ```

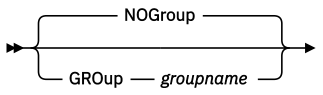
DEFINE VSWITCH Statement

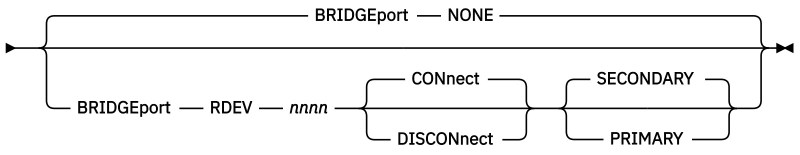


RDEV Options

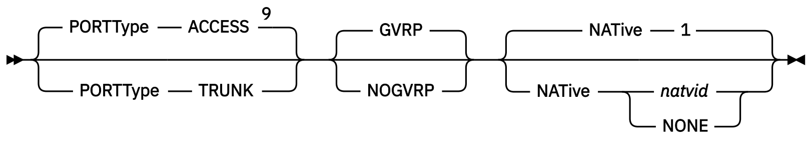


Ethernet Options





Other Options



Notes:

[1] You can specify the operands in any order, as long as *switchname* is the first operand specified, and *portname* is the last operand specified, if applicable.

[2] Use caution with the RDEV, CONNECT, and DISCONNECT operands, as they can apply to either the UPLINK or the BRIDGEPORT connection. The UPLINK keyword must be used if any of these operands are specified following the BRIDGEPORT keyword, but are intended to apply to the UPLINK connection.

[3] PRIQueuing ON is the default and only allowed option for a TYPE IVL virtual switch.

[4] ETHernet is the default and only allowed option for a TYPE IVL virtual switch.

[5] PORTBASED is the default and only allowed option for a TYPE IVL virtual switch.

[6] The PORTNAME operand is deprecated and is not recommended for use.

[7] You can specify a maximum of 3 port names.

[8] You can specify a maximum of 3 real device numbers.

[9] Port Type ACCESS is the default and only allowed option for a TYPE IVL virtual switch.

Purpose

Use the DEFINE VSWITCH statement or command to create a CP system-owned switch (a virtual switch) to which virtual machines can connect. Each switch is identified by a *switchname*. A z/VM user can create the appropriate simulated network interface card (NIC) and connect it to the virtual switch. z/VM guests that are connected to the same virtual switch can exchange messages by using the same communication software that they would use to drive a real NIC. More formation about virtual switches and virtual networking options in z/VM is available: See *Virtual Switch Priority Queuing Function*.

How to Specify

Include as many statements as needed; they are optional. You can place DEFINE VSWITCH statements anywhere in the system configuration file. If you specify more than one statement with the same *switchname* operand, the first definition will be accepted and subsequent statements will receive an error.

Operands

switchname
is the name of the new virtual switch. The *switchname* is a single token (1–8 alphanumeric characters) that identifies this virtual switch for subsequent commands.

TYPE
specifies the type of virtual switch to be created, specifically the hardware and protocol the virtual switch will emulate.

QDIO

defines a simulated virtual switch that uses QDIO or EQDIO devices to create a network. The network is comprised of simulated QDIO adapters that reside on the same z/VM system and real network devices that are located on an external or physical network.

A QDIO type virtual switch can accept connections from only a simulated QDIO adapter. External connectivity to network devices on a physical network is achieved by using the RDEV or GROUP keywords.

IVL
defines an Inter-VSwitch Link which provides the communication facility to implement an IVL domain. An IVL domain is a grouping of up to 16 systems running z/VM connected by an IVL LAN segment. All the active members within an IVL domain provide control operations that support the creation and management of shared virtual networking components such as Shared Port Groups. The IVL virtual switch provides communication infrastructure to exchange control information and data necessary to manage global networking objects that can span multiple systems running z/VM. The IVL virtual switch must be defined with external connectivity (using the RDEV or GROUP keywords) and its UPLINK connection must remain operational in order to support global networking objects.

Only one IVL virtual switch may be created on a system running z/VM.

Guest NICs may not be coupled to an IVL virtual switch.

DOMain [A | B | C | D | E | F | G | H]
defines the domain to which the IVL virtual switch belongs. See usage note “18” on page 120 for more information about an IVL virtual switch.

GLObal
identifies this virtual switch as a member of a global virtual switch. A global virtual switch is a collection of virtual switches that share the same name and the same networking characteristics. This collection of virtual switches spans multiple systems running z/VM but logically operates as a single switch.

Global virtual switches using a shared port group must reside on LPARs in the same CEC.

Creation of Global virtual switches will be deferred until IVL virtual switch UPLINK port connects the host to the IVL domain. Message HCP3178I will be displayed for each deferred Global virtual switch. Subsequent MODIFY VSWITCH statements with the same *switchname* will also be deferred.

See usage note “17” on page 120 for more information about a global virtual switch.

LOCal
LOCAL means that the virtual switch is not a member of a global virtual switch.

UPLINK
enables and specifies connectivity for a virtual switch UPLINK port. The UPLINK port is a special port that typically is used to connect the virtual switch to a physical switch, essentially bridging the virtual switch's simulated network to a physical network.

RDEV *nnnn*
RDEV *nnnn*.P*n*

specifies a real device to be used as an UPLINK port that connects the virtual switch to the real hardware LAN segment.

Specify each real device as a hexadecimal number. The specification can include a hexadecimal port number that is prefixed by lower case or upper case letter P. If no port number is included, it defaults to port 0. The port number follows the device number and is separated by a period.

For example, to specify port 1 for real device 300, specify 300.P1. The port number value is limited by how many ports the adapter hardware feature supports.

The following table describes the parameters for each device type:

Device type	Number of Devices	Device number range	Port number range
QDIO	3	X'0001' - X'FFFD'	X'0' - X'F'
EQDIO	1	X'0001' - X'FFFF'	X'0'

You can specify a maximum of three real device numbers. If you specify more than one device number, then each must be separated from the others by at least one blank.

When the virtual switch is defined with the GROUP attribute for an exclusive port group, any devices identified by the RDEV keyword are used for failover in the event of a real switch failure of the link aggregation group. Failover in this environment is to a single device that is connected to a second real switch.

Each OSA-Express (QDIO) real device number represents a trio of devices that cooperate to provide a complete network interface. In contrast, each Network Express (EQDIO) real device number represents a complete network interface.

For example, a specification of a QDIO device as RDEV 100 actually uses device numbers 100, 101, and 102 to provide a connection to the real hardware LAN segment. In contrast, a specification of an EQDIO device as RDEV 200 uses only device number 200 to provide the real hardware LAN connection.

When an RDEV list is specified, one of those devices represents the active interface and any other devices represent backup interfaces. For example, a specification of RDEV 111 222 333 means that the first device (QDIO devices 111-113 or EQDIO device 111) would probably become the active connection to the real hardware LAN segment. Unless the GROUP option is specified, devices 222 and 333 represent the interfaces to use as backup to the active device interface. If there is a problem with the connection via RDEV 111, RDEV 222 is used next. If there is a problem with the connection via RDEV 222, RDEV 333 is used next.

The list feature provides dynamic recovery for QDIO or EQDIO device failures when the VSWITCH command is issued without the GROUP attribute or for real switch failures when the VSWITCH command is issued with the GROUP attribute. (Failure of an OSA adapter in an aggregated group is automatically handled by the virtual switch. The virtual switch transfers the data flow to the remaining OSA adapters in the group.)

Including both QDIO and EQDIO devices in an RDEV list at the same time is not supported. Only devices of one type will be activated, while devices of the other type will display an error message.

RDEV NONE means that the virtual switch does not connect to the real LAN segment when defined with NOGROUP. When the virtual switch is defined with GROUP, then RDEV NONE means that there is no link aggregation group failover in the event that the real switch should fail.

The RDEV operand cannot be used to specify device numbers when the virtual switch is configured to use a shared port group. MAC address takeover is managed by the shared port group to maintain connectivity after a failure.

Restriction: The number of devices identified by the RDEV keyword combined with the number of devices in an associated GROUP cannot exceed eight devices.

CONnect
indicates that the currently configured virtual switch UPLINK port must be activated, and traffic can flow through the specified UPLINK ports devices.

DISCONnect

indicates that the currently configured virtual switch UPLINK port must not be activated, and that no traffic is to flow through the specified UPLINK ports device(s).

A virtual switch can be functional without a connection to a real LAN segment, and traffic flows only between virtual machines coupled to the virtual switch.

PRIQueuing

enables or disables guest priority queuing support on all outbound data transmissions from the virtual switch uplink port to an external network. Priority queuing is a capability of the OSA adapter when multiple output queues are defined for a single network connection. Each queue is weighted by a priority on how often it gets serviced. The highest priority queue is serviced first and more often followed by the next highest priority and so on. No queue is starved and all will get serviced at some point by the OSA adapter. More information about virtual switch exploitation of priority queuing is available: See *Virtual Switch Priority Queuing Function*in z/VM: Connectivity.

If PRIQueuing is to be enabled on a virtual switch, then the OSA-Express adapters that are configured to the virtual switch's uplink port must be configured by IOCP to enable the feature within the adapter (PQ_ON). In contrast, Network Express adapters (OSH devices) are always enabled for priority queuing. Even when a device is enabled for priority queuing, a virtual switch can be defined or be set with PRIQUEUING OFF.

For more information, see DEFINE CHPID / PATH, SET PORT GROUP, and SET VSWITCH in z/VM: CP Commands and Utilities Reference. See also NICDEF Directory Statement, MODIFY PORT Statement, and MODIFY VSWITCH Statement in z/VM: CP Planning and Administration.

OFF

The virtual switch will not exploit priority queuing. A single input queue is established for inbound transmissions from the external network and a single output queue is established for outbound transmissions. All outbound data to the external network is transmitted with equal priority. This is the default for TYPE QDIO virtual switches.

The OFF option is not allowed for a TYPE IVL virtual switch.

ON

The virtual switch will exploit priority queuing. If the OSA adapter that is used by a virtual switch uplink port is enabled for priority queuing, then z/VM will establish one input queue and four output queues when activating its network connection. This will allow z/VM to transmit data to the external network at four different priorities. CP will use the highest priority queue for control and management traffic. The other three queues (low, normal and high) can be used for virtual NICs' network connections. This is the default for TYPE IVL virtual switches.

For an IVL virtual switch, z/VM will attempt to establish an active network connection with the first OSA device specified. The result depends on the device type and configuration:

- If the device is an OSA-Express adapter, then the result depends also on the adapter configuration.
  - If the adapter is configured with priority queuing (PQ_ON), then z/VM will establish an active network connection and use priority queuing.
  - If the adapter is configured without priority queuing (PQ_OFF), then z/VM will establish an active network connection and force virtual switch priority queuing OFF. A warning message will be displayed to inform the customer to configure priority queuing via IOCP.
- If the device is a Network Express adapter, then the device is always enabled for priority queuing. z/VM will establish an active network connection and use priority queuing.

For a non-IVL virtual switch, the result also depends on device type and configuration:

- If the device is an OSA-Express adapter, then the result depends also on the adapter configuration.
  - If the adapter is configured with priority queuing (PQ_ON), then z/VM will establish an active network connection and use priority queuing.

  - If the adapter is configured without priority queuing (PQ_OFF), then z/VM will not establish an active network connection and an error message will be displayed.
- If the device is a Network Express adapter, the device is always enabled for priority queuing. z/VM will establish an active network connection and use priority queuing.

The policy used to select which priority a specific datagram is transmitted to the external network is determined by the type of virtual switch.

For an IVL virtual switch, the priority of outbound transmissions is handled by z/VM. IVL management traffic will be queued and transmitted on a high priority queue, and encapsulated production data failover traffic will be sent on a lower priority queue. The default for TYPE IVL virtual switches is PRIQueuing ON so that IVL network management data and encapsulated production data can be prioritized appropriately. If an OSA-Express device is used as an IVL Vswitch uplink, it is strongly recommended to have priority queueing enabled.

For virtual switches other than IVL type, the priority can be set to low, normal or high for all packets sent from a guest NIC's network connection to an external network via the SET VSWITCH command. For more information, see the PQUPLINKTX operand in SET VSWITCH in z/VM: CP Planning and Administration.

NOUPLINK

indicates the virtual switch will never have connectivity to a physical network through the UPLINK port. This option removes the UPLINK port from the virtual switch. Once the UPLINK port is removed, it can never be added back to the virtual switch.

Defining a virtual switch UPLINK port with either the RDEV or GROUP operands while also removing the UPLINK port with the NOUPLINK operand will cause the command to fail.

The NOUPLINK option is not allowed for a TYPE IVL virtual switch.

QUEuestorage *min*M *max*M

specifies high and low limits for the amount of memory that CP and the Queued Direct I/O Hardware Facility can use for buffers for each data connection.

The amount of memory is defined as a range from a low limit (*min*) to a high limit (*max*). The limits are expressed in megabytes and can include the megabyte abbreviation suffix (M or m). A blank space is permitted between the number and the suffix. The low limit can be 1M-256M. The high limit can be 1M-256M and cannot be less than the low limit.

When only one value is specified, it is interpreted as the low limit, and the high limit defaults to 256M. If the QUEUESTORAGE operand is omitted, the default range is 8M-256M.

When an OSA-Express (QDIO) adapter is active, the high limit is ignored and only the low limit determines the amount of memory that can be consumed for QDIO buffers on a single data connection. Because the OSA-Express (QDIO) interface can operate with no more than 8M of memory, if the QUEUESTORAGE low limit (*min*) is higher than 8M, then the current value will be 8M instead of *min*. Although the current value in this case is outside the configured range, it is not considered an error.

When a Network Express (EQDIO) Adapter is active, the amount of memory allocated for EQDIO buffers will vary within the specified range (*min*-*max*). The current value is adjusted based on the level of network demand.

When multiple network devices are defined in a link aggregation group, then each device port within the group adheres to the same specifications.

The following examples of the QUEUESTORAGE operand compare the memory usage for QDIO and EQDIO devices.

Table 9. QUEUESTORAGE examples

QUEUESTORAGE fragment	Range of memory usage for QDIO device	Range of memory usage for EQDIO device
QUEUESTORAGE	8M	8M-256M
QUEUESTORAGE 4M	4M	4M-256M
QUEUESTORAGE 4M 6M	4M	4M-6M
QUEUESTORAGE 12 M 64 M	8M	12M-64M
QUEUESTORAGE 256M	8M	256M

CONTRoller *
CONTRoller *userid1*

identifies the z/VM user ID that controls the OSA-Express device that is connected to the virtual switch at the device number identified by *rdev*.

CONTROLLER * means that CP selects any eligible virtual machine that is configured as a VSWITCH controller. For more information, see usage note “3” on page 117.

If you specify multiple real devices by using the RDEV keyword or the GROUP keyword, then specify CONTROLLER *. The controller functions are then spread across multiple VSWITCH controllers, which provides more flexibility in case of a failure.

You can also provide a pool of specific controllers to be chosen from. Use the SET VSWITCH command or the MODIFY VSWITCH system configuration statement and specify a list of user IDs after the CONTROLLER keyword. However, this is not the recommended practice.

The CONTROLLER operand is optional.

Note: The CONTROLLER value is ignored when the VSWITCH manages EQDIO devices to connect the real hardware LAN segment. However, when a BRIDGEPORT HiperSockets connection is configured, a controller virtual machine is required to manage the HiperSockets interface.

LOG OFF|ON|Verbose

sets virtual switch activity-logging behavior for an active Network Express (EQDIO) interface.

The LOG setting affects the messages for an EQDIO device. The messages are displayed on the system operator console.

The LOG settings do not apply to an OSA-Express (QDIO) device because activity logging of a QDIO device is managed by a controller virtual machine.

LOG OFF

sets virtual switch activity-logging OFF. Normal activity is not logged to the system operator console. LOG OFF is the default VSwitch logging mode.

Note: Errors and unusual events are always logged to the system operator console regardless of the LOG setting.

LOG ON

sets virtual switch activity-logging to display brief messages that describe each significant event associated with EQDIO device operation. This setting is intended to be used by IBM personnel when diagnosing EQDIO interface problems.

LOG Verbose

sets virtual switch activity-logging to display more detailed messages that describe each significant event associated with EQDIO device operation. This setting is intended to be used by IBM personnel when diagnosing EQDIO interface problems.

IP

specifies IP transport for the virtual switch.

The IP transport type generally ignores the Ethernet network header (if provided) and uses the IP header to direct network packets on the LAN.

IP transport is compatible with only an OSA-Express (QDIO) uplink configuration. IP transport is not compatible with a Network Express (EQDIO) uplink configuration.

If neither IP nor ETHERNET are specified and TYPE QDIO is specified, then by default the transport is set to IP NONROUTER.

ETHernet
specifies Ethernet transport for the virtual switch.

The ETHERNET transport type uses Ethernet (Layer 2) network headers to direct network packets on the LAN.

ETHERNET transport is compatible with any VSWITCH uplink configuration.

For an IVL virtual switch the ETHERNET setting is required. An IVL virtual switch can function correctly only at the Ethernet (layer 2) level. The command fails if anything other than ETHERNET is specified for an IVL virtual switch.

If neither IP nor ETHERNET are specified and TYPE QDIO is specified, then by default the transport is set to IP NONROUTER.

NONrouter
indicates that the OSA-Express device identified by the RDEV keyword is not a router to the virtual switch. If a datagram is received at this device for an unknown IP address, the datagram is discarded. NONROUTER is valid for only an IP VSWITCH and is the default option.

PRIrouter
indicates that the OSA-Express device identified by the RDEV keyword acts as a primary router to the virtual switch. If a datagram is received at this device for an unknown IP address, the datagram is passed to the virtual switch. PRIROUTER is valid for only an IP VSWITCH.

GROup *groupname*

specifies that the virtual switch is associated with a group and configured for IEEE 802.3ad Link Aggregation. The *groupname* value is a single token (1-8 alphanumeric characters) that identifies the group. Use the SET PORT GROUP command to specify the attributes of the group and the OSA-Express (QDIO) or Network Express (EQDIO) devices that comprise the group. This option can be specified only when the virtual switch is defined with the ETHERNET transport attribute.

Before a GLOBAL virtual switch is associated with a group, the port group *groupname* must be defined by using the SET PORT GROUP command.

A LOCAL virtual switch can be associated with a group even if the group was not previously defined by using the SET PORT GROUP command. If necessary, the DEFINE VSWITCH command defines a group when the DEFINE VSWITCH command associates a group with a LOCAL virtual switch.

NOGroup
specifies that the virtual switch does not use link aggregation.

BRIDGEport
configures and specifies connectivity for a virtual switch Bridge Port. The Bridge Port is a special port that is used to connect the virtual switch to a HiperSockets CHPID, essentially bringing the HiperSockets CHPID to the virtual switch's simulated and physical networks.

Configuration of a virtual switch Bridge Port is supported only when all of the following conditions are met:

- The Bridge facility is supported by the processor.
- The virtual switch transport type is ETHERNET.
- The virtual switch type is QDIO.

Additionally, the ISOLATION adapter cannot be used when the virtual switch has a Bridge Port.

Only guest operating systems running in a virtual machine under z/VM and exploiting QDIO Enhanced Buffer State Management (QEBSM) are eligible to be bridged from the HiperSockets CHPID to the virtual switch's simulated and physical networks.

NONE
specifies that the virtual switch does not have a Bridge Port.

RDEV *nnnn*
is a real device number to be used as a Bridge Port to connect the virtual switch to a HiperSockets CHPID. Only devices that have been configured with either of the HiperSockets CHPID parameters EXTERNAL_BRIDGED can be specified. Use these parameters on the DEFINE CHPID command or the appropriate CHPARM in the IOCP (x4 for EXTERNAL_BRIDGED).

The device selected must be compatible to the type of virtual switch created; for example EXTERNAL_BRIDGED for a QDIO type virtual switch. Connectivity to the HiperSockets CHPID will be prevented when the specified device does not match the virtual switch's type.

You must specify a single real device number as a hexadecimal number between X'0001' and X'FFFD'. The real device number specified represents a trio of devices. For example, specifying BRIDGEPORT RDEV 508 means the devices, 508-50A, are used to provide the connection to the HiperSockets CHPID.

CONnect
indicates that the device identified by the RDEV keyword will be immediately activated, allowing the connection to be used as the active or standby Bridge Port.

DISCONnect
indicates that the device identified by the RDEV keyword will be placed in the inactive state. If this connection is the active Bridge Port connection, another virtual switch with a Bridge Port on the same CHPID in standby state will take over the active Bridge Port connection.

SECONDARY
indicates that this virtual switch should be assigned a role as a SECONDARY Bridge Port for the HiperSockets CHPID. This is the default. See usage note "16" on page 120 for more information about the SECONDARY Bridge Port role.

PRIMARY
indicates that this virtual switch must be assigned the role as the PRIMARY Bridge Port for the HiperSockets CHPID. See usage note "16" on page 120 for more information about the PRIMARY Bridge Port role.

IPTimeout *nnn*
specifies the length of time in minutes that a transient IP address table entry remains in the IP address table for the virtual switch. A transient entry is an entry added to the table by one guest on behalf of another. This operand has no meaning for ETHERNET networks.

nnn is a number from 1 to 240. 5 minutes is the default value.

VLAN
used to enable and configure IEEE standard 802.lQ VLAN support for the virtual switch being defined. A VLAN-aware virtual switch provides VLAN controls at the user level (with SET VSWITCH GRANT VLAN and PORTTYPE commands) that can not be overridden by a guest host. If this option is omitted for a type QDIO virtual switch, then the virtual switch is set to VLAN UNAWARE.

UNAWARE
indicates the virtual switch does not support IEEE standard 802.lQ. All frames flow within the virtual switch regardless of presence or absence of Virtual Local Area Network (VLAN) tags. Any VLAN tags present in the frames will be ignored within the switch (however the guest hosts may perform VLAN filtering at the virtual NIC level).

defvid
defines the virtual switch as a VLAN-aware switch supporting IEEE standard 802.lQ. The *defvid* defines the default VLAN ID to be assigned to guest ports when no VLAN ID is coded on the SET VSWITCH GRANT VLAN command, MODIFY VSWITCH GRANT VLAN statement, the SET VSWITCH PORTNUMBER command, the SET VSWITCH VLANID command, or through an ESM. It is a number

from 1 to 4094. A VLAN-aware virtual switch provides VLAN controls at the user level (with SET VSWITCH GRANT, SET VSWITCH PORTNUMBER and SET VSWITCH VLANID commands) that may not be overridden by a guest host.

AWARE
defines the virtual switch as a VLAN-aware switch supporting IEEE standard 802.lQ without a default VLAN ID. When a virtual switch is specified as VLAN AWARE, one or more VLAN IDs must be assigned to each guest port by either a SET VSWITCH GRANT VLAN command, MODIFY VSWITCH GRANT VLAN statement, the SET VSWITCH PORTNUMBER command, the SET VSWITCH VLANID command, or through an ESM. Failure to assign an explicit VLAN ID causes all untagged frames transmitted from the port to be discarded. In the case of a VLAN-unaware guest using a PORTTYPE ACCESS all outbound frames will be discarded until a VLAN ID is set for the port.

PORTType ACCESS
defines the default porttype attribute for guests authorized for the virtual switch. For PORTTYPE ACCESS, the guest is unaware of VLAN IDs and sends and receives only untagged traffic.

PORTType TRUNK
defines the default porttype attribute for guests authorized for the virtual switch. For PORTTYPE TRUNK, the guest is VLAN aware and sends and receives tagged traffic for those VLANs to which the guest is authorized. If the guest is also authorized to the *natvid*, untagged traffic sent or received by the guest is associated with the native VLAN ID (*natvid*) of the virtual switch.

PORTTYPE TRUNK is not allowed for a TYPE IVL virtual switch.

GVRP
indicates that the VLAN IDs in use on the virtual switch should be registered with GVRP-aware switches on the LAN. This provides dynamic VLAN registration and VLAN registration removal for networking switches. This eliminates the need to manually configure the individual port VLAN assignments.

GVRP is not supported for EQDIO devices.

NOGVRP
Do not register VLAN IDs with GVRP-aware switches on the LAN. When NOGVRP is specified VLAN port assignments must be configured manually.

NATive *natvid*
defines the native VLAN ID that is to be associated with untagged frames that are received and transmitted by the virtual switch. The *natvid* option is available only on a VLAN-aware switch. The value of *natvid* must be a number 1-4094. If the NATIVE operand is omitted, then the value of *natvid* defaults to 1 for a type QDIO virtual switch.

NATive NONE
causes all untagged frames to be discarded.

USERBASED
operational differences between USERBASED and PORTBASED VSwitches have been eliminated.

USERBASED specifies that user based rules will be applied when relocating a guest in an SSI. A user-defined port may or may not be preserved over a relocation. Port numbers assigned by z/VM will not be preserved. New port numbers will be assigned on the destination system. If portnumber predictability across an SSI is required, use the PORTBASED option on the VSwitch in each SSI member.

For more information, see *z/VM: Connectivity*.

USERBASED is not allowed for a TYPE IVL virtual switch.

PORTBASED
operational differences between USERBASED and PORTBASED VSwitches have been eliminated.

PORTBASED specifies that port based rules will be applied when relocating a guest in an SSI. User-defined port numbers will be preserved over a relocation. Port numbers assigned by z/VM will not be preserved. In this case a new port number will be allocated on the destination system.

For more information, see *z/VM: Connectivity*.

PORTname *portname*
must be 1-8 characters, and a maximum of three port names can be specified.

Notes:

1. The PORTNAME operand is deprecated and is not recommended for use.
2. If the PORTNAME operand is used, the following coordination requirement applies to OSA-Express adapters. When two or more virtual switches specify a PORTNAME value and use the same OSA-Express device, then the PORTNAME values must match. The requirement applies when the virtual switches are defined on the same LPAR or different LPARs on the same CPC.

   The coordination requirement does not apply to a virtual switch that does not specify a PORTNAME value.

   The coordination requirement does not apply to Network Express adapters.
3. The PORTNAME value cannot be changed while the virtual switch is connected to the hardware LAN segment. If the device is already started, issue the SET VSWITCH *switchname* DISCONNECT command.

Usage Notes

1. The DEFINE VSWITCH statement creates a virtual switch. The MODIFY VSWITCH statement can be used to modify a virtual switch by authorizing users to use the switch. Authorization to a virtual switch may also be provided by an External Security Manager.
2. Accounting is set for the switch based on the default accounting state as set by the SET VMLAN ACCOUNT SYSTEM command or configuration statement. If accounting is turned on after the virtual switch is defined, the virtual switch will need to be redefined for accounting to take effect.
3. A virtual switch OSA-Express (QDIO) connection to a real hardware LAN segment is not operational until an eligible controller virtual machine is selected to manage the OSA-Express interface. CP selects an eligible virtual machine to be the controller by using the following criteria:

   - If CONTROLLER *userid1* is specified on the DEFINE or SET VSWITCH commands or the DEFINE VSWITCH or MODIFY VSWITCH System Configuration statements, with either a single user ID or a list of user IDs, then only those user IDs are eligible to be selected.
   - If CONTROLLER * is specified or allowed to default, CP selects from any eligible virtual machine that is configured as a VSWITCH controller.

   Notes:

   a. It is recommended that the VSWITCH controller be at the same release level as CP, although all supported releases are allowed.
   b. A controller virtual machine is not required and is not used to manage a Network Express (EQDIO) device interface. An EQDIO device is controlled by VSwitch internal logic. If, however, a BRIDGEPORT HiperSockets connection is configured, a controller virtual machine is required to manage the HiperSockets device interface.

   A virtual machine becomes an eligible QDIO controller when all the following criteria are met:

   - The virtual machine is running a TCPIP MODULE at a release level that supports the function that is required for the virtual switch.
   - An IUCV *VSWITCH statement is included in the virtual machine's CP directory entry.
   - The TCP/IP VSWITCH CONTROLLER statement is coded, and has defaulted to be ON or is explicitly set to ON in the TCP/IP configuration file or through an OBEYFILE command.
   - The virtual machine has completed initialization.
   - The virtual machine has virtual device numbers available for CP to attach the control device.

     The virtual device range used by CP is specified in the VSWITCH CONTROLLER TCP/IP configuration statement. If no VDEV range is specified, CP uses the virtual device number (*vdev*) that matches the *rdev* number that is specified on the DEFINE VSWITCH or SET VSWITCH command. See the VSWITCH CONTROLLER Statement in *z/VM: TCP/IP Planning and Customization* for more information.

Note: Do not code DEVICE and LINK TCP/IP configuration statements for the device. Do not attach the real QDIO device to a controller virtual machine. These steps are handled automatically by VSWITCH processing when an eligible controller is selected.

If an eligible virtual machine is not found or none of the *rdevs* are operational, a message is issued and the virtual switch operates in a local LAN environment without an uplink device.

4. The virtual switch supports two modes of operation for data transport in support of both TCP/IP and non-IP based applications. In deciding which mode to deploy for your network, some things to consider about deploying an ETHERNET virtualized LAN segment are:

   - Do you intend to use EQDIO devices to connect to the real hardware LAN segment? If so, ETHERNET transport is required.
   - Do you intend to use a link aggregation group to connect to the real hardware LAN segment? If so, ETHERNET transport is required.
   - Do you intend to use a Hipersockets bridge port on the virtual switch? If so, ETHERNET transport is required.
   - Do your servers or applications require their own unique MAC addresses (load balancers)?
   - Do you plan to deploy non-IP based applications on your network (SNA or NetBIOS, for example)?
   - Do you want to build a virtual LAN segment that operates closely to its physical counterpart?

   The key attributes of each transport mode with their operational characteristics are as follows:

   - ETHERNET (Layer 2)
     - Supports all applications that deploy ETHERNET (IEEE 802).
     - ETHERNET frames are transported on the LAN segment.
     - All destinations are identified by MAC address.
     - MAC addresses are locally administered by the LAN administrator through z/VM CP commands or configuration statements.
     - Each host connection is identified by a single MAC address.
     - This is a Link Layer transport, in which all hosts maintain their respective ARP caches.
     - VLAN tagging resides within the ETHERNET frames per IEEE 802.1Q specifications.
     - When GROUP attribute is specified, the frames can be transported as part of the IEEE 802.3ad standard.
     - When a Bridge Port is defined the virtual switch provides physical LAN connectivity for the target HiperSocket channel.
   - IP
     - Supports IP for TCP/IP applications only.
     - IP packets are transported on the LAN segment.
     - All destinations are identified by IP addresses.
     - IP address assignments are set by the host running in the guest virtual machine.
     - Each host may have more than one IP address (multi-homed).
     - This is Link Layer independent (that is, no MAC addresses), and ARP processing is offloaded onto the OSA-Express adapter.
     - VLAN tagging resides in internal QDIO headers.
     - All hosts share the OSA-Express MAC address.

   A virtual switch operates in only one mode for a given instance. For example, if the switch is configured as IP, then all communications on the LAN segment must be IP based. The same is also true when the switch is configured in ETHERNET mode. These transport modes affect the method of data transfer for the virtual switch. The other operations of the switch, such as guest authorization, failover, controller configuration, and so on, function the same for both modes.

5. The IP transport type is IPv4 only. In order to support IPv6 through the virtual switch RDEV, the ETHERNET transport is required.
6. The MODIFY VSWITCH statement and the SET VSWITCH command cannot be used to change the type of transport. To change the type of transport, the virtual switch must be redefined.
7. Use the QUERY CONTROLLER command output to find the virtual machines that are the virtual switch controllers. Use the QUERY VSWITCH command to display information about the virtual switch.
8. CP manages the devices that are used to control a virtual switch's connection to a real LAN segment.

   For a Network Express (EQDIO) device, CP connects the real device to the VSwitch and directly controls the network device.

   For an OSA-Express (QDIO) device, CP attaches the devices to a controller virtual machine. CP also defines devices of type VSWITCH-OSD to the controller:

   - CP concatenates *switchname* with *vdev* and "DEV" to form the device name.
   - CP concatenates *switchname* with *vdev* and "LINK" to form the link names.

   For example, the VSWBETA connection using virtual device 300 would be configured with DEVICE VSWBETA0300DEV and LINK VSWBETA0300LINK. These names appear in the TCP/IP query and trace information that is obtained from the controller. Link status indicates whether the link is ready (active) or inactive (backup).

   DEVICE and LINK statements must not be included in the TCP/IP configuration file for these devices.

   The port number specified by *nnnn*.P*nn* is used by the controller when the device is initialized. If the port number is not specified, it will default to port 0. If the port number is not supported, initialization of the device is terminated.
9. Multiple real devices can be specified on the RDEV operand. This feature allows failover to an alternate real device in the event of a failure with the current OSA-Express or exclusive link aggregation group. All real devices specified must be active and connected to the same hardware LAN in order to effectively and dynamically failover to an alternate device. In addition, the alternate devices must be defined on separate CHPIDs.
10. PRIROUTER is required only when IP forwarding (routing) nodes will be coupled to the switch. Router nodes provide connectivity for their LAN segments (remote nodes) through their switch connection. When Router nodes are deployed, their switch connection must be configured as PRIROUTER. In addition to this, the switch itself must also be configured as PRIROUTER to the OSA-E adapter. This will insure delivery of datagrams destined for LAN segments that are connected through routers coupled to a switch. Only one connection on each OSA-Express card can be designated as PRIROUTER. If the switch is successful in establishing PRIROUTER on the OSA-Express card, no other node (or switch) sharing the same OSA-Express card will be able to act as PRIROUTER. If another connection has already been established as PRIROUTER, the switch will be left with NONROUTER status (which is reflected in the QUERY VSWITCH response).
11. NONROUTER is the default mode for the switch. Every node is directly coupled to the switch and the associated IP destinations are registered with the OSA-Express connection. This is the most efficient way to use the virtual switch. In this mode, packets with an unrecognized IP destination are automatically sent out through the switch connection.
12. For a VLAN aware virtual switch, you should specify the same native VLAN ID (*natvid*) as specified in your configuration of any physical switches in your network. Most hardware manufacturers use a default of 1.
13. The SET VSWITCH command cannot be used to change the VLAN awareness attribute. To change the attribute, detach the virtual switch and define it with the correct attribute by using the DEFINE VSWITCH command.

14. If the virtual switch is defined as VLAN UNAWARE, any attempts to define a VLAN membership will fail. When the CP access list is used, SET or MODIFY VSWITCH GRANT with VLAN membership list fails. In addition, SET or MODIFY VSWITCH VLANID fails and SET or MODIFY VSWITCH PORTNUMBER with VLAN members fails. When an ESM is used, the COUPLE command fails if a VLAN list is returned by the ESM.
15. The SET VSWITCH command cannot be used to change the USERBASED/PORTBASED attribute. The virtual switch will need to be redefined.
16. A virtual switch's Bridge Port can be configured to take on the role as either the PRIMARY or as a SECONDARY Bridge Port for the HiperSockets CHPID. SECONDARY is the default role assigned when defining a virtual switch with a Bridge Port. A single PRIMARY Bridge Port and up to four SECONDARY Bridge Ports can be defined per HiperSockets CHPID.

    A functional virtual switch configured for the PRIMARY role is always selected as the *active* Bridge Port connection. Functional virtual switches configured with the SECONDARY role provide backup (standby) capability in the event of failure on the part of the active Bridge Port connection.

    The first virtual switch Bridge Port successfully connecting to the HiperSockets CHPID (whether PRIMARY or SECONDARY) will take on the responsibility as the active Bridge Port connection. When the active Bridge Port connection is a virtual switch with a SECONDARY role, then its responsibility as the active Bridge Port connection will be relinquished to a connection made by a virtual switch requesting the PRIMARY role.
17. A Global virtual switch is a collection of virtual switches that share the same networking characteristics. This collection of virtual switches spans multiple systems running z/VM but logically operates as a single switch.

    Virtual switches defined with the same name and the GLOBAL option are said to be members of the same global virtual switch when they reside in systems running z/VM that belong to the same IVL domain. (Definition and activation of an IVL virtual switch allows a system running z/VM to join an IVL domain.

    The following conditions must be fulfilled in order to create or change a global virtual switch member:

    - The system's IVL virtual switch must be defined and its UPLINK port connected. In other words, the host must be an active member of an IVL domain.

      Creation of Global virtual switches specified in the SYSTEM CONFIG file will be deferred until the IVL virtual switch UPLINK port connects the system to the IVL domain. Message HCP3178I will be displayed for each deferred Global virtual switch.
    - If any other virtual switch exists in the IVL domain with the same name the following attributes must match or the DEFINE VSWITCH or SET VSWITCH will fail with message HCP3170E.
      - IP or ETHERNET
      - ISOLATION
      - VEPA
      - VLAN AWARE or UNAWARE
      - NATIVE *natvid*
      - USERBASED or PORTBASED
    - The following additional restrictions exist if a global virtual switch is configured to use a shared port group:
      - RDEV *device_addr* is not allowed.
18. A virtual switch defined to be TYPE IVL is an Inter-VSwitch Link which provides communication infrastructure to exchange control information and data necessary to manage global networking objects that can span multiple systems running z/VM.

    A single IVL virtual switch accommodates all inter-LPAR control traffic for a system. This virtual switch is defined by the system administrator using a CP command or a statement in the System Configuration file. The DEFINE VSWITCH specifies the RDEV or GROUP operand to configure OSA-

Express adapters to provide the required connectivity to systems running z/VM in other LPARs so they can all join the same IVL domain.

An IVL domain is a group of systems running z/VM that have each defined an IVL VSwitch with an UPLINK port configured and connected to the same external LAN segment. Configuration of an IVL virtual switch defines the system's IVL domain membership.

These domains allow isolation of global networking traffic through the enumeration of different domain letters A-H. Each is assigned a separate, reserved multicast MAC address. For example, communications for default Domain A is assigned to 03-FF-FF-FF-FF-01. (MAC address prefix 03-FF-FF is reserved for IVL communications for all systems running z/VM.)

The VLAN associated with a VLAN-aware IVL virtual switch can be modified using the IVLPORT VLAN operand on the SET VSWITCH command. Up to 8 IVL domains can be defined per VLAN.

Note that up to 16 systems can be members of the same IVL domain.

When a global virtual switch is in use the IVL virtual switch UPLINK port must remain operational in order to support full function for a global virtual switch that spans multiple systems.

For more information, see IVL (Inter-VSwitch Link) Overview in *z/VM: Connectivity*.

19. The QUERY VSWITCH command shows the configured QUEUESTORAGE range (*min-max*) as well as the current value. The value is static for an OSA-Express (QDIO) adapter, but the current value might vary for a Network Express (EQDIO) adapter. The current value might be outside the configured range in the following circumstances:
    - OSA-Express (QDIO) shows a current value of 8M if the configured range is greater than 8M. The OSA-Express (QDIO) interface can operate with a maximum of 8M of memory.
    - Network Express (EQDIO) might show a current value that is outside the configured range if queried soon after setting a new range.
20. The following special considerations apply to a CHPID that is defined with the LINK_AGGREGATION parameter for Network Express (OSH) devices.

    The LINK_AGGREGATION parameter on the DEFINE CHPID command, or the corresponding CHPARM in the IOCP (x02), indicates that the specified CHPID is to be used only for link aggregation. When the CHPID is defined in link aggregation mode, the device can be used only as part of a port group for z/VM virtual switch link aggregation. When the CHPID is not defined in link aggregation mode, attempts to bring up a z/VM virtual switch that is configured to use the device as part of a link aggregation port group will fail if either of the following conditions are true:
    - A NETH device is also configured on the port.
    - The system is managed by IBM Dynamic Partition Manager.

    It is recommended that all CHPIDs that are used as link-aggregation port group members are specifically designated for link aggregation mode in the IOCP. For migration purposes, z/VM supports a device that is part of a link aggregation port group on a CHPID that is not defined in link aggregation mode.

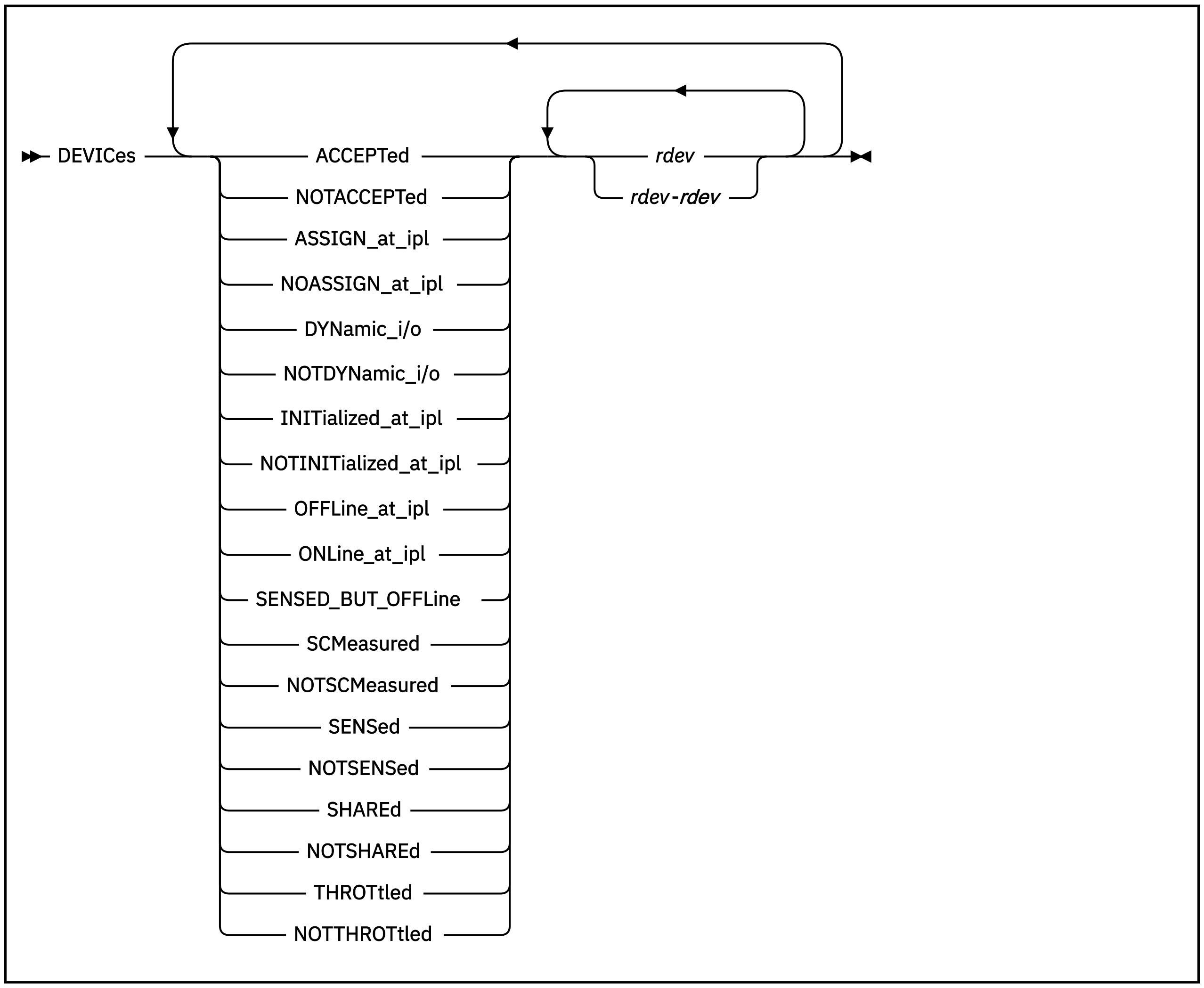
Examples

1. Define a switch named BIGANG that connects to a real LAN through device fd00:
```
  define vswitch bigang rdev fd00
```
2. Define a switch named ETH1 that uses Link Aggregation for a port group named ETH1GRP:
```
  define vswitch eth1 ethernet group eth1grp
```
3. Define a switch named LINPROD that uses OSA-Express hardware ports:
```
define vswitch linprod rdev 9c00.p0 9d00.p1 9e00
```
   In this example, device 9e00 connects to port 0 on the OSA-Express adapter.

4. Define a switch named ETH0 that connects to a real LAN through device 1e1b and to a HiperSockets CHPID through device 7000:

```
define vswitch eth0 type qdio rdev 1e1b ethernet bridgeport rdev 7000
```

DEVICES Statement



Purpose

Use the DEVICES statement to specify how CP handles specific devices during initialization. You can specify whether CP does the following actions:

- CP accepts (and builds real device blocks for) specified devices.
- CP allows dynamic changes for specified devices.
- CP initializes the specified devices during IPL.
- CP measures the subchannels for specified devices.
- CP assigns the tape drive to the system when the device is being brought online.
- CP uses the information returned from a sense ID request to help define the device.
- CP shares DASD between independent operating systems.
- CP limits (throttles) the I/O rate for specified devices.

How to Specify

Include as many statements as needed; they are optional. You can place DEVICES statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Parameters

ACCEPTed
CP accepts the specified device or devices.

rdev
rdev-rdev
is the real device number of the device or devices that you are affecting. The variable *rdev* must be a hexadecimal number between X'0000' and X'FFFF'. You can specify a single device, a range of devices, or any combination thereof.

NOTACCEPTed
CP does not accept the specified device or devices.

Note: CP ignores any devices you specify on the NOTACCEPTED operand, even if you have explicitly defined those devices by using RDEVICE statements in the system configuration file or by using RDEVICE macroinstructions in the HCPRIO ASSEMBLE file.

ASSIGN_at_ipl
CP assigns the tape drive to the system when the device is being brought online.

NOASSIGN_at_ipl
CP does not assign the tape drive when the device is being brought online.

DYNamic_i/o
CP allows dynamic I/O changes on the specified device or devices.

Note: If you want to allow dynamic I/O changes on ***any*** devices, do not forget to specify FEATURES ENABLE DYNAMIC_I/O in your system configuration file. This FEATURES operand enables or disables dynamic I/O changes for your system's processor. So, if you do not enable dynamic I/O changes on the processor, CP does not let you make dynamic I/O changes on any individual device or devices.

NOTDYNamic_i/o
CP does not allow dynamic I/O changes on the specified device or devices.

INITialized_at_ipl
CP initializes the specified device or devices during IPL. This operand is functionally equivalent to the ONLINE_AT_IPL operand.

NOTINITialized_at_ipl
CP does not initialize the specified device or devices during IPL. This operand is functionally equivalent to the OFFLINE_AT_IPL operand.

OFFLine_at_ipl
CP does not initialize the specified device or devices at IPL.

ONLine_at_ipl
CP initializes the specified device or devices at IPL.

SENSED_BUT_OFFLine
CP does not initialize the specified device or devices at IPL, but still issues a sense ID request to determine the device class or type.

SCMeasured
CP collects subchannel measurement data for the subchannels of the specified device or devices.

NOTSCMeasured
CP does not collect subchannel measurement data for the subchannels of the specified device or devices.

SENSed
CP uses the information returned from a sense ID request to determine the device class or type.

NOTSENSed
CP does not use the information returned from a sense ID request to determine the device class or type.

SHAREd
indicates that the specified DASD device or devices are shared between independent operating systems.

NOTSHAREd
indicates that the specified DASD device or devices are not be shared with any other independent operating system.

THROTtled
CP limits (throttles) the rate of I/O coming from the specified device or devices.

NOTTHROTtled
CP does not limit (throttle) the I/O rate of the specified device or devices.

Usage Notes

1. If an IODF statement is defined in the system configuration file with the *osconfig* parameter specified, then the software I/O configuration is controlled by HCD. In this case, the following DEVICES statements are ignored if they are specified:
   - DEVICES ACCEPTED
   - DEVICES NOTACCEPTED
   - DEVICES DYNAMIC_I/O
   - DEVICES NOTDYNAMIC_I/O
   - DEVICES OFFLINE_AT_IPL
   - DEVICES ONLINE_AT_IPL
   - DEVICES SENSED_BUT_OFFLINE
   - DEVICES SENSED
   - DEVICES NOTSENSED
   - DEVICES SHARED
   - DEVICES NOTSHARED

   For more information, see "IODF Statement" on page 181.
2. If you do not specify the DEVICES statement in the system configuration file, the following behavior results:
   - CP accepts all devices.
   - CP allows dynamic changes on all devices (if you specified FEATURES ENABLE DYNAMIC_I/O).
   - CP initializes all devices.
   - CP measures all devices.
   - CP senses the characteristics of all devices.
   - CP does not share any DASD devices.
   - CP does not throttle any devices.
   - CP does not assign any tape drives to CP at IPL.
   - For any devices that you did not explicitly define with the RDEVICE statement in the system configuration file, CP senses the device if possible and dynamically builds an RDEV for it.
   - If you specify the wrong device type on the RDEVICE statement, CP does not bring the device online because of conflicting device information. That is, if you define a 3490 tape drive and the device is really a 3590 tape drive, CP does not bring the device online. If you specify the wrong device class, CP does not bring the device online. That is, if you define a tape drive and the device is really a DASD, CP does not bring the device online.

   For more information, see "RDEVICE Statement" on page 227.
3. To allow or prevent dynamic changes after IPL, use the SET DYNAMIC_I/O command. For more information, see SET DYNAMIC_I/O in *z/VM: CP Commands and Utilities Reference*.

4. To start or stop subchannel measuring after IPL, use the SET SCMEASURE command. For more information, see SET SCMEASURE in *z/VM: CP Commands and Utilities Reference*.
5. You can also share DASD devices between independent operating systems by specifying the SHARED operand on the SET RDEVICE command, by issuing the SET SHARED command, or by specifying the SHARED operand on the RDEVICE statement. For more information, see "RDEVICE Statement" on page 227 and "RDEVICE Statement (DASD)" on page 236. See also SET RDEVICE and SET SHARED in *z/VM: CP Commands and Utilities Reference*.
6. To define new devices after IPL, use the SET RDEVICE command. For more information, see SET RDEVICE in *z/VM: CP Commands and Utilities Reference*.
7. For each category (ACCEPTED, DYNAMIC_I/O, INITIALIZED_AT_IPL, MEASURED, ONLINE_AT_IPL, SENSED, SHARED, THROTTLED, and ASSIGN_AT_IPL), CP processes the information on the DEVICES statement sequentially. If you specify more than one DEVICES statement or you overlap ranges of real device numbers, CP uses the last specification. For example, if you specify:

   ```
   DEVICES   NotAccepted  1000-1fff,
             Accepted     1800-18ff,
             NotAccepted  1820-182f
   ```

   CP accepts devices at address 1800 through 181F and at address 1830 through 18FF. CP does not accept devices at address 1000 through 17FF, 1820 through 182F, and 1900 through 1FFF.

   Note: The INITIALIZED_AT_IPL and ONLINE_AT_IPL operands perform exactly the same function. You can use these two operands (and their negative counterparts, NOTINITIALIZED_AT_IPL and OFFLINE_AT_IPL) in any combination. If you overlap ranges of real device numbers, CP uses the last specification.
8. Similar to the OFFLINE_AT_IPL operand, the SENSED_BUT_OFFLINE operand applies only during system initialization. Devices brought online by dynamic I/O after system initialization are not affected by the SENSED_BUT_OFFLINE setting.
9. If multiple subchannel set support is enabled, DEVICES settings defined by the SET DEVICES command or the DEVICES system configuration statement apply to all subchannel sets. For example, if device 1234 is NOTACCEPTED, then so is device 11234. If device 01234 is NOTACCEPTED and device 11234 is added dynamically, it is ignored.

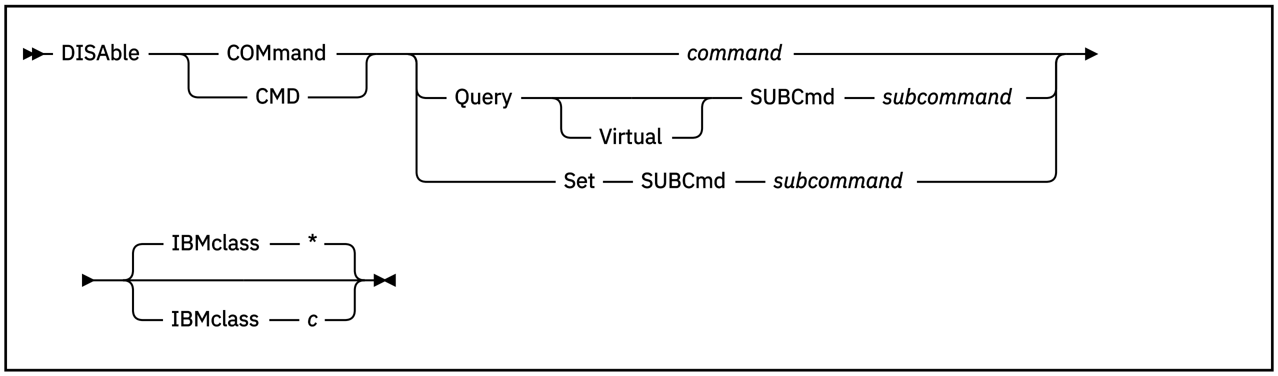
Examples

1. The following DEVICES statement specifies these settings:
   - CP does not accept a group of terminals, except for one.
   - CP allows dynamic I/O changes on all devices, except for 1905 through 1943.
   - CP initializes all devices, except 8E0 through 8EF.
   - CP measures all devices, except DASD 2F04.
   - CP leaves a group of disks offline or not initialized.
   - CP does not sense a group of tape devices.
   - CP shares a string of DASDs at 1100 through 114F.
   - CP throttles the I/O rate to a shared database at 460 through 48F.
   - CP assigns tape drive 181 to CP.

   ```
   Devices NotAccepted 2f12-2f22,           /* Don't accept terminals  */
           Accepted 2f1f,                   /* ... except David's      */
           Dynamic_I/O 0000-ffff,           /* Dynamic changes are ok  */
                                            /*    on these devices.    */
           NotDynamic_I/O 1905-1943,        /* No changes allowed on   */
                                            /*     these ... ever!     */
           Initialized_at_IPL 0000-ffff,    /* Initialize everything...*/
           NotInitialized_at_IPL 08e0-08ef, /* ... almost.             */
           SCMeasured 0000-ffff,            /* Measure everything ...  */
           NotSCMeasured 2f04,              /* ... almost.             */
   ```

```
           Offline_at_IPL 0a00-0aff,        /* Backup DASD string      */
           NotSensed 0100-01ff,             /* Don't sense these tapes */
           NotShared 0000-ffff,             /* Don't share anything!   */
           Shared 1100-114f,                /* ... except these.       */
           Throttled 0460-048f Rate 20      /* Limit the I/O on the    */
                                            /*    shared database DASD */
           Assign_at_ipl 181                /* Assign 181 to CP.       */
```

DISABLE COMMAND / CMD Statement



Purpose

Use the DISABLE COMMAND or CMD statement to prevent CP from processing requests for the specified CP command during and after initialization.

You can also prevent processing of CP commands after initialization by using the DISABLE COMMAND or CMD commands. For more information, see DISABLE COMMAND / CMD in *z/VM: CP Commands and Utilities Reference*

How to Specify

Include as many statements as needed; they are optional. You can place DISABLE COMMAND or CMD statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

command
: is the name of the command that you are disabling. The variable *command* is a 1-character to 12-character alphanumeric string.

Query SUBCmd *subcommand*
: tells CP the name of the CP QUERY subcommand that you are disabling. The variable *subcommand* is a 1-character to 12-character alphanumeric string.

Query Virtual SUBCmd *subcommand*
: tells CP the name of the CP QUERY VIRTUAL subcommand that you are disabling. The variable *subcommand* is a 1-character to 12-character alphanumeric string.

Set SUBCmd *subcommand*
: tells CP the name of the CP SET subcommand that you are disabling. The variable *subcommand* is a 1-character to 12-character alphanumeric string.

IBMclass *
: tells CP to disable all versions of the specified command or subcommand. If omitted, IBMCLASS * is the default.

IBMclass *c*
: tells CP to disable a specific version of the specified command or subcommand. The variable *c* can be any 1 of the following:

  A
  : this is a system-control command to be used by the primary system operator.

  B
  : this is a command for operational control of real devices.

C
: this is a command to alter host storage.

D
: this is a command for system-wide control of spool files.

E
: this is a command to examine host storage.

F
: this is a command for service control of real devices.

G
: this is a general-use command used to control the functions of a virtual machine.

0
: (zero) this command has no specific IBM class assigned.

Usage Notes

1. To remove the command processing code from the system execution space, use the CPXUNLOAD command. For more information, see CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.
2. To load the command processing code into the system execution space, use the CPXLOAD statement or command. For more information, see "CPXLOAD Statement" on page 76. See also CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
3. To deactivate a CP command while defining it, use the DISABLE operand of the DEFINE COMMAND or CMD statement or command. For more information, see "DEFINE COMMAND / CMD Statement" on page 90. See also DEFINE COMMAND / CMD command in *z/VM: CP Commands and Utilities Reference*.
4. To activate a CP command:
   - To activate a command while defining the command, use the ENABLE operand of the DEFINE COMMAND or CMD statement or command. For more information, see "DEFINE COMMAND / CMD Statement" on page 90. See also DEFINE COMMAND / CMD command in *z/VM: CP Commands and Utilities Reference*.
   - To activate a command after defining the command, use the ENABLE COMMAND or CMD statement or command. For more information, see "ENABLE COMMAND / CMD Statement" on page 146. See also ENABLE COMMAND / CMD in *z/VM: CP Commands and Utilities Reference*.
5. To change the definition of an existing CP command, use the MODIFY COMMAND or CMD statement or command. For more information, see "MODIFY COMMAND / CMD Statement" on page 188. See also MODIFY COMMAND / CMD in *z/VM: CP Commands and Utilities Reference*.
6. Once defined, COMMANDS, SUBCOMMANDS, and ALIASES cannot be deleted. They can be altered in various appropriate ways, but they remain in existence until a SHUTDOWN or RESTART IPL is done.
7. To display the address of the CP command table entry block, the current IBM class, and the current privilege class for a specified CP command, use the LOCATE CMDBK command. For more information, see LOCATE CMDBK in *z/VM: CP Commands and Utilities Reference*.
8. For more information about enabling and disabling commands, see Defining and Modifying Commands and Diagnose Codes in *z/VM: CP Exit Customization*.

Examples

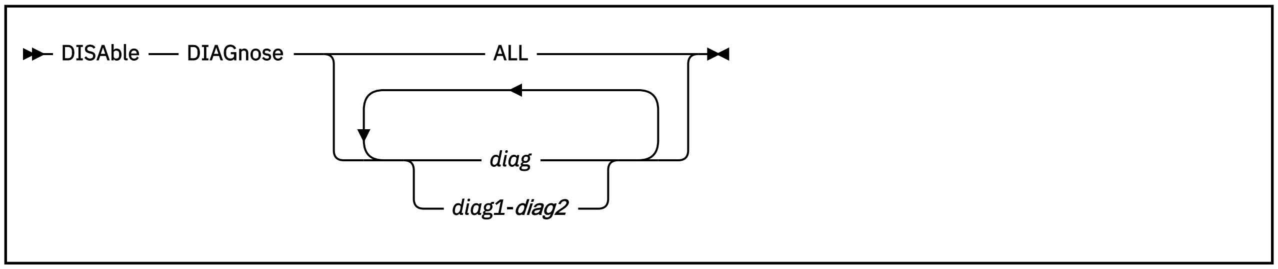
1. To disable the class D version of the CP PURGE command, but not the class G version of the CP PURGE command, use the following:

   ```
   Disable Command  purge  IBMclass d
   ```

2. To disable the CP SHUTDOWN command, use the following:

   ```
   Disable Command shutdown
   ```

DISABLE DIAGNOSE Statement



Purpose

Use the DISABLE DIAGNOSE statement to prevent CP from processing requests for one or more locally-developed DIAGNOSE codes during and after initialization.

You can also prevent processing of locally-developed DIAGNOSE codes after initialization using the DISABLE DIAGNOSE command. For more information, see DISABLE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place DISABLE DIAGNOSE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

ALL
tells CP to disable all existing DIAGNOSE codes.

Note: This operand disables all DIAGNOSE codes: the locally-defined ones, the IBM-supplied ones, and any supplied by third-party software vendors.

diag
diag1-diag2
is the number of the DIAGNOSE code that you are disabling. Each *diag* must be a hexadecimal number between X'0000' and X'03FC' and must be a multiple of 4. You can specify a single DIAGNOSE code, a range of DIAGNOSE codes, or any combination thereof.

Usage Notes

1. CP treats disabled DIAGNOSE codes as if they were never defined. If you try to use a disabled DIAGNOSE code in a program, CP will give you a program check specification exception.
2. To load the DIAGNOSE processing code into the system execution space, use the CPXLOAD statement or CPXLOAD CP command. For more information, see "CPXLOAD Statement" on page 76. See also CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
3. To define a new DIAGNOSE code, use the DEFINE DIAGNOSE statement or CP command. For more information, see "DEFINE DIAGNOSE Statement" on page 96. See also DEFINE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.

   Note: Unless you specify the ENABLE operand of the DEFINE DIAGNOSE statement or command, the new DIAGNOSE code is initially in a disabled state after being defined.
4. To activate a new DIAGNOSE code after defining it, use the ENABLE DIAGNOSE statement or ENABLE DIAGNOSE CP command. For more information, see "ENABLE DIAGNOSE Statement" on page 148. See also ENABLE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.

5. To change the definition of an existing DIAGNOSE code, use the MODIFY DIAGNOSE statement or command. For more information, see "MODIFY DIAGNOSE Statement" on page 192. See also MODIFY DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
6. To display information about a DIAGNOSE code (status, entry point name, and privilege class) after initialization, use the QUERY DIAGNOSE command. For more information, see QUERY DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
7. To display the address of the CP DIAGNOSE code table block for a DIAGNOSE code after initialization, use the LOCATE DGNBK command. For more information, see LOCATE DGNBK in *z/VM: CP Commands and Utilities Reference*.
8. Once defined, DIAGNOSE codes cannot be deleted. They can be altered in various appropriate ways, but they remain in existence until a SHUTDOWN or RESTART IPL is done.
9. To remove the DIAGNOSE processing code from the system execution space, use the CPXUNLOAD command. For more information, see CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.
10. Many external security managers (ESMs) do not support DIAGNOSE codes above X'03FC'. For this reason, CP does not support DIAGNOSE codes above X'03FC'. The DIAGNOSE codes between X'0000' and X'03FC' are divided as follows:

    X'0000' to X'00FC'
    Reserved for IBM use

    X'0100' to X'01FC'
    Reserved for customer use

    X'0200' to X'03FC'
    Reserved for IBM use.
11. For more information about user-defined DIAGNOSE codes, see Defining and Modifying Commands and Diagnose Codes in *z/VM: CP Exit Customization*.

Examples

1. To have CP disable DIAGNOSE code X'100', use the following:

   ```
   Disable Diagnose 100
   ```

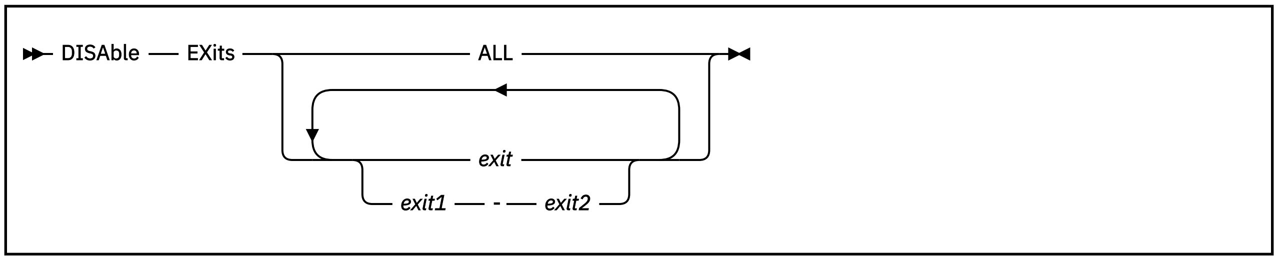
   After initialization, any virtual machine that issues DIAGNOSE code X'100' receives a program specification exception in that virtual machine.
2. To have CP disable all DIAGNOSE codes on your system, use the following:

   ```
   Disable Diagnose All
   ```
3. To have CP disable all locally-defined DIAGNOSE codes, use the following:

   ```
   Disable Diagnose 100-1fc
   ```
4. To have CP disable all locally-defined DIAGNOSE codes except DIAGNOSE code X'01F0', use the following:

   ```
   Disable Diagnose 100-1ec 1f4-1fc
   ```

DISABLE EXITS Statement



Purpose

Use the DISABLE EXITS statement to prevent CP from calling all entry points and external symbols associated with one or more exit points during and after initialization.

You can also prevent CP from calling one or more exit points after initialization by using the DISABLE EXITS command. For more information, see DISABLE EXITS in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place DISABLE EXITS statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

ALL
tells CP to disable all existing CP exit points.

exit
exit1-exit2
is the number of the exit point (or exit points) that you do not want CP to use. Each *exit* must be a hexadecimal number between X'0000' and X'FFFF'. You can specify a single exit point number, a range of exit point numbers, or any combination thereof.

Usage Notes

1. To load the exit point code into the system execution space, use the CPXLOAD statement or command. For more information, see "CPXLOAD Statement" on page 76. See also CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
2. To add to, change, or replace the list of entry points and external symbols associated with an exit point and to enable or disable that exit point, use the ASSOCIATE EXIT statement or command. For more information, see "ASSOCIATE EXIT Statement" on page 58. See also ASSOCIATE EXIT in *z/VM: CP Commands and Utilities Reference*.
3. To activate an exit point after defining it, use the ENABLE EXITS statement or command. For more information, see "ENABLE EXITS Statement" on page 150. See also ENABLE EXITS in *z/VM: CP Commands and Utilities Reference*.
4. To display status and usage statistics information about a specific exit point after initialization, use the QUERY EXITS command. For more information see QUERY EXITS in *z/VM: CP Commands and Utilities Reference*.
5. To display the address of the CP exit block for a specific exit point, use the LOCATE XITBK command. For more information see LOCATE XITBK in *z/VM: CP Commands and Utilities Reference*.

6. To display the address of the CP indirect call locator block for a specific exit point, use the LOCATE ICLBK command. For more information see LOCATE ICLBK in *z/VM: CP Commands and Utilities Reference*.
7. To change the definition of an existing dynamic exit point, or remove the exit point from the system, use the MODIFY EXIT statement or command. For more information, see "MODIFY EXIT Statement" on page 195. See also MODIFY EXIT in *z/VM: CP Commands and Utilities Reference*.
8. To remove the exit point code from the system execution space, use the CPXUNLOAD command. For more information see CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.
9. After processing a DISABLE EXITS statement (or command), CP updates the status of the exit point in its CP exit block, but does not erase the CP exit block. CP will not erase any CP exit blocks until the next IPL.
10. For more information about user-defined exit points, see Benefits of using CP exits in *z/VM: CP Exit Customization*.

Examples

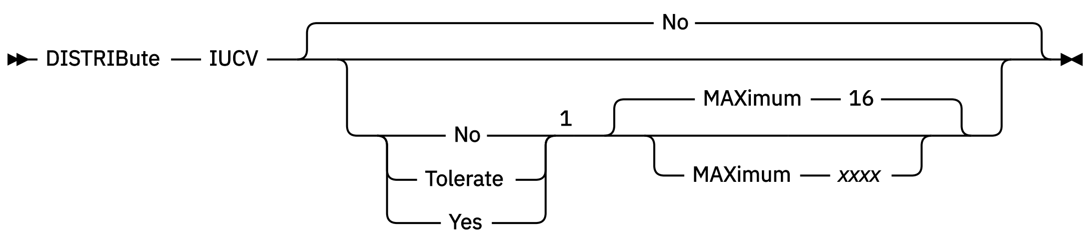
1. To stop CP from calling the entry points and external symbols associated with CP Exits 1, 2, 3, 4, and 6, use the following:

   ```
   Disable Exits 1-4 6
   ```

2. To stop CP from calling the entry points and external symbols associated with all CP exit points, use the following:

   ```
   Disable Exits All
   ```

DISTRIBUTE Statement



Notes:

[1] MAXIMUM specified with NO is meaningful only within an SSI cluster.

Purpose

Use the DISTRIBUTE statement to specify distribution features for the local system.

How to Specify

Include as many statements as needed; they are optional. You can place DISTRIBUTE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

IUCV
indicates that IUCV distribution option is to be set.

No
indicates that no Distributed IUCV traffic will be allowed to or from other nodes within the ISFC collection. This is the default.

Tolerate
indicates that Distributed IUCV is allowed to or from any other node in the ISFC collection. However, IUCV traffic will only leave the local node if the application specifies the target system name.

Yes
indicates that this node participates fully in Distributed IUCV across the entire ISFC collection. This means that target searches will be done to the ISFC collection instead of the default of just the local system.

MAXimum *xxxx*
indicates the number of megabytes of the largest distributed IUCV send allowed from the host. *xxxx* is a value between 1 and 1024. The default maximum is 16.

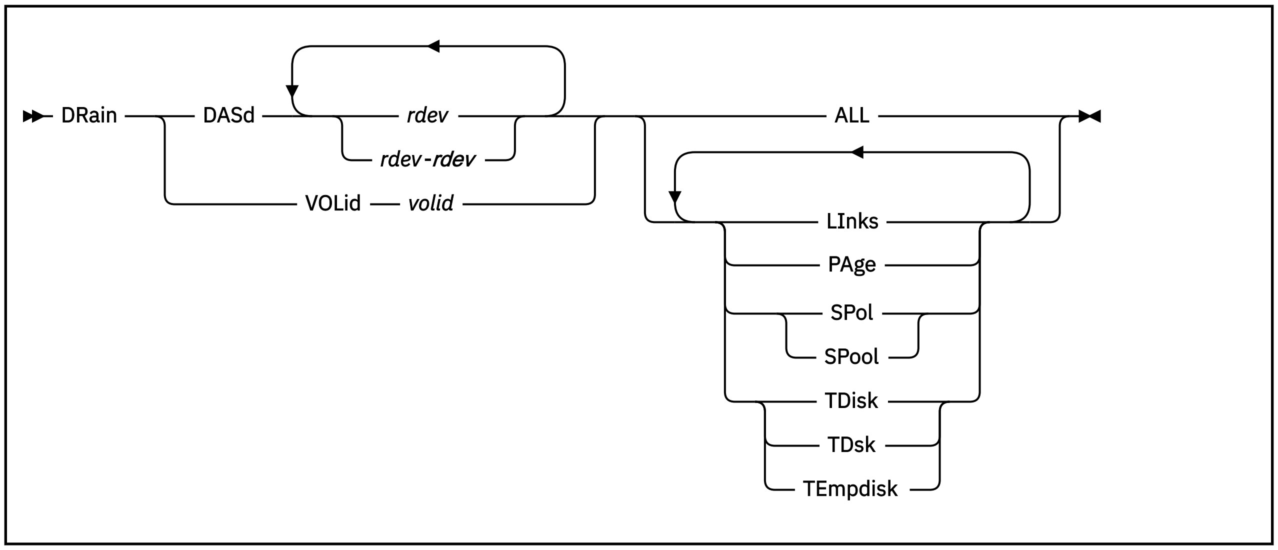
Note: It is recommended that all nodes in a CS collection have the same value specified for MAXIMUM because the maximum value is enforced only by the originating node.

Usage Notes

1. When TOLERATE is specified, IUCV will be distributed via the TARGET parameter of the IUCV macro. When YES is specified, IUCV will first attempt to satisfy a CONNECT on the local system and then will attempt to locate the target on a system within the CS collection. The only exception is if the application specifies that the CONNECT must be satisfied either locally or on a particular target system.

2. In the configuration file for an SSI cluster, the DISTRIBUTE IUCV statement governs the use of this feature beyond the cluster (for example, if the cluster is part of a larger ISFC collection). NO is the default value. However, regardless of the setting, cross-system IUCV is automatically available among the members of the cluster, and a MAXIMUM value can be specified with NO to define the IUCV send limit within the cluster.
3. IUCV applications will behave the same in a distributed environment as they do on a local system with the following exceptions:
   - PURGE and REJECT will only be honored on the local system. Once a message is sent to the other system it is considered to be delivered.
   - The PRIORITY and MSGLIMIT directory specifications must be present on both systems if they are to be honored.
   - The default maximum data length is 16 MB per message. The maximum can be altered via the SYSTEM CONFIG file by using the DISTRIBUTE statement.

DRAIN (Disk) Statement



Purpose

Use the DRAIN statement to stop new operations on specified DASD. You can stop CP from:

- Writing pages to the specified device or devices
- Letting users link to minidisks on the specified device or devices
- Allocating spool space on the specified device or devices
- Allocating temporary minidisks on the specified device or devices.

How to Specify

Include as many statements as needed; they are optional. You can place DRAIN DASD statements anywhere in the system configuration file, but you must place DRAIN VOLID statements after the CP_OWNED statement (see "CP_OWNED Statement" on page 73) with the same volume serial number.

If you specify the same device on more than one statement, CP keeps a cumulative list of operations. For example, if you have one statement indicating that you do not want paging on a device and another statement indicating that you do not want temporary disks allocated on that same device, CP prevents paging **and** temporary disk allocation on that device. The second statement does not overrule the first statement.

Operands

DASd *rdev*
is the real device number of the DASD you want drained. The variable *rdev* is a hexadecimal number between X'0000' and X'FFFF'.

DASd *rdev-rdev*
is a range of real device numbers specifying the DASD you want drained. Each *rdev* is a hexadecimal number between X'0000' and X'FFFF'.

VOLid *volid*
is the volume serial number of the volume you want drained.

ALL
tells CP not to allow any new operations on this device, including:

- Writing pages during page-out
- Allowing minidisk linking

- Allocating space for new spool records.
- Allocating temporary disk space.

LInks
tells CP not to allow users to link to minidisks on this device.

PAge
tells CP not to write pages to this device during page-out operations.

SPol
SPool
tells CP not to allocate space on this device for new spool records.

TDisk
TDsk
TEmpdisk
tells CP not to allocate temporary disk space on this device.

Usage Notes

1. When you drain a DASD (or one of its operations), CP makes the DASD appear to be full. Because the DASD appears to be full, CP tries to allocate from the next appropriate device, if possible. Issuing DRAIN (Disk) or START (Disk) commands does not cause CP to adjust the system counters. For example, if you are draining spool space on a DASD, CP does not adjust the amount of spool space that is unused and available to zero, even though, logically, there appears to be no available spool space on that DASD. Issuing a CP DETACH command adjusts the appropriate system counters to account for the space being removed from the system. For more information, see DRAIN (Disk), START (Disk), and DETACH in *z/VM: CP Commands and Utilities Reference*.
2. You can attach a draining DASD to a virtual machine.
3. Issuing CP ATTACH and DETACH commands does not affect the DRAINING status indicators, so you can drain a DASD before attaching it to the system. For more information, see ATTACH and DETACH in *z/VM: CP Commands and Utilities Reference*.
4. CP processes the system configuration file during an IPL, which means CP has not attached any volumes to the system when processing your DRAIN statements. Therefore, when you specify a DRAIN VOLID statement, you can only use volumes that were previously specified in the system configuration file on CP_OWNED statements.
5. Use the CP START (Disk) command or statement to tell CP to resume normal allocation on the device and use the QUERY DASD DRAINING command to get information about the draining status of your DASD. For more information, see "START (Disk) Statement" on page 276. See also START (Disk) and QUERY DASD in *z/VM: CP Commands and Utilities Reference*.

Examples

Use the DRAIN and START statements shown in Figure 4 on page 138 to have CP:

- Drain all operations on all DASD between X'0700' and X'07FF',
- Allow users to link to minidisks on DASD X'0700', and
- Ensure that CP can write pages to the CP-owned paging pack (SYSPG1), if someone moves that DASD to one of the addresses that are draining (X'0700' through X'07FF')

```
Drain  DASD   0700-07ff  All        /* Do not allow any activity on
                                       these DASD ...              */

Start  DASD   0700       Links      /* ... except for 700, which has
                                       a minidisk I need to link to */

Start  Volid  SYSPG1     Page       /* ... and, if Operations moves
                                       the CP-owned paging pack any-
                                       where in the 700-7FF range, it
                                       has to be useable for system
                                       paging!                     */
```

Figure 4. Example DRAIN and START Statements

After you IPL the system, you can use the CP QUERY DASD DRAINING command to see the results of your DRAIN (Disk) and START (Disk) statements. For example, if the CP-owned volume did not fall into the range of drained devices, you would see:

```
Vol-ID Rdev   Draining: PAge  LInks SPool TDisk
SYS700 0700             Yes   No    Yes   Yes
SYS701 0701             Yes   Yes   Yes   Yes
SYS702 0702             Yes   Yes   Yes   Yes
 .
 .
 .
SYS7FE 07FE             Yes   Yes   Yes   Yes
SYS7FF 07FF             Yes   Yes   Yes   Yes
```

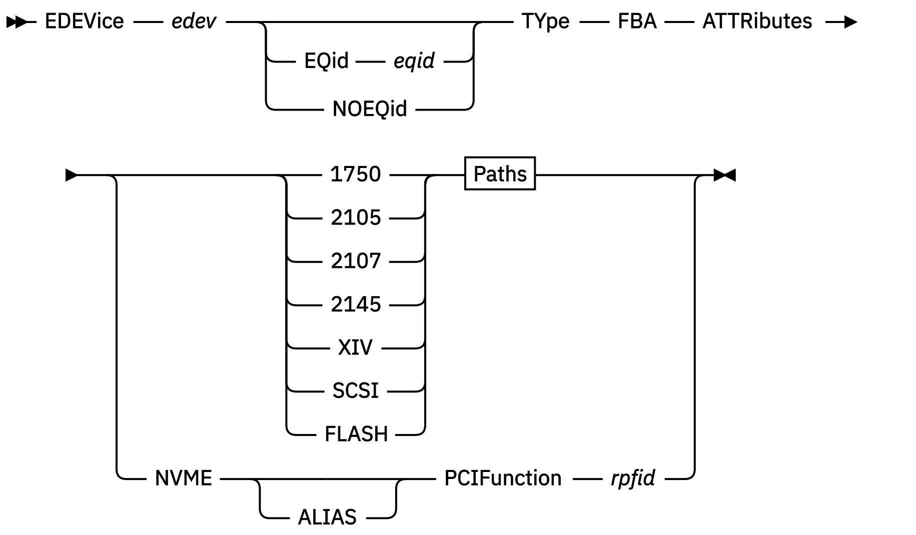
If the CP-owned volume SYSPG1 fell into the range of drained devices, you would see:

```
Vol-ID Rdev   Draining: PAge  LInks SPool TDisk
 .
 .
 .
SYSPG1 0777             No    Yes   Yes   Yes
 .
 .
 .
```

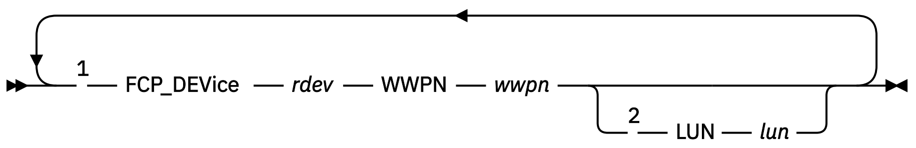
If the CP-owned volume SYSPG1 had a real device number of X'0700', you would see:

```
Vol-ID Rdev   Draining: PAge  LInks SPool TDisk
 .
 .
 .
SYSPG1 0700             No    No    Yes   Yes
 .
 .
 .
```

EDEVICE Statement



Paths



Notes:

[1] You can specify a maximum of 8 "paths" to the device.

[2] The LUN operand is required when you define the first path to a SCSI EDEVICE. For subsequent paths, the LUN operand is optional. If you specify the LUN operand, the value must match the LUN on the first path.

Purpose

Use the EDEVICE statement to define an emulated device that represents a real SCSI device or a real NVMe device that is connected to a PCIe adapter.

How to Specify

Include as many EDEVICE statements as needed; they are optional and can be placed anywhere in the system configuration file. If you specify more than one statement with the real device number, CP uses the last statement. For example, if you specify:

```
RDEVice 0500 Type AFP
 .
 .
 .
EDEVice 0500 TYpe FBA ATTR 2105 FCP_DEV 2000 WWPN 5005076300CE04DA,
                  LUN 6100000000000000
```

CP defines an emulated FBA DASD at real device number 0500, and not an advanced function printer.

Operands

edev
: specifies the device number that is associated with the emulated device. The *edev* number corresponds directly with an *rdev* number. Thus, a new number cannot be defined where an *rdev* number already exists. An *edev* in the active configuration can be entered as a 4-digit hexadecimal device number between X'0000' and X'FFFF'. For an *edev* in the active or standby configuration, the device number can be a 5-digit hexadecimal number between X'00000' and X'3FFFF' with the leading digit specifying the subchannel set of the device for which you want information to be displayed.

EQid *eqid*
: assigns the device equivalency ID (EQID) *eqid* to the EDEV. The *eqid* is a string of 1–8 alphanumeric characters. Note that when VARYing this device online, a system-generated EQID will override this equivalency ID if one can be generated.

NOEQid
: removes a previously assigned EQID from this RDEV. After the device is varied online, it reverts back to a system-generated EQID.

TYpe
: specifies the emulated device type, selected by the immediately following parameter.

FBA
: identifies the emulated device as an FBA DASD.

ATTRibutes
: specifies the name of an attribute set to be associated with the real device.

1750
: specifies that IBM 1750 device attributes are to be used for the real device that is being emulated. See Usage Note "4" on page 142 for more information.

2105
: specifies that IBM 2105 device attributes are to be used for the real device that is being emulated.

2107
: specifies that IBM 2107 device attributes are to be used for the real device that is being emulated.

2145
: specifies that IBM 2145 device attributes are to be used for the real device that is being emulated. See Usage Note "5" on page 142 for more information.

XIV®
: specifies that IBM XIV device attributes are to be used for the real device that is being emulated.

SCSI
: specifies that general SCSI device attributes are to be used for the real device that is being emulated.

  Note: When specifying this value, you should exercise caution in defining more than one path to the device. Be sure that the device actually supports multiple paths. Defining multiple paths to a device that does not support multiple paths could result in data-integrity problems on the device.

FLASH
: specifies that IBM FLASH device attributes are to be used for the real device that is being emulated. This attribute is used for IBM FlashSystem hardware and does not require a SAN Volume Controller (2145) in use.

NVME
: specifies that NVMe device attributes are to be used for the device that is emulated. This attribute is used for NVMe devices that are connected via PCIe functions.

FCP_DEVice *rdev*
: specifies the real device number of the FCP device to be used for a specific path to a SCSI device. The *rdev* must be a 1 - 4 digit hexadecimal number between X'0000' and X'FFFF'.

WWPN *wwpn*
: specifies the world wide port name for a specific path to a SCSI device. The *wwpn* must be a 16 digit hexadecimal number between X'0000000000000000' and X'FFFFFFFFFFFFFFFF'.

If fewer than 16 digits are specified for the *wwpn*, the number will be padded with leading zeros to make it a 16 digit number.

LUN *lun*
specifies the logical unit number for a specific path to a SCSI device. The *lun* must be a 16 digit hexadecimal number between X'0000000000000000' and X'FFFFFFFFFFFFFFFF'. Note, however, that the number of digits recognized by the SCSI device may vary by manufacturer, type and model.

If fewer than 16 digits are specified for the *lun*, the number will be padded with leading zeros to make it a 16 digit number, but this will almost certainly produce an invalid value. You must be careful to specify all 16 digits, using *trailing* zeros, as per the note below.

Attention
Because CP cannot know how many digits your device recognizes, you must enter *all 16 digits*. If your SCSI device is identified by fewer than 16 digits, you must use *trailing* zeros to fill out the 16 digit *lun*. Note that if you enter nonzero digits following the digits identifying the SCSI device, those extra digits might be ignored by your device. It is therefore possible to define multiple EDEVs that differ only in their rightmost unsupported digits, and the EDEVs will in fact represent the same SCSI device. (For example, if your device recognizes a 4 digit *lun*, 5A51888811221122 and 5A51999933443344 would both represent the same device.) This could lead to unintended device sharing and hence might introduce data-integrity exposures. **Always identify your SCSI device by its actual logical unit number as defined in your storage-area network, padded on the right with zeros.**

ALIAS
specifies that an alias device is to be defined.

PCIFunction *rpfid*
is the ID of a real PCI function that is associated with an NVMe adapter. Specify the ID as eight hexadecimal digits. You can omit leading zeros. If an EDEVICE is already associated with the PCI function, another one is defined. One base device can be defined for roughly each one terabyte of device capacity. A combination of up to 127 base and alias devices can be defined for a single NVME PCIe function.

Usage Notes

1. When defining emulated devices to represent real SCSI devices, there should be a one-to-one relationship between an emulated device and a real SCSI device. All paths defined for an emulated device should represent paths to the same real SCSI device. If the paths for one emulated device are associated with more than one real SCSI device, or if more than one emulated device is associated with the same real SCSI device, then data integrity problems could occur.
2. Path validation for an emulated device occurs when the emulated device is varied online. Any invalid path will be deleted from the EDEV. A path is considered to be invalid for any of the following reasons:
   - The device specified by the FCP_DEVice parameter does not exist.
   - The device specified by the FCP_DEVice parameter is not an FCP device.
   - The device specified by the FCP_DEVice parameter is an offline FCP device.
   - The device specified by the FCP_DEVice parameter is dedicated to a virtual machine.
   - The value specified by the WWPN parameter is not a valid world wide port name in your configuration.
   - The value specified by the LUN parameter is not a valid logical unit number for the specified world wide port name.
3. You can add paths and change the attribute name associated with a SCSI system residence volume by specifying an EDEVICE statement for it. The EDEVICE statement must specify the path already associated with the device (passed from SAPL) in order to be valid. By default a SCSI system residence volume is associated with the general SCSI attributes.

4. Devices (LUNs) on an IBM 1750 (DS6000) storage controller can be configured through any of the I/O ports on the controller, but the access times for a given device through different ports can be different. For a given device, half of the ports will provide faster access times than the other half of the ports. The channel paths connected to the ports with faster access are called "preferred paths" for the device, and the remaining channel paths are called "non-preferred paths".

   Previous releases of z/VM required knowledge of the preferred/non-preferred value for each path on a SET EDEV command when using the 1750 attribute. Now, z/VM is able to determine during device initialization which paths are preferred and which are not, so operands on the SET EDEV command that explicitly state the preferred/non-preferred value for such a device will be ignored. Preferred paths will still be given preference over non-preferred paths during the path selection for an I/O request.
5. When using an IBM 2145, a host that has a LUN mapping for a virtual disk will normally be able to access that virtual disk via any of the ports connected to either of the two nodes in the I/O group supporting that virtual disk. Each virtual disk has a set of preferred ports. These ports will give slightly better performance than the ports that are non-preferred. By default, the cache component assigns ownership of even numbered virtual disks to one node of a caching pair and the ownership of odd numbered virtual disks to the other node. To provide flexibility, the ownership for a given virtual disk can be explicitly assigned to a given node when the virtual disk is created. A node that is explicitly assigned as an owner of a virtual disk is known as the preferred node.

   Previous releases of z/VM required knowledge of the preferred/non-preferred value for each path on a SET EDEV command when using the 2145 attribute. Now, z/VM is able to determine during device initialization which paths are preferred and which are not, so operands on the SET EDEV command that explicitly state the preferred/non-preferred value for such a device will be ignored. Preferred paths will still be given preference over non-preferred paths during the path selection for an I/O request.
6. If an EQID is specified along with the CLEAR parameter, the EQID is ignored and the EDEV is cleared without setting an EQID.
7. When setting an EQID for an EDEV, no other device can have the same EQID assigned to it. Furthermore, when the device is brought online, if a system-generated EQID can be assigned to the device, the user-defined EQID will be removed and the system-generated ID will be used.
8. If the system cannot generate a unique EQID for this device, CP will issue the following message:

   HCP048E
   A unique EQID cannot be generated for EDEV *edev* based upon the hardware information provided.
9. If the specified EQID has already been assigned to one or more other devices, CP will issue the following message:

   HCP048E
   Specified EQID already assigned to a different device.
10. For system attached paging EDEVICEs, z/VM has an optimized path for issuing SCSI I/O requests directly without any simulation overhead. For non-paging EDEVICEs, there is simulation overhead incurred with every I/O request. Therefore, it is recommended that EDEVICEs only be used for disks which are accessed infrequently, such as a system disk used to store z/VM kernel images. z/VM has industry leading virtualization support for FCP subchannel attachment, which should be considered the primary approach for configuring any I/O intensive environments.
11. When configuring a single LUN on an XIV, SVC (2145), or a device incorporating SVC technology (such as V7000, V840, and V9000) as an EDEVICE, each associated FCP (source) subchannel should be configured to only one target port (WWPN) of the storage system. In addition, it is recommended that each (source) FCP subchannel be defined on a different (source) FCP channel path (or CHPID). This provides for best LUN availability in the event of a link failure.
12. When you vary on the device, the device attribute is checked dynamically. If the wrong value was set, the ATTR value will be updated to the correct device attribute. Any further paths added or deleted using the SET EDEVICE command will require specifying the correct ATTR operand. When setting the

ATTR operand for non-IBM devices, the above behavior is ignored, and the specifed ATTR value will be used as the specified device attribute for the emulated device.

Examples

1. To define an emulated device 8181 for a 2105 SCSI device with one path, enter the following:

```
edev 8181 type fba attr 2105,
               fcp_dev 900 wwpn 200400A0B80BA987 lun 0002000000000000
```

2. To define an emulated device 4343 for a general SCSI device with two paths, enter the following:

```
edev 4343 type fba attr scsi,
               fcp_dev 800 wwpn 200400A0B80BA687 lun 0001000000000000,
               fcp_dev 801 wwpn 200400A0B80BA688 lun 0001000000000000
```

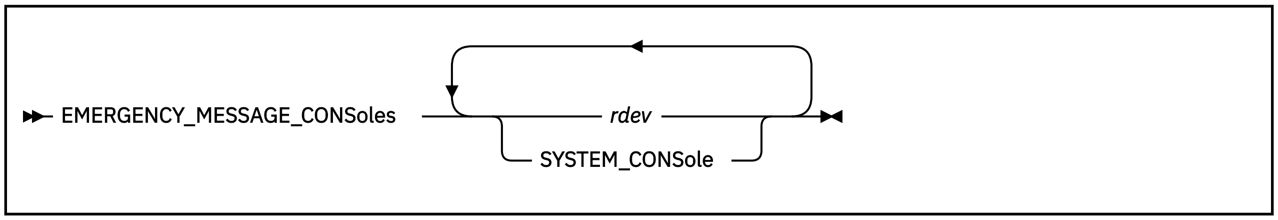
3. To define an emulated device 5004 for a 2145 device with two paths where the LUN is optional on the second path, enter the following::

```
edev 5004 type fba attr 2145
               fcp_dev 800 wwpn 200400A0B80BA687 lun 0002000000000000
               fcp_dev 801 wwpn 200400A0B80BA688
```

4. Define emulated base devices (0F80, 0F81) and emulated alias devices (0F90, 0F91, 0F92) for a real PCIe function 00000071 that is associated with an NVMe adapter:

```
edev f80 type fba attr nvme pcif 00000071
edev f81 type fba attr nvme pcif 00000071
edev f90 type fba attr nvme alias pcif 00000071
edev f91 type fba attr nvme alias pcif 00000071
edev f92 type fba attr nvme alias pcif 00000071
```

EMERGENCY_MESSAGE_CONSOLES Statement



Purpose

Use the EMERGENCY_MESSAGE_CONSOLES statement to define the list of console addresses which CP notifies when there is a system emergency (for example, an impending abend, shutdown, or dump). During initialization, CP creates a list of valid emergency consoles from the ones you have defined on this statement. The maximum number of unique console IDs that you can specify is 100. If CP abends, or if a software re-IPL occurs, CP sends messages to all the consoles on the list it created.

How to Specify

This statement is optional and you can place it anywhere in the system configuration file. Although you can include as many statements as needed, we recommend you only specify one. This statement defines the entire list of consoles that CP uses to notify you of impending abends or other system emergencies. If you specify more than one statement, CP redefines the list each time and only uses the list of consoles specified on the very last EMERGENCY_MESSAGE_CONSOLES statement that you specify. CP will not combine multiple statements into one comprehensive list of consoles.

Operands

rdev
adds the real device number or numbers to the list of CP emergency consoles that are sent emergency messages. The variable *rdev* must be a hexadecimal number between X'0000' and X'FFFF', and they must be locally-attached 3270-type supported displays. (For a complete list of supported 3270-type displays, see *z/VM: General Information*.)

SYSTEM_CONSole
adds the system console to the list of consoles that are sent emergency messages. For more information about how to view the messages that are produced on the system console, see *z/VM: System Operation* .

Usage Notes

1. If an IODF statement is defined in the system configuration file with the *osconfig* parameter specified, then the software I/O configuration will be controlled by HCD. In this case, the EMERGENCY_MESSAGE_CONSoles statement is ignored if it is specified. For more information, see "IODF Statement" on page 181.
2. SYSTEM 3270, 2250 and 3250 displays are not valid emergency message consoles.
3. If you do not specify an EMERGENCY_MESSAGE_CONSOLES statement, CP sends all emergency messages to the operator consoles that you listed on the OPERATOR_CONSOLES statement in the system configuration file before IPL. During IPL, CP checks all the consoles in this list to see if they are operational. If they are, CP includes them in the list of operator consoles. For more information, see "OPERATOR_CONSOLES Statement" on page 217.
4. If the CONS=*addr* or CON=*addr* parameter was specified on the Stand-Alone Program Loader (SAPL) panel to specify a console address and this console address is not already specified on the EMERGENCY_MESSAGE_CONSOLES statement, it is added to the list of emergency message consoles.

There will be a total of 101 emergency message consoles if the maximum number of 100 is specified on the EMERGENCY_MESSAGE_CONSOLES statement and the additional console address is added to the list.

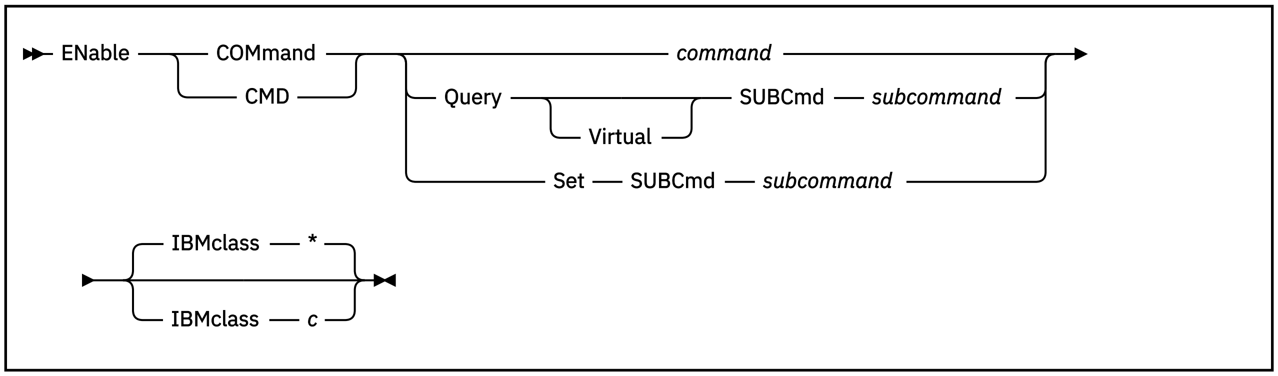
5. The system console is automatically added to the list of emergency message consoles if it is not already in the list. For more information about how to view the messages that are produced on the system console, see *z/VM: System Operation*.

Examples

1. To have CP send emergency messages to the operator's and system programmer's consoles, use the following EMERGENCY_MESSAGE_CONSOLES statement:

```
Emergency_Message_Consoles 0bc0 0bc1   /* Oper and SysProg consoles */
```

ENABLE COMMAND / CMD Statement



Purpose

Use the ENABLE COMMAND or CMD statement to permit CP to process requests for the specified CP command during and after initialization.

You can also permit processing of CP commands after initialization using the ENABLE COMMAND or CMD commands. For more information, see ENABLE COMMAND / CMD in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place ENABLE COMMAND or CMD statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

command
is the name of the command that you are enabling. The variable *command* is a 1-character to 12-character alphanumeric string.

Query SUBCmd *subcommand*
tells CP the name of the CP QUERY subcommand that you are enabling. The variable *subcommand* is a 1-character to 12-character alphanumeric string.

Query Virtual SUBCmd *subcommand*
tells CP the name of the CP QUERY VIRTUAL subcommand that you are enabling. The variable *subcommand* is a 1-character to 12-character alphanumeric string.

Set SUBCmd *subcommand*
tells CP the name of the CP SET subcommand that you are enabling. The variable *subcommand* is a 1-character to 12-character alphanumeric string.

IBMclass *
tells CP to enable all versions of the specified command or subcommand. If omitted, IBMCLASS * is the default.

IBMclass *c*
tells CP to enable a specific version of the specified command or subcommand. The variable *c* can be any 1 of the following:

A
this is a system-control command to be used by the primary system operator.

B
this is a command for operational control of real devices.

C
: this is a command to alter host storage.

D
: this is a command for system-wide control of spool files.

E
: this is a command to examine host storage.

F
: this is a command for service control of real devices.

G
: this is a general-use command used to control the functions of a virtual machine.

0
: (zero) this command has no specific IBM class assigned.

Usage Notes

1. To load the command processing code into the system execution space, use the CPXLOAD statement or command. For more information, see "CPXLOAD Statement" on page 76. See also CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
2. To activate a command while defining the command, use the ENABLE operand of the DEFINE COMMAND or CMD statement or command. For more information, see "DEFINE COMMAND / CMD Statement" on page 90. See also DEFINE COMMAND / CMD command in *z/VM: CP Commands and Utilities Reference*.
3. To deactivate a CP command:
   - To deactivate a CP command while defining it, use the DISABLE operand of the DEFINE COMMAND or CMD statement or command. For more information, see "DEFINE COMMAND / CMD Statement" on page 90. See also DEFINE COMMAND / CMD command in *z/VM: CP Commands and Utilities Reference*.
   - To deactivate a command after defining the command, use the DISABLE COMMAND or CMD statement or command. For more information, see "DISABLE COMMAND / CMD Statement" on page 128. See also DISABLE COMMAND / CMD in *z/VM: CP Commands and Utilities Reference*.
4. To change the definition of an existing CP command, use the MODIFY COMMAND or CMD statement or command. For more information, see "MODIFY COMMAND / CMD Statement" on page 188. See also MODIFY COMMAND / CMD in *z/VM: CP Commands and Utilities Reference*.
5. To display the address of the CP command table entry block, the current IBM class, and the current privilege class for a specified CP command, use the LOCATE CMDBK command. For more information, see LOCATE CMDBK in *z/VM: CP Commands and Utilities Reference*.
6. To remove the command processing code from the system execution space, use the CPXUNLOAD command. For more information, see CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.
7. For more information about enabling and disabling commands, see Defining and Modifying Commands and Diagnose Codes in *z/VM: CP Exit Customization*.

Examples

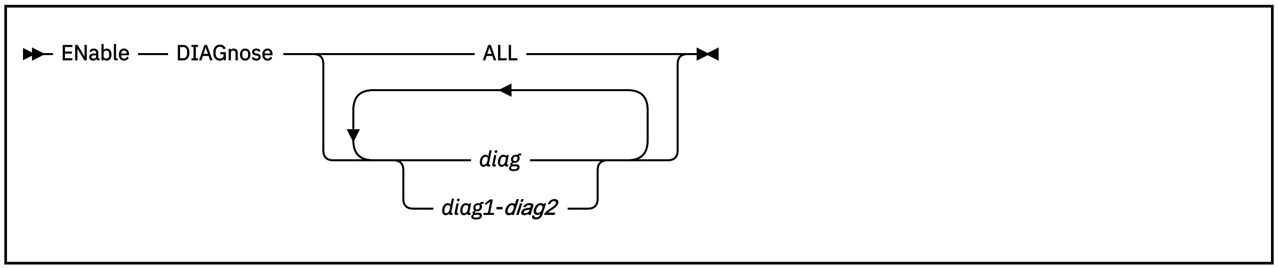
1. To activate the CP SHUTDOWN command (after a prior DISABLE COMMAND statement had deactivated it), use the following:

   ```
   Enable Command shutdown
   ```

2. To activate the class <ANY> version of the CP SET PRIVCLASS command (after a prior DISABLE COMMAND statement had deactivated it), use the following:

   ```
   Enable Command Set SubCmd privclass IBMclass 0
   ```

ENABLE DIAGNOSE Statement



Purpose

Use the ENABLE DIAGNOSE statement to permit CP to process requests for one or more locally-developed DIAGNOSE codes during and after initialization.

You can also permit processing of locally-developed DIAGNOSE codes after initialization using the ENABLE DIAGNOSE command. For more information, see *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place ENABLE DIAGNOSE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

ALL
tells CP to enable all existing DIAGNOSE codes.

Note: This operand enables all DIAGNOSE codes: the locally-defined ones, the IBM-supplied ones, and any supplied by third-party software vendors.

diag
diag1-diag2
is the number of the DIAGNOSE code that you are enabling. Each *diag* must be a hexadecimal number between X'0000' and X'03FC' and must be a multiple of 4. You can specify a single DIAGNOSE code, a range of DIAGNOSE codes, or any combination thereof.

Usage Notes

1. To define a new DIAGNOSE code, use the DEFINE DIAGNOSE statement or CP command. For more information, see "DEFINE DIAGNOSE Statement" on page 96. See also DEFINE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.

   Note: Unless you specify the ENABLE operand of the DEFINE DIAGNOSE statement or command, the new DIAGNOSE code is initially in a disabled state after being defined.
2. To load the DIAGNOSE processing code into the system execution space, use the CPXLOAD statement or CPXLOAD CP command. For more information, see "CPXLOAD Statement" on page 76. See also CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
3. If you do not specify the ENABLE operand, a new DIAGNOSE code is initially in a disabled state after being defined. CP treats disabled DIAGNOSE codes as if they were never defined. If you try to use a disabled DIAGNOSE code in a program, CP will give you a program check specification exception.
4. To change the definition of an existing DIAGNOSE code, use the MODIFY DIAGNOSE statement or command. For more information, see "MODIFY DIAGNOSE Statement" on page 192. See also MODIFY DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.

5. To display information about a DIAGNOSE code (status, entry point name, and privilege class) after initialization, use the QUERY DIAGNOSE command. For more information, see QUERY DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
6. To display the address of the CP DIAGNOSE code table block for a DIAGNOSE code after initialization, use the LOCATE DGNBK command. For more information, see LOCATE DGNBK in *z/VM: CP Commands and Utilities Reference*.
7. To deactivate a DIAGNOSE code after defining it, use the DISABLE DIAGNOSE statement or command. For more information, see "DISABLE DIAGNOSE Statement" on page 130. See also DISABLE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
8. To deactivate a DIAGNOSE code while defining it, use the DISABLE operand of the DEFINE DIAGNOSE statement or command. For more information, see "DEFINE DIAGNOSE Statement" on page 96. See also DEFINE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
9. To remove the DIAGNOSE processing code from the system execution space, use the CPXUNLOAD command. For more information, see CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.
10. Many external security managers (ESMs) do not support DIAGNOSE codes above X'03FC'. For this reason, CP does not support DIAGNOSE codes above X'03FC'. The DIAGNOSE codes between X'0000' and X'03FC' are divided as follows:

    X'0000' to X'00FC'
    Reserved for IBM use

    X'0100' to X'01FC'
    Reserved for customer use

    X'0200' to X'03FC'
    Reserved for IBM use.
11. For more information about user-defined DIAGNOSE codes, see Defining and Modifying Commands and Diagnose Codes in *z/VM: CP Exit Customization*.

Examples

1. To have CP enable DIAGNOSE code X'100', use the following:

```
Enable Diagnose 100
```

2. To have CP enable all DIAGNOSE codes on your system, use the following:

```
Enable Diagnose All
```

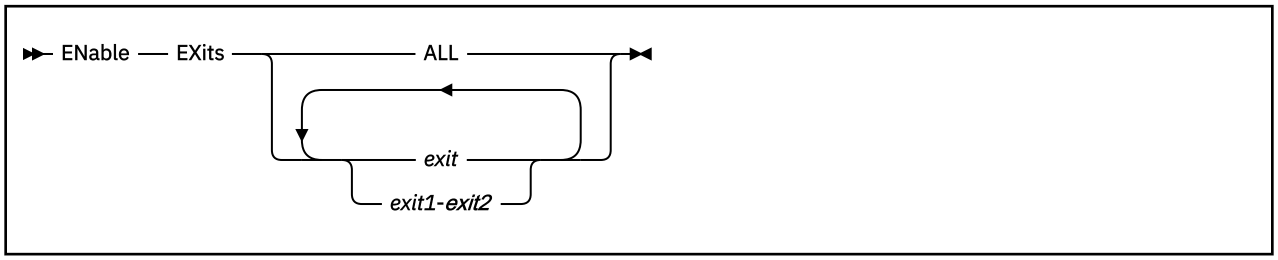
3. To have CP enable all locally-defined DIAGNOSE codes, use the following:

```
Enable Diagnose 100-1fc
```

4. To have CP enable all locally-defined DIAGNOSE codes, except DIAGNOSE code X'180', use the following:

```
Enable Diagnose 100-17c 184-1fc
```

ENABLE EXITS Statement



Purpose

Use the ENABLE EXITS statement to permit CP to call all entry points and external symbols associated with one or more exit points during and after initialization.

You can also enable CP to call the entry points and external symbols that are associated with an exit point after initialization by using the ENABLE EXITS command. For more information, see ENABLE EXITS in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place ENABLE EXITS statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

ALL
tells CP to enable all existing CP exit points.

exit
exit1-exit2
is the number of the exit point (or exit points) that you want CP to start using. Each *exit* must be a hexadecimal number between X'0000' and X'FFFF'. You can specify a single exit point number, a range of exit point numbers, or any combination thereof.

Usage Notes

1. To load the exit point code into the system execution space, use the CPXLOAD statement or command. For more information, see "CPXLOAD Statement" on page 76. See also CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
2. To associate one or more entry points or external symbols with a specific exit point and to enable or disable that exit point, use the ASSOCIATE EXIT statement or command. For more information, see "ASSOCIATE EXIT Statement" on page 58. See also ASSOCIATE EXIT in *z/VM: CP Commands and Utilities Reference*. You can also use the ASSOCIATE EXIT statement to change the entry points and external symbols that are associated with a specific entry point.
3. If the list of entry points and external symbols associated with this exit point contain any entry points or external symbols that CP does not know about, CP just ignores them and continues normal processing. That is, CP will continue to process the other members of the list associated with this exit point. CP does not ignore an exit point because it cannot find one entry point or external symbol in the list. CP only ignores an exit point if it cannot find all the entry points and external symbols in the list.
4. To display whether there are any unknown entry points or external symbols associated with an exit point, use the QUERY UNRESOLVED command. For more information, see QUERY UNRESOLVED in *z/VM: CP Commands and Utilities Reference*.

5. To display status and usage statistics information about a specific exit point after initialization, use the QUERY EXITS command. For more information, see QUERY EXITS in *z/VM: CP Commands and Utilities Reference*.
6. To display the address of the CP exit block for a specific exit point after initialization, use the LOCATE XITBK command. For more information, see LOCATE XITBK in *z/VM: CP Commands and Utilities Reference*. Again, if you have not associated one or more entry points or external symbols with the specified exit point, there is no CP exit block for CP to locate and display. Instead, CP issues error message HCP2752E.
7. To display the address of the CP indirect call locator block for a specific exit point, use the LOCATE ICLBK command. For more information, see LOCATE ICLBK in *z/VM: CP Commands and Utilities Reference*.
8. To change the definition of an existing dynamic exit point, or remove the exit point from the system, use the MODIFY EXIT statement or command. For more information, see "MODIFY EXIT Statement" on page 195. See also MODIFY EXIT in *z/VM: CP Commands and Utilities Reference*.
9. To stop CP from calling the entry points and external symbols associated with one or more exit points after defining those exit points, use the DISABLE EXITS command. For more information, see DISABLE EXITS in *z/VM: CP Commands and Utilities Reference*.
10. To remove the customer-written CP routines from the system execution space:
    a. Use the DISASSOCIATE command to revoke all entry point and external symbol assignments that were made with the ASSOCIATE EXIT statement or command. For more information, see DISASSOCIATE in *z/VM: CP Commands and Utilities Reference*.
    b. Use the CPXUNLOAD command to unload the customer-written CP routines. For more information, see CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.
11. For more information about user-defined exit points, see Benefits of using CP exits in *z/VM: CP Exit Customization*.

Examples

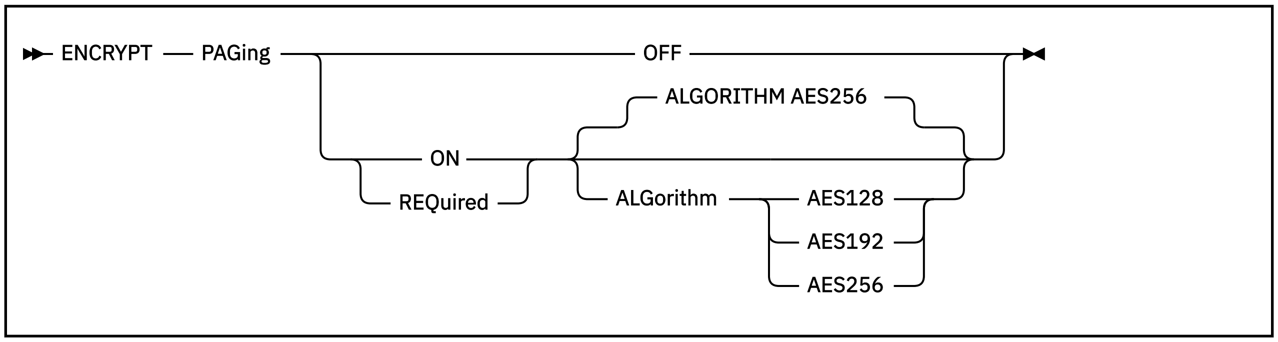
1. To have CP start using the entry points and external symbols associated with CP Exit 99 after initialization, use the following:

```
Enable Exits 99
```

2. To have CP start using the entry points and external symbols associated with all CP exit points after initialization, use the following:

```
Enable Exits All
```

ENCRYPT Statement



Purpose

Use the ENCRYPT statement to specify settings for your system's host level encryption.

How To Specify

The ENCRYPT statement is optional. You can place it anywhere in the system configuration file. If you specify more than one ENCRYPT statement, the last statement overrides any previous specifications.

Operands

PAGing
: indicates the host function modified in this statement is CP paging of guest memory and virtual-disk-in-storage (VDISK).

OFF
: disables host level encryption for the function indicated.

ON
: enables host encryption for the function indicated. If the required hardware support is not available, encryption reverts to OFF and the system IPL continues.

REQUIRED
: enables host-level encryption for the function indicated, and locks this setting until the next system IPL. That is, once you specify REQUIRED, the SET ENCRYPT command cannot be used for this CP function. If hardware support is not available, the system IPL will fail.

ALGorithm
: Specifies the symmetric encryption cipher to be used by this host function. All algorithms currently supported require a specific level of CPACF (hardware feature 3863) to be enabled for the system. For more information, see *z/VM: Migration Guide*.

AES128
AES192
AES256
: indicates the Advanced Encryption Standard (AES) algorithm is to be used for this host service, with CBC block mode. The bit size of the key determines the strength of the encryption to be performed.

Usage Notes

1. By default, host-level encryption is set to OFF.
2. Encryption can be enabled only if the appropriate hardware support exists for your LPAR or CEC. If ON is specified and the LPAR in which the z/VM system is running is missing such support, an error message is issued, but IPL continues.

3. REQUIRED should be specified only on systems where appropriate hardware support exists. Specifying REQUIRED on a system without the appropriate hardware causes the system IPL to fail with a wait-state condition of HCP1393W.

   Therefore, it is recommended that REQUIRED be enabled dynamically (via the SET ENCRYPT command) rather than specified in the system configuration statement. For environments requiring 100% encryption compliance, this can be accomplished by setting ENCRYPT PAGING ON in the system configuration file, and then inserting the SET ENCRYPT PAGING REQUIRED command at an early point in the system IPL process. For example, you can add SET ENCRYPT PAGING REQUIRED to:

   - a COMMAND statement in the CP directory for the system operator virtual machine (OPERATOR), or
   - the PROFILE EXEC of the AUTOLOG1 virtual machine.

   For best practices in specifying the ENCRYPT statement and the SET ENCRYPT command, see Chapter 28, “Device Encryption Planning,” on page 707.
4. The algorithm can be selected at the time of the IPL, or the first time ENCRYPT is enabled on the system. Once selected, the algorithm cannot be changed until the next system IPL.
5. Enabling encryption will increase CPU utilization relative to the strength of the encryption algorithm selected. For more information, see Major Factors Affecting Performance in *z/VM: Performance*.

Examples

To cause the paging data that is stored on CP-owned DASD volumes to be encrypted, use the following ENCRYPT statement:

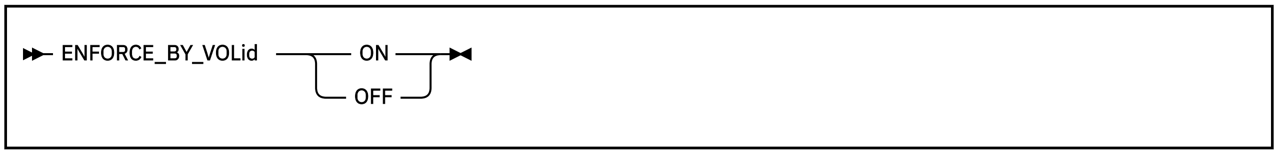
```
ENCRYPT PAGING ON ALGORITHM AES256
```

To ensure that encryption of paging data is disabled at the time of system IPL, use the following ENCRYPT statement:

```
ENCRYPT PAGING OFF
```

ENCRYPT PAGING OFF is the default setting for z/VM.

ENFORCE_BY_VOLID Statement



Purpose

Use the ENFORCE_BY_VOLid configuration statement to enforce attachment of DASD devices by their VOLIDs on the ATTACH command.

How to Specify

The ENFORCE_BY_VOLid statement is optional. If you do not specify the statement, CP will not enforce attachment of DASD devices by their VOLIDs. If you specify the statement more than once, CP will use the last ENFORCE_BY_VOLid statement entered.

Operands

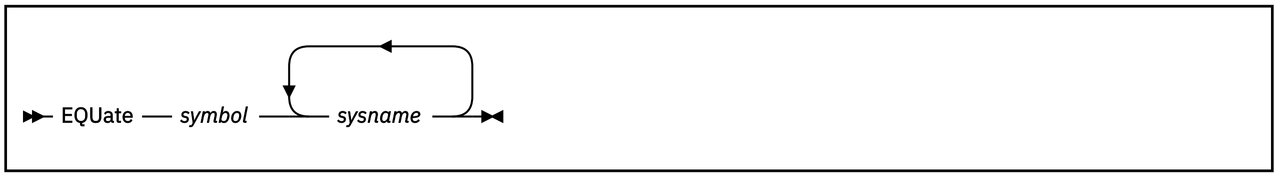
ON
tells CP to enforce attachment of DASD devices by VOLID only.

OFF
tells CP not to enforce attachment of DASD devices by VOLID only.

Usage Notes

1. When the ENFORCE_BY_VOLID ON statement is specified, the VOLIDs for all DASD devices must be specified on the CP ATTACH command and the DEDICATE directory statement. For more information, see "DEDICATE Directory Statement" on page 500. See also ATTACH in *z/VM: CP Commands and Utilities Reference*.

EQUATE Statement



Purpose

Use the EQUATE statement to create nicknames for systems or groups of systems. After creating a nickname, you can use it as a record qualifier in the system configuration file to limit the scope of the statement and to group systems that have common properties. This allows you to create generic system configuration files that can be shared across multiple systems.

How to Specify

Include as many statements as needed; they are optional. You must define a nickname before you use it, so place your EQUATE statements anywhere in the system configuration file before any statements using those nicknames. If you specify more than one statement with the same nickname, CP uses the last statement. CP will not combine multiple statements with the same nickname into one comprehensive group.

Operands

symbol
is the nickname for a system or group of systems that you are grouping together because they have similar properties and should therefore be treated the same way when CP processes the system configuration file. Each *symbol* is a 1- to 16-character alphanumeric nickname with no imbedded blanks.

sysname
is the name of the system (or systems) you want included in the specified nickname group. You can use generic system names to request a specific subset of systems. A generic system name is a 1- to 8-character string with asterisks (*) in place of one or more arbitrary characters and percent signs (%) in place of exactly one arbitrary character. For example:

```
Equate yorktown y%tvm*
```

creates a nickname that includes all systems that start with Y and have TVM as their third, fourth, and fifth characters.

Usage Notes

1. A *sysname* must be defined on a SYSTEM_IDENTIFIER statement before it can be included in a nickname group. For more information, see "SYSTEM_IDENTIFIER Statement" on page 287.
2. You can use multiple names in an EQUATE statement and still distinguish between the systems. For example:

```
Equate  vm3  sys1  sys2  sys4
```

creates the nickname VM3 for systems SYS1, SYS2, and SYS4. If you later have statements using these names:

```
Sys4: …
Sys1: …
Vm3:  …
```

The statement starting with Sys4: only applies to system SYS4; the statement starting with Sys1: only applies to system SYS1; and the statement starting with Vm3: applies to systems SYS1, SYS2, and SYS4.

Examples

1. If you have five systems set aside for business and five set aside for research, you can use an EQUATE statement to give each of the two groups a nickname. Then you can use the nicknames as record qualifiers on other statements in the system configuration file.

   For example:

   ```
   Equate business atlanta boston,     /* Define BUSINESS nickname     */
                   chicago clevland,
                   newyork
   Equate research maine raleigh,      /* Define RESEARCH nickname     */
                   rdsys1 testsys,
                   yorktown
    .
    .
    .
   Business: Features Disable Set_PrivClass  /* Don't let users set    */
                                             /* their own privilege    */
                                             /* classes                */
    .
    .
    .
   Research: Features Enable Set_PrivClass   /* Let users set their    */
                                             /* own privilege classes  */
   ```

   allows the users on the five research systems to change their own privilege classes and allows the system operator on those system to change the privilege classes of any users logged on those systems. It also prevents the users and system operators on the five business systems from changing privilege classes.

EXTERNAL_SYNTAX Statement

```
►► EXTERNAL_SYNtax ── EPName ── epname ─►◄
```

Purpose

Use the EXTERNAL_SYNTAX statement to add locally-developed system configuration file statements to the system without modifying the system configuration file processor, HCPZSC ASSEMBLE.

How to Specify

Include as many statements as needed; they are optional. You can place EXTERNAL_SYNTAX statements anywhere in the system configuration file, as long as they appear before the first invocation of the new configuration file statement. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

EPName *epname*
tells CP the name of the external symbol that identifies the start of your syntax definition. This would be the label that is generated or coded on the HCPDOSYN macro which defines the syntax of your locally-developed system configuration file statement.

Usage Notes

1. If your locally-developed system configuration file statement has the same name (or minimum abbreviation) as a system configuration file statement shipped by IBM, your statement will override the existing statement.

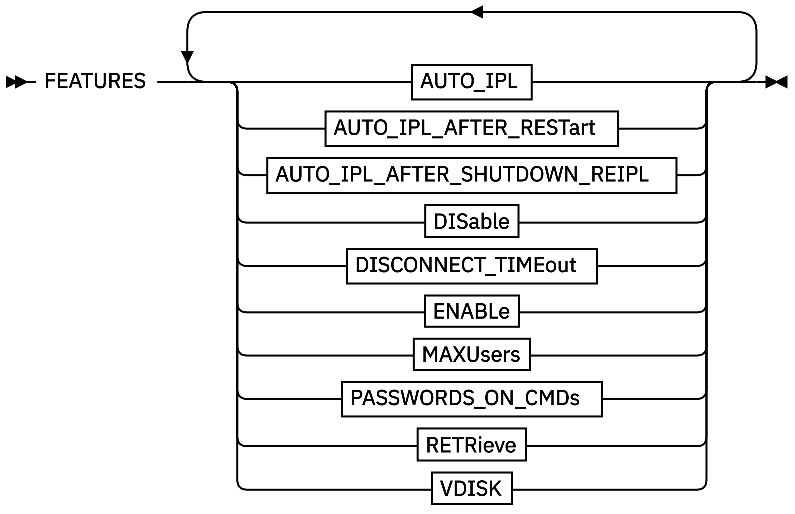
Examples

1. To define and use your own system configuration file statement during initialization, use the following:

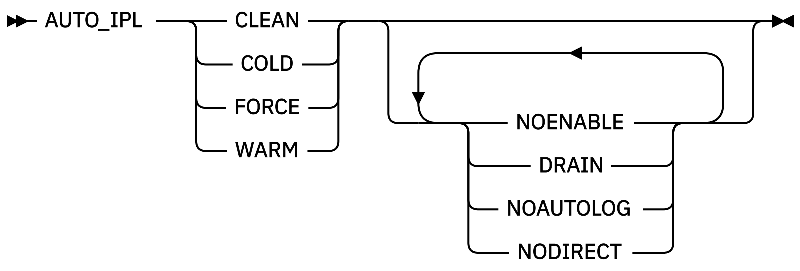
```
CPXLoad tttsyn text a          /* Load the module                    */
               Temporary,      /* ... in case we must unload it later    */
               NoControl,      /* ... to unload without more checking    */
               Nodelay,        /* ... load immediately from parmdisk     */
               Lock            /* ... make sure all of it is in storage  */
                               /*     if it is > 4K.  (Not necessary,    */
                               /*     unless you are paranoid, like me!) */

/*  Let us hook into the front of the standard syntax definitions       */

External_Syntax EPname tttsyntx
```

In this example, the module TTTSYN contains your syntax tree, which starts at label TTTSYNTX. This example shows how you would get CP to load the module and make CP consider its additional syntax during initialization.

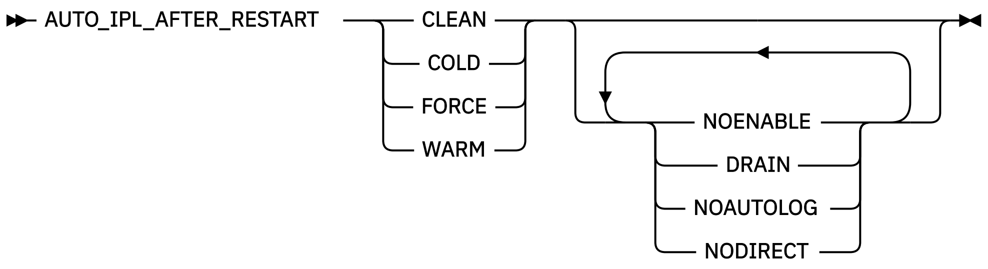
FEATURES Statement



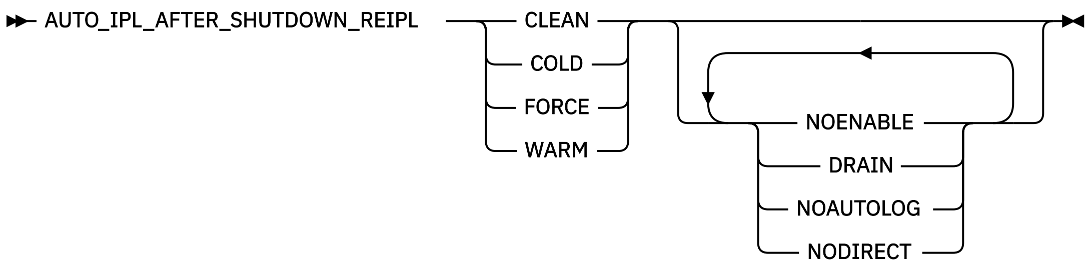
AUTO_IPL



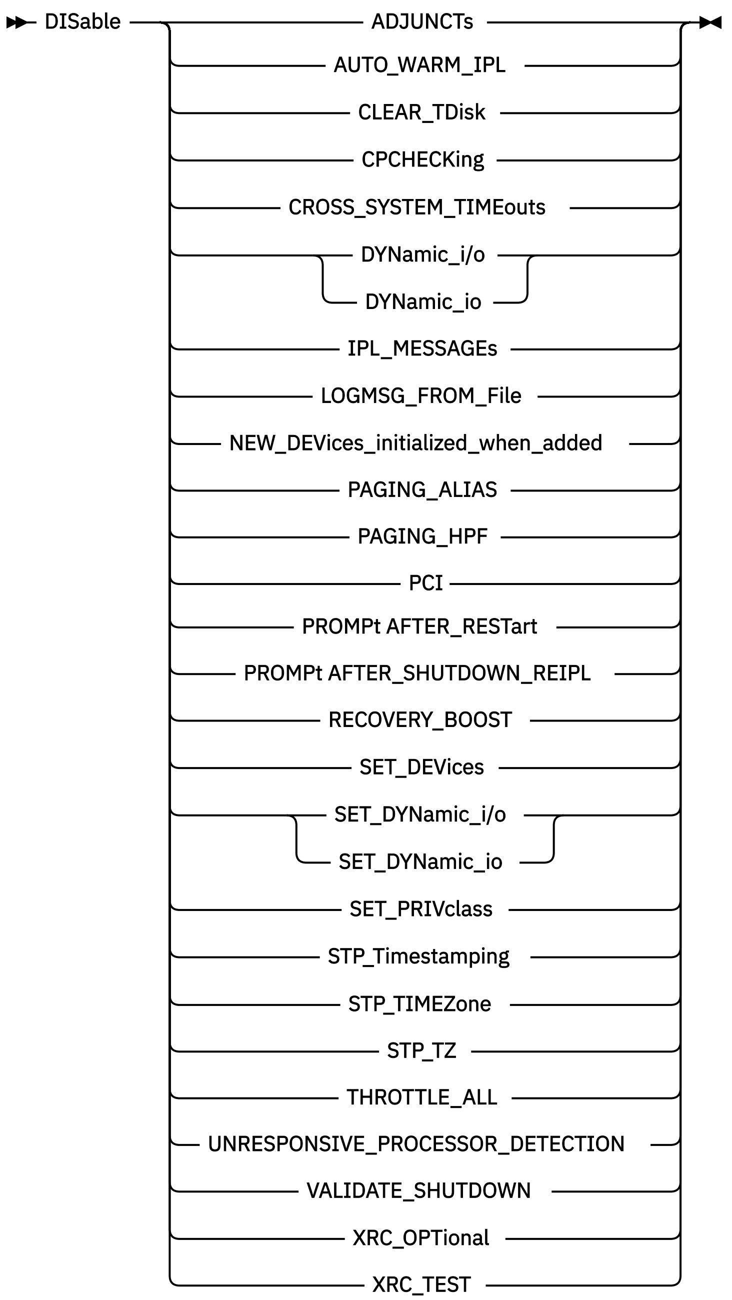
AUTO_IPL_AFTER_RESTART



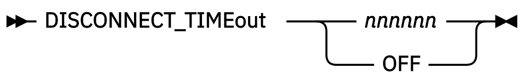
AUTO_IPL_AFTER_SHUTDOWN_REIPL



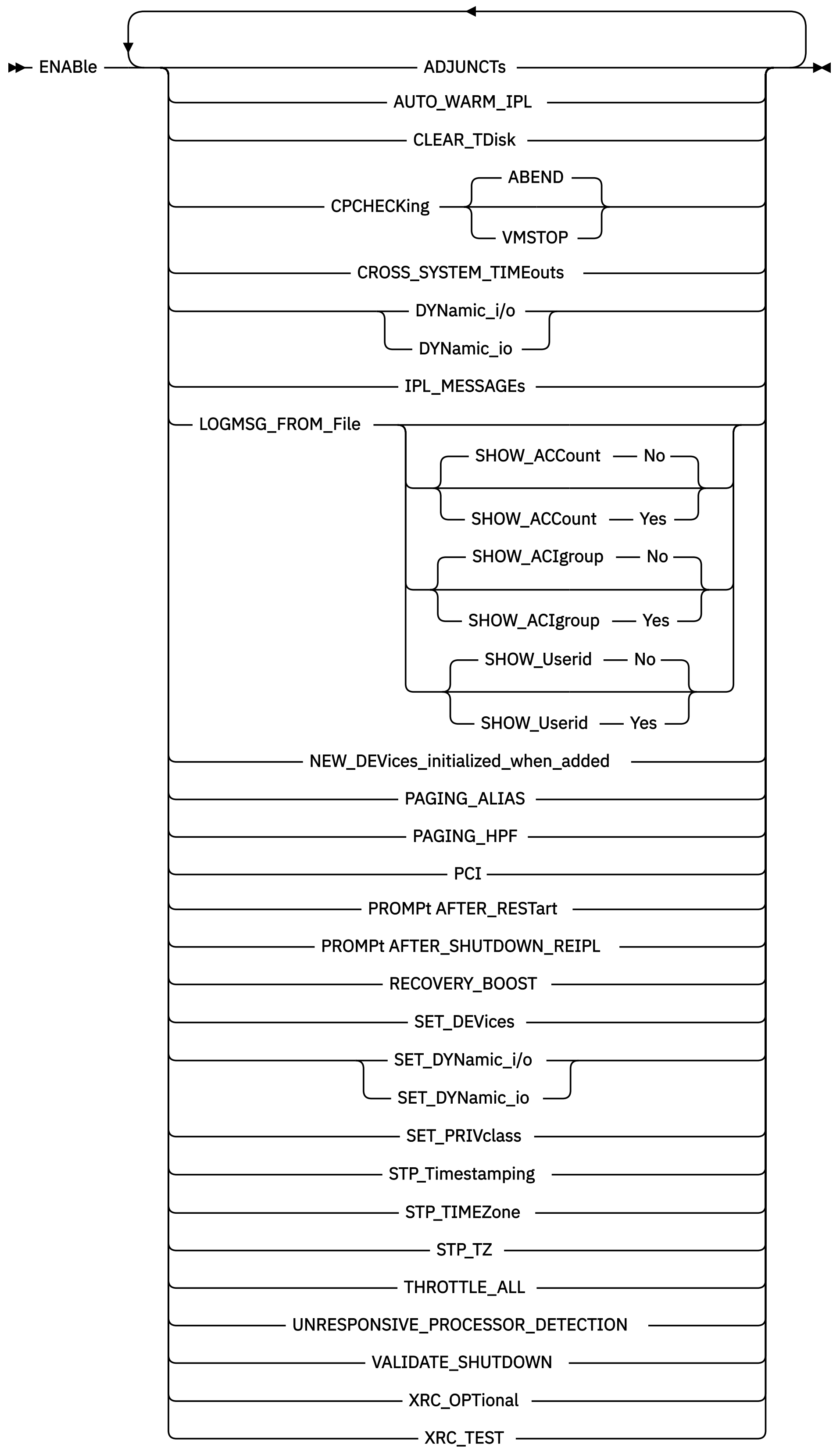
DISable



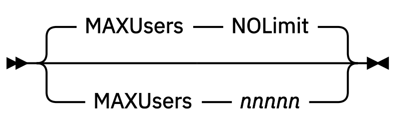
DISCONNECT_TIMEout



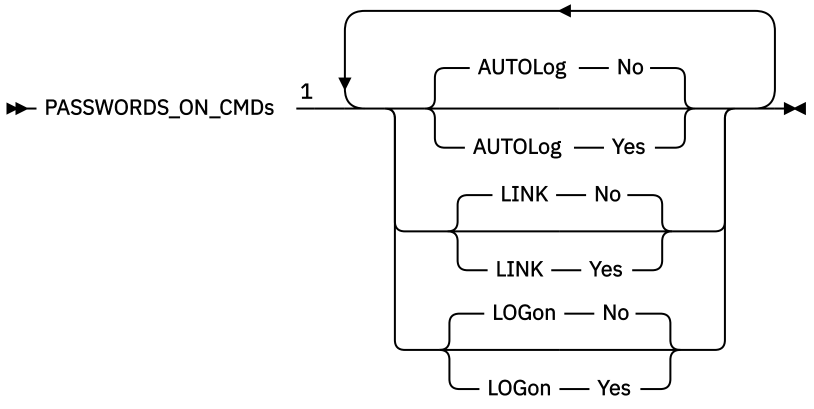
ENABle


ENABle
ADJUNCTs
AUTO_WARM_IPL
CLEAR_TDisk
CPCHECKing
ABEND
VMSTOP
CROSS_SYSTEM_TIMEouts
DYNamic_i/o
DYNamic_io
IPL_MESSAGEs
LOGMSG_FROM_File
SHOW_ACCount — No
SHOW_ACCount — Yes
SHOW_ACIgroup — No
SHOW_ACIgroup — Yes
SHOW_Userid — No
SHOW_Userid — Yes
NEW_DEVices_initialized_when_added
PAGING_ALIAS
PAGING_HPF
PCI
PROMPt AFTER_RESTart
PROMPt AFTER_SHUTDOWN_REIPL
RECOVERY_BOOST
SET_DEVices
SET_DYNamic_i/o
SET_DYNamic_io
SET_PRIVclass
STP_Timestamping
STP_TIMEZone
STP_TZ
THROTTLE_ALL
UNRESPONSIVE_PROCESSOR_DETECTION
VALIDATE_SHUTDOWN
XRC_OPTional
XRC_TEST

MAXUsers



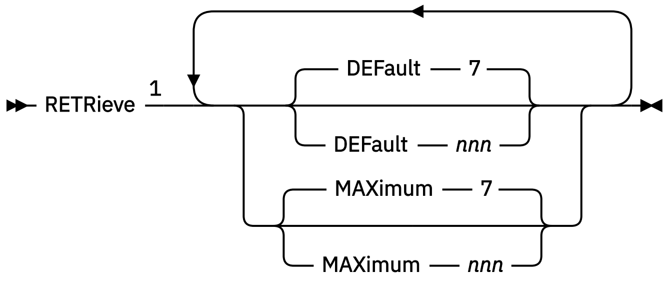
PASSWORDS_ON_CMDs



Notes:

[1] You must specify at least one operand.

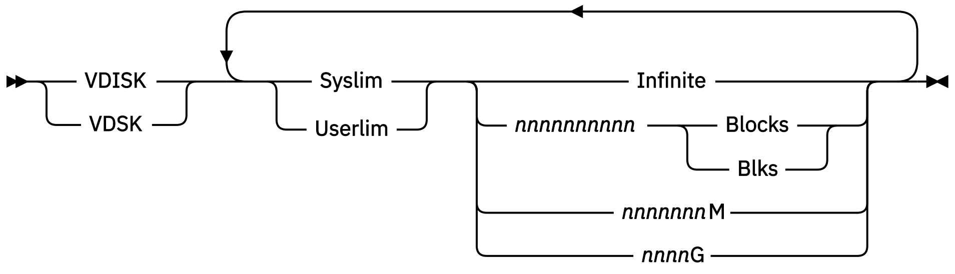
RETRieve



Notes:

[1] You must specify at least one operand.

VDISK



Purpose

Use the FEATURES statement to set certain attributes of the system at system initialization.

How to Specify

Include as many statements as needed; they are optional. You can place FEATURES statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Attention: If the FEATURES statement contains a reference to an unsupported feature, the entire FEATURES statement is rejected. If you have a FEATURES statement for a more advanced level of z/VM, but you IPL an earlier level of z/VM that does not support the advanced feature, the entire FEATURES statement is rejected. A better practice is to use one FEATURES statement per feature. In this way, only the single FEATURES statement with the unsupported feature would be rejected.

Operands

AUTO_IPL
defines the kind of start to be performed when the system is initialized. The system will be started without prompting the system operator and without changing the TOD clock.

AUTO_IPL_AFTER_RESTart
defines the kind of start to be performed when the system is initialized after a system restart. The system will be restarted without prompting the system operator and without changing the TOD clock

AUTO_IPL_AFTER_SHUTDOWN_REIPL
defines the kind of start to be performed when the system is initialized after a SHUTDOWN REIPL. The system will be restarted without prompting the system operator and without changing the TOD clock.

COLD
tells CP to perform a cold start. This will purge all spool files, accounting records, error recording records, symptom records, and the system log message. System data files will not be purged.

If spooling errors that could result in the loss of system data files (NSS, DCSS, TRF, IMG, UCR, NLS) are encountered, the system operator will be prompted and given the opportunity to stop. If no spooling errors are encountered, the system operator will not be prompted and there will be no opportunity to stop.

CLEAN
tells CP to perform a clean start. This will purge all spool files, system data files, accounting records, error recording records, symptom records, and the system log message. The system operator will not be prompted and there will be no opportunity to stop.

FORCE
tells CP to perform a force start. If spooling errors are encountered, the spool files and system data files in error will be purged. The system operator will not be prompted and there will be no opportunity to stop.

WARM
tells CP to perform a warm start. If spooling errors are encountered, the system operator will be prompted and given the opportunity to stop.

NOENABLE
tells CP not to enable terminal access after system initialization.

DRAIN
tells CP to drain unit-record devices after system initialization.

NOAUTOLOG
tells CP to bypass automatic logon of virtual machines after system initialization.

NODIRECT
tells CP to bring up the system without a User Directory.

DISable
disables the following system attributes during IPL. Except as noted, each option is initially disabled until enabled using the ENABLE operand.

ADJUNCTs
tells CP not to allow adjunct virtual machine support. The default is to enable adjunct support.

AUTO_WARM_IPL
tells CP to go through all the usual prompts in the IPL process.

CLEAR_TDisk
tells CP to clear only cylinder 0 or the first eight blocks on the temporary minidisk when it detaches the minidisk. The default for automatic clearing is ON.

CPCHECKing
tells CP that internal CP checking is not to be executed.

CROSS_SYSTEM_TIMEouts
specifies that timeouts detected at the device and logical link level within the ISFC inter-system communications component are ignored. Also, in an SSI cluster environment, CROSS_SYSTEM_TIMEOUTS specifies that missing or late heartbeat signals between members in the SSI cluster are ignored. Disabling CROSS_SYSTEM_TIMEOUTS prevents the SSI member from recognizing unresponsive member systems and from initiating normal error recovery.

This option is intended for use in diagnostic situations only—typically, when running members of an SSI cluster as z/VM guests, in which a member might be stopped for long periods of time when using the CP TRACE command or similar facilities to debug code within the CP nucleus or a CPXLOADed nucleus extension. Without this option in effect, manual commands such as SET SSI DOWN, DEACTIVATE ISLINK, and ACTIVATE ISLINK might be required to restore proper operation.

In order to join an SSI cluster, a member's CROSS_SYSTEM_TIMEOUTS setting (either ENABLE CROSS_SYSTEM_TIMEOUTS or DISABLE CROSS_SYSTEM_TIMEOUTS) must be the same as that of the first member who joined; a nonconforming member is not permitted to join the cluster.

DYNamic_i/o
DYNamic_io
tells CP not to allow dynamic I/O changes on this processor.

IPL_MESSAGEs
tells CP not to display IPL or SHUTDOWN messages or prompts during system initialization.

If IPL_MESSAGES are disabled and spooling errors are encountered during an automatic WARM or COLD IPL, wait state HCP2516W will be issued.

LOGMSG_FROM_File
tells CP not to look for any LOGMSG files on disk. Instead, CP should use information from the class B CP SET LOGMSG command. For more information, see SET LOGMSG in *z/VM: CP Commands and Utilities Reference*.

NEW_DEVices_initialized_when_added
tells CP to create a real device control block (RDEV) for a new I/O device, but not to initialize (bring online) that device when your system receives an I/O machine check (IOMCK) for adding a new device to the system. To bring the device online, use the CP VARY (Real Device) command. For more information, see VARY (Real Device) in *z/VM: CP Commands and Utilities Reference*.

PAGING_ALIAS
tells CP not to use HyperPAV aliases for paging and not to give the system operator the ability to use the CP SET PAGING command to affect the usage of HyperPAV aliases for paging. For more information, see SET PAGING in *z/VM: CP Commands and Utilities Reference*.

To consolidate paging subsystem options, it is recommended that you use the PAGING ALIAS statement in the system configuration file rather than using the PAGING_ALIAS operand on the FEATURES statement.

PAGING_HPF
tells CP not to use transport-mode channel programs for paging and not to give the system operator the ability to use the CP SET PAGING command to affect the usage of transport-mode channel programs for paging. For more information, see SET PAGING in *z/VM: CP Commands and Utilities Reference*.

To consolidate paging subsystem options, it is recommended that you use the PAGING HPF statement in the system configuration file rather than using the PAGING_HPF operand on the FEATURES statement.

PCI
is ignored and has no effect. PCI functions are always allowed to come online to the system for use.

PROMPt AFTER_RESTart
tells CP not to force a prompt to the operator when CP bounces.

PROMPt AFTER_SHUTDOWN_REIPL
tells CP not to force a prompt to the operator when CP is performing a SHUTDOWN REIPL.

RECOVERY_BOOST
requests that CP not use the System Recovery Boost machine facility. The System Recovery Boost allows z/VM to boost general purpose processors, running as subcapacity, to full capacity for up to 60 minutes during z/VM system initialization and workload bring-up, and for up to 30 minutes during workload quiesce, system shutdown, and system abend processing. The default is to enable the facility.

SET_DEVices
tells CP not to allow users to execute CP SET DEVICES commands to change the way the way CP handles specific real devices after initialization. For more information, see SET DEVICES in *z/VM: CP Commands and Utilities Reference*.

SET_DYNamic_i/o
SET_DYNamic_io
tells CP not to allow users to execute CP SET DYNAMIC_I/O commands to enable or disable CP's ability to dynamically change the processor's I/O configuration after initialization. For more information, see SET DYNAMIC_I/O in *z/VM: CP Commands and Utilities Reference*.

SET_PRIVclass
tells CP not to give end users the authority to use the CP SET PRIVCLASS command to change their own privilege classes, and tells CP not to give the system operator the authority to use the CP SET PRIVCLASS command to change the privilege classes of users logged on to the system. For more information, see SET PRIVCLASS in *z/VM: CP Commands and Utilities Reference*.

STP_Timestamping
tells CP not to enable the STP protocol for timestamping purposes.

STP_TIMEZone / STP_TZ
tells CP not to enable the STP protocol in order to obtain timezone information automatically from the STP server.

THROTTLE_ALL
tells CP to allow throttling of ALL devices on the system except CP OWNED DASD.

UNRESPONSIVE_PROCESSOR_DETECTION
specifies that detection of unresponsive processors will not occur on second level systems. A common cause of an unresponsive processor is that it is looping continuously and is no longer doing productive work. If the master processor is unresponsive, the system will appear hung and unlikely to be able to process commands. Disabling unresponsive processor detection prevents the CP from recognizing an unresponsive master or non-master processor and from initiating normal error recovery, which could be to restart the unresponsive processor or could be to abend the system.

This option is intended for use in diagnostic situations when running as a second level system where CP might be stopped for long periods of time when using the CP TRACE command or similar facilities to debug code within the CP nucleus or a CPXLOADed nucleus extension. Virtual processors that aren't being traced could run enough to recognize the traced processor is being unresponsive. Specifying DISABLE UNRESPONSIVE_PROCESSOR_DETECTION only affects CP when it is running second level; when running first, this specification is ignored.

VALIDATE_SHUTDOWN
tells CP that the SYSTEM operand on the SHUTDOWN command is not required. This is the default.

XRC_OPTional
tells CP not to allow non-timestamped I/O to be issued whenever STP is in an unsynchronized state.

XRC_TEST
tells CP not to timestamp I/O when CP is running as a virtual machine guest.

DISCONNECT_TIMEout *nnnnnn*
sets the interval between a forced disconnect of a virtual machine and its logoff to the specified number of minutes. The default is 15 minutes.

DISCONNECT_TIMEout OFF
disables the automatic logoff of a virtual machine that is forcibly disconnected.

ENABle
enables the following system attributes during IPL. Except as noted, each option is initially disabled until it is enabled.

ADJUNCTs
tells CP to allow adjunct virtual machine support if authorized and defined in the user directory entry for that virtual machine. This is the default setting for this feature.

AUTO_WARM_IPL
tells CP to attempt a warm start without changing the TOD clock. If there is no warm start data, if the TOD clock is not set, or if spooling errors are encountered, CP will prompt the operator for more information.

CLEAR_TDisk
tells CP to automatically clear all previously written data and directory areas on TDISK DASD space. CP will change TDISK DASD space to binary zeros during CP initialization, when attaching a CP-owned volume that contains TDISK allocations, and when a user detaches a TDISK minidisk. By clearing this space, you prevent users from accidentally accessing old temporary disk space and provides your system with additional security. The default for automatic clearing is ON.

CPCHECKing
tells CP to activate internal CP checking to confirm assertions established in the CP code and take the appropriate action based on whether ABEND or VMSTOP is specified.

ABEND
Indicates that an abend should occur. Whether it is a hard abend or a soft abend is controlled by the specific assertion case. It is typically a hard abend. This is the default when internal CP checking is activated.

VMSTOP
This parameter is permitted only when CP is itself running in a virtual machine, and is generally useful only for debugging CP. Specifying this parameter causes CP to issue DIAGNOSE code X'8' when an untrue assertion is encountered, specifying a command string length of zero. This causes the virtual machine to stop and for CP to post a read to the console. A message is sent to the console (using DIAGNOSE code X'8' and the CP MESSAGE * command) providing information about the cause of the stop.

CROSS_SYSTEM_TIMEouts
specifies that timeouts caused by device and logical link failures within the ISFC inter-system communications component are treated as errors. Also, in an SSI cluster environment, CROSS_SYSTEM_TIMEOUTS causes error processing to occur for missing or late heartbeat signals between members.

This option is the default setting and is what should be used in a production environment in order to ensure that the ISFC collection or SSI cluster reacts properly to network and member failures.

In order to join an SSI cluster, a member's CROSS_SYSTEM_TIMEOUTS setting (either ENABLE CROSS_SYSTEM_TIMEOUTS or DISABLE CROSS_SYSTEM_TIMEOUTS) must be the same as that of the first member who joined; a nonconforming member is not permitted to join the cluster.

DYNamic_i/o
DYNamic_io
tells CP to allow dynamic I/O changes on this processor.

IPL_MESSAGEs
tells CP to display IPL or SHUTDOWN messages and prompts during system initialization (the default).

LOGMSG_FROM_File
tells CP to display the contents of the SYSTEM LOGMSG file on the lowest accessed CP disk. You can optionally choose to have CP read and display additional files at a user's terminal using one or more of the following operands:

SHOW_ACCount No
(the default) tells CP not to display a file called *accountid* LOGMSACC (*accountid* is the account ID of a specific user) in response to a CP QUERY LOGMSG command.

SHOW_ACCount Yes
tells CP to display a file called *accountid* LOGMSACC (*accountid* is the account ID of a specific user) in response to a CP QUERY LOGMSG command.

SHOW_ACIgroup No
(the default) tells CP not to display a file called *acigroup* LOGMSACI (*acigroup* is the ACI group of a specific user) in response to a CP QUERY LOGMSG command.

SHOW_ACIgroup Yes
tells CP to display a file called *acigroup* LOGMSACI (*acigroup* is the ACI group of a specific user) in response to a CP QUERY LOGMSG command.

SHOW_Userid No
(the default) tells CP not to display a file called *userid* LOGMSUSR (*userid* is the user ID of a specific user) in response to a CP QUERY LOGMSG command.

SHOW_Userid Yes
tells CP to display a file called *userid* LOGMSUSR (*userid* is the user ID of a specific user) in response to a CP QUERY LOGMSG command.

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

NEW_DEVices_initialized_when_added
tells CP to automatically create a real device control block (RDEV) and initialize (bring online) the associated I/O device when you add a new device, causing an I/O machine check (IOMCK).

PAGING_ALIAS
tells CP to use HyperPAV aliases when supported by the target paging device and to give the system operator the ability to use the CP SET PAGING command to affect the usage of HyperPAV aliases for paging. For more information, see SET PAGING in *z/VM: CP Commands and Utilities Reference*.

PAGING_HPF
tells CP to use transport-mode channel programs when supported by the target paging device and to give the system operator the ability to use the CP SET PAGING command to affect the usage of transport-mode channel programs for paging. For more information, see SET PAGING in *z/VM: CP Commands and Utilities Reference*.

For optimal use of HPF, it is recommended that you do not have a mixed environment of HPF-capable and non-HPF-capable channel paths to a paging device.

PCI
is ignored and has no effect. PCI functions are always allowed to come online to the system for use.

PROMPt AFTER_RESTart
tells CP to force a prompt to the operator when CP bounces, so that REIPL can be stopped or the type of start desired (WARM, FORCE, etc.) can be specified.

PROMPt AFTER_SHUTDOWN_REIPL
tells CP to force a prompt to the operator when CP is performing a SHUTDOWN REIPL, so that REIPL can be stopped or the type of start desired (WARM, FORCE, etc.) can be specified.

Note that in this case, the user will not be prompted to change the Time of Day (TOD) clock.

RECOVERY_BOOST
requests that CP use the System Recovery Boost machine facility. The System Recovery Boost allows z/VM to boost general purpose processors, running as subcapacity, to full capacity for up to 60 minutes during z/VM system initialization and workload bring-up, and for up to 30 minutes during workload quiesce, system shutdown, and system abend processing. The default is to enable the facility.

SET_DEVices
tells CP to allow users to execute CP SET DEVICES commands to change the way the way CP handles specific real devices after initialization. For more information, see SET DEVICES in *z/VM: CP Commands and Utilities Reference*.

SET_DYNamic_i/o
SET_DYNamic_io
tells CP to allow users to execute CP SET DYNAMIC_I/O commands to enable or disable CP's ability to dynamically change the processor's I/O configuration after initialization. For more information, see SET DYNAMIC_I/O in *z/VM: CP Commands and Utilities Reference*.

SET_PRIVclass
tells CP to give end users the authority to use the CP SET PRIVCLASS command to change their own privilege classes, and tells CP to give the system operator the authority to use the CP SET PRIVCLASS command to change the privilege classes of users logged on to the system. For more information, see SET PRIVCLASS in *z/VM: CP Commands and Utilities Reference*.

STP_Timestamping
tells CP to enable the STP protocol (if the STP facility is installed) and apply timestamps to all XRC-capable DASD devices.

STP_TimeZone / STP_TZ
tells CP to enable the STP protocol (if the STP facility is installed) and obtain timezone information automatically from the STP server.

THROTTLE_ALL
tells CP to allow throttling of ALL devices on the system including CP OWNED DASD.

UNRESPONSIVE_PROCESSOR_DETECTION
specifies that detection of unresponsive processors occurs. A common cause of an unresponsive processor is that it is looping continuously and is no longer doing productive work. If the master processor is unresponsive, the system will appear hung and unlikely to be able to process commands. Detection of unresponsive processors is the default and allows CP to initiate normal error recovery, which could be to restart the unresponsive processor or could be to abend the system.

VALIDATE_SHUTDOWN
tells CP to require the user to specify the SYSTEM operand on the SHUTDOWN command.

XRC_OPTional
when STP_Timestamping is also enabled, this will allow non-timestamped I/O to be issued whenever STP is in an unsynchronized state, as opposed to deferring I/O until STP synchronization completes.

XRC_TEST
tells CP to timestamp I/O regardless of STP availability. This option is meant only for vendor and testing purposes, and can only be specified for systems running within a virtual machine.

MAXusers NOLimit
(the default) tells CP that the number of users who can log on at one time is not limited by the installation. Therefore, the system-defined limit of 99,999 logged on users is the maximum number allowed.

MAXusers *nnnnn*
defines the maximum number of users who can log on to the system at one time. The variable *nnnnn* is a decimal number from 1 through 99999.

PASSWORDS_ON_CMDs
tells CP whether to accept passwords in the command syntax (in clear text) when users issue the CP AUTOLOG, XAUTOLOG, LINK, or LOGON commands. If the setting is NO, CP accepts the command only without the password, then prompts the user for the password and masks the input field.

AUTOLog No
(the default) tells CP to not accept passwords entered by users who are issuing the CP AUTOLOG or XAUTOLOG command.

AUTOLog Yes
tells CP to accept passwords entered by users who are issuing the CP AUTOLOG or XAUTOLOG command. For AUTOLOG, users *must* enter the password as part of the command.

LINK No
(the default) tells CP to not accept passwords entered by users who are issuing the CP LINK command.

LINK Yes
tells CP to accept passwords entered by users who are issuing the CP LINK command.

LOGon No
(the default) tells CP to not accept passwords entered by users who are issuing the CP LOGON command.

LOGon Yes
tells CP to accept passwords entered by users who are issuing the CP LOGON command.

For more information, see AUTOLOG, XAUTOLOG, LINK, and LOGON in *z/VM: CP Commands and Utilities Reference*.

RETRieve
defines the default and maximum number of retrieve buffers allowed per user on your system. These numbers determine how many console input lines a user can retrieve. Before retrieving a buffer, users must define a program function (PF) key as a retrieve key, using the CP SET PF*nn* RETRIEVE command. After defining a retrieve key, users can press that PF key when they want to retrieve a command that they issued previously. For more information, see SET PFnn RETRIEVE in *z/VM: CP Commands and Utilities Reference*.

DEFault *nnn*
tells CP to define *nnn* default retrieve buffers per user on your system. The variable *nnn* is a decimal number from 0 to 255. If omitted, the default is 7.

Note: The number of default retrieve buffers must be less than or equal to the number of maximum retrieve buffers.

MAXimum *nnn*
tells CP to define *nnn* maximum retrieve buffers per user on your system. The variable *nnn* is a decimal number from 0 to 255. If omitted, the default is 7.

VDISK
VDSK
defines installation defaults for the system and user limits on the maximum amount of host storage available for allocation as virtual disks in storage. If an installation default is not defined, CP uses a built-in default.

You can supersede the installation or built-in default by using the CP SET VDISK command to set the current system limit or user limit. The DEFAULT operand on the SET VDISK command resets the current limit to the installation default, or, if none is defined, to the built-in default. The CP QUERY VDISK command displays current and default system and user limits. For more information, see QUERY VDISK and SET VDISK in *z/VM: CP Commands and Utilities Reference*.

Note: Use of virtual disks in storage increases the load on system paging, so you should set limits in proportion to the availability of paging space.

Syslim
sets the total resource available for allocating virtual disks in storage on the system.

If an installation default is not defined, CP calculates the built-in default from the available host storage. It takes 2050 non-pageable pages of page and segment data tables to support each gigabyte of address space for virtual disks in storage. The limit is set so these non-pageable structures can consume no more than 1/4 of the DPA pages. The limit is further reduced, if necessary, so the virtual storage pages used for address spaces for virtual disks in storage can consume no more than 1/4 of the total available paging space.

Userlim
sets the maximum resource available for virtual disks in storage created by a single user using the CP DEFINE command. This limit does not apply to virtual disks in storage defined by MDISK statements in the directory. If an installation default is not defined, the built-in default is 0.

Infinite
indicates all available host real storage may be allocated to virtual disks in storage up to the limit of 2147483648 512-byte blocks.

***nnnnnnnnnn* Blocks**
***nnnnnnnnnn* Blks**
specifies the number of 512-byte blocks of storage available for virtual disks in storage. If the number specified is equal to or greater than 2147483648, the limit is set to INFINITE.

***nnnnnnn*M**
specifies the number of megabytes of storage available for virtual disks in storage. If the number specified is equal to or greater than 1048576, the limit is set to INFINITE.

***nnnn*G**
specifies the number of gigabytes of storage available for virtual disks in storage. If the number specified is equal to or greater than 1024, the limit is set to INFINITE.

Usage Notes

1. ⚠ **Warning:** When enabling or disabling a new feature in the system configuration file, it is recommended to add a separate FEATURES statement on a new line rather than including the new option on an existing FEATURES statement. This will avoid issues with existing FEATURES statements when IPLing a CPLOAD module that does not include support for the new option. For more information, see "Adding New Operands to Existing Statements" on page 54.
2. If an IODF statement is defined in the system configuration file, then the hardware I/O configuration will be controlled by HCD. In this case, the following FEATURES statements are ignored if they are specified:

   - FEATURES ENABLE DYNAMIC_I/O
   - FEATURES ENABLE DYNAMIC_IO
   - FEATURES ENABLE SET_DYNAMIC_I/O
   - FEATURES ENABLE SET_DYNAMIC_IO
   - FEATURES DISABLE DYNAMIC_I/O
   - FEATURES DISABLE DYNAMIC_IO
   - FEATURES DISABLE SET_DYNAMIC_I/O
   - FEATURES DISABLE SET_DYNAMIC_IO

   If the *osconfig* parameter is specified on the IODF statement, then the software I/O configuration will be controlled by HCD. In this case, the following FEATURES statements are ignored if they are specified:

   - FEATURES ENABLE SET_DEVICES

- FEATURES DISABLE SET_DEVICES

For more information, see "IODF Statement" on page 181.

3. FEATURES ... SET_PRIVCLASS sets the parameters for the CP SET PRIVCLASS command, which general users can use to change their privilege classes to be all or a subset of the classes specified in their virtual machine definitions. For more information, see SET PRIVCLASS in *z/VM: CP Commands and Utilities Reference*.
4. At logon, CP gives each user enough storage for the default number of retrieve buffers. If users want to retrieve more than the default number of console input lines, they can use the CP SET RETRIEVE command to increase their number of retrieve buffers to some number less than or equal to the maximum number of retrieve buffers that you are defining for the system using this FEATURES statement. To find out the maximum number of retrieve buffers allowed on a running system, users can issue the CP QUERY RETRIEVE command. For more information, see QUERY RETRIEVE and SET RETRIEVE in *z/VM: CP Commands and Utilities Reference*.
5. If you enable AUTO_WARM_IPL and no warm start data has been saved, or if invalid warm start data is encountered, CP will only ask the operator whether it should try a FORCE start instead.
6. If you enable AUTO_WARM_IPL in the system configuration file and you want to go through the full series of prompts during the IPL, you can specify the PROMPT keyword as part of the IPL parameters on the Stand-Alone Program Loader (SAPL). For more information, see Passing IPL Parameters in *z/VM: System Operation*.
7. To change whether new devices are initialized when they are added to the system after IPL, use the CP SET NEW_DEVICES command. For more information, see SET NEW_DEVICES in *z/VM: System Operation*.
8. To allow users to dynamically change a device's I/O configuration after IPL, you must specify the ENABLE DYNAMIC_I/O operand on the FEATURES statement in the system configuration file or you must issue the CP SET DYNAMIC_I/O ON command after IPL. If you do not turn this function on either during or after IPL, CP will reject all attempts to dynamically change the I/O configuration. This means that CP rejects the following dynamic I/O commands:

```
     DEFINE CHPID          DEFINE CU            DEFINE DEVICE
     DELETE CHPID          DELETE CU            DELETE DEVICE
     MODIFY CHPID          MODIFY CU            MODIFY DEVICE

     DEFINE PATH           DEFINE CNTLUNIT      DEFINE IODEVICE
     DELETE PATH           DELETE CNTLUNIT      DELETE IODEVICE
     MODIFY PATH           MODIFY CNTLUNIT      MODIFY IODEVICE

     QUERY CONFIGMODE                           QUERY DYNAMIC_I/O
     SET CONFIGMODE        SET DEVICES          SET DYNAMIC_I/O
```

9. To change whether I/O operations from guest operating systems can be limited (or controlled) after IPL, use the CP SET DEVICES command. For more information, see SET DEVICES in *z/VM: CP Commands and Utilities Reference*.
10. Because the SYSLimit does not apply during system initialization, it is possible to find more LAN segments in the system than the number defined as the system limit.
11. Although the PASSWORDS_ON_CMDs operand can be set to prevent passwords from being entered with the commands (in clear text) on the terminal screen, passwords may still be included on the command line by REXX execs using the DIAG() and DIAGRC() functions. If you require access security without the possibility of passwords being stored in clear text, you should consider installing an external security manager (ESM).
12. In order to use the STP_Timestamping feature, STP must be set up on the Central Processor Complex (CPC), where the CPC is either a member of an STP-only Collaborative Timing Network (CTN) or a stratum 2 or higher member of a mixed CTN. In order to use the STP_TIMEZone / STP_TZ feature, STP must be set up on the CPC, and the CPC must be a member of an STP-only CTN.
13. The ENABLE and DISABLE options control attributes during IPL. This is defined as an IPL from the HMC. These attributes might not apply during CP bounce processing (that is, as initiated by the SHUTDOWN REIPL command, or restarting after an abend). For example, if you disable the

AUTO_WARM_IPL attribute, you will be prompted for the type of start you want to perform and given the opportunity to change the TOD clock, but only if you have IPLed from the HMC.

14. An ADJUNCT statement must exist in the user's directory entry (for example, ADJUNCT CMSADJ) to authorize that user to create an adjunct configuration. This statement must specify a valid template definition for the requested adjunct machine. The USER definition for the adjunct template must have the NOLOG option specified.
15. The FEATURES DISABLE/ENABLE ADJUNCTS setting can be overridden by the SET ADJUNCTS OFF/ON command.

Examples

1. To have CP:
   - Enable all log message support
   - Allow end users and the operator to issue the SET PRIVCLASS command
   - Clear all temporary disk space after users are through with it
   - Allow operators to dynamically change the I/O configuration
   - Prompt the operator for startup information after a software restart (bounce)
   - Allow operators to limit the I/O operations from guest operating systems
   - Initialize new devices when they are added to the system
   - Disable the automatic warm start feature
   - Give users a default of 7 retrieve buffers and let them go up to a maximum of 255 buffers
   - Allow an unlimited number of users to be logged on at one time
   - Prompt users for passwords on the CP AUTOLOG, XAUTOLOG, and LOGON commands, but not the CP LINK command
   - Set the installation default system limit on space for virtual disks in storage to '32GB' and the installation default user limit to 800 blocks

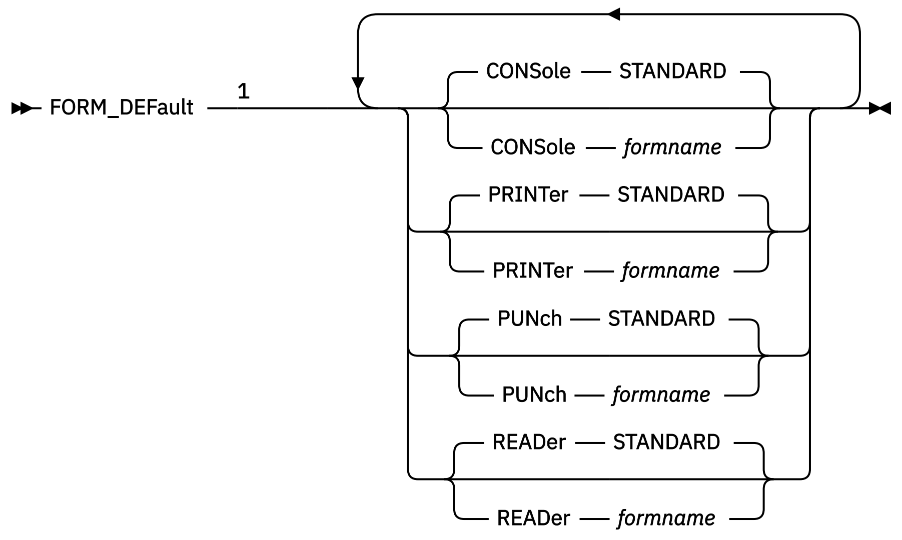
   use the following FEATURES statement:

```
/*----------------------- Features Statement ------------------------*/
Features,

Enable,

   Logmsg_From_File,        /* Allow log messages from files         */
        Show_ACCount Yes,      /* If account logmsg exists, show it  */
        Show_ACIgroup Yes,     /* If acigroup logmsg exists, show it */
        Show_Userid Yes,       /* If userid logmsg exists, show it   */

   Set_Privclass,           /* Let the SET PRIVCLASS command be
                               executed.  If this option is disabled,
                               then the SET PRIVCLASS command is  not
                               allowed.                              */

   Clear_Tdisk,             /* Clear all temporary disk space after
                               use                                   */

   Dynamic_I/O,             /* Allow dynamic I/O changes on this
                               system                                */

   Prompt_on_Bounce,        /* If CP bounces, prompt operator for
                               startup conditions                    */

   Throttling,              /* Allow people to limit I/O from guests */

   New_Devices_Initialized_when_Added, /* Initialize devices if we get
                                          an IPI I/O machine check   */

Disable,

   Auto_Warm_Ipl,           /* Do not perform Auto_Warm_IPL's, let
```

```
                                 operator see the CP start-up message
                                 prompts and time message              */
```

```
  /* Set some additional features ... */

  Retrieve,

     Default 7,      /* Give each user 7 default retrieve buffers      */
     Maximum 255,    /* Set the maximum system buffers per user to 255 */

  MaxUser NoLimit,      /* Don't set a maximum number of user limit    */

  Passwords_On_Cmds,

     AutoLog No,        /* Prompt user for password on AUTOLOG command */
     Link Yes,          /* Do not prompt for password on LINK command  */
     Logon No,          /* Prompt user for password on LOGON command   */

  Vdisk,

     Syslim 32G,        /* Set total virtual disks in storage to 32GB  */
     Userlim 800 Blks,  /* User can define max 800 blocks using DEFINE */

/*-------------------- End of Features Statement --------------------*/
```

FORM_DEFAULT Statement



Notes:

[1] You must specify at least one of the following operands.

Purpose

Use the FORM_DEFAULT statement to define default user form names for CP to use when it creates files on virtual printers, virtual punches, virtual consoles, or real card readers.

How to Specify

Include as many statements as needed; they are optional. You can place FORM_DEFAULT statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

CONSole *formname*
defines the default user (and, implicitly, operator) forms for virtual console spool files. If omitted, the default is STANDARD.

PRINTer *formname*
specifies the default user form for virtual printer spool files. If omitted, the default is STANDARD.

PUNch *formname*
specifies the default user form for virtual punch spool files. If omitted, the default is STANDARD.

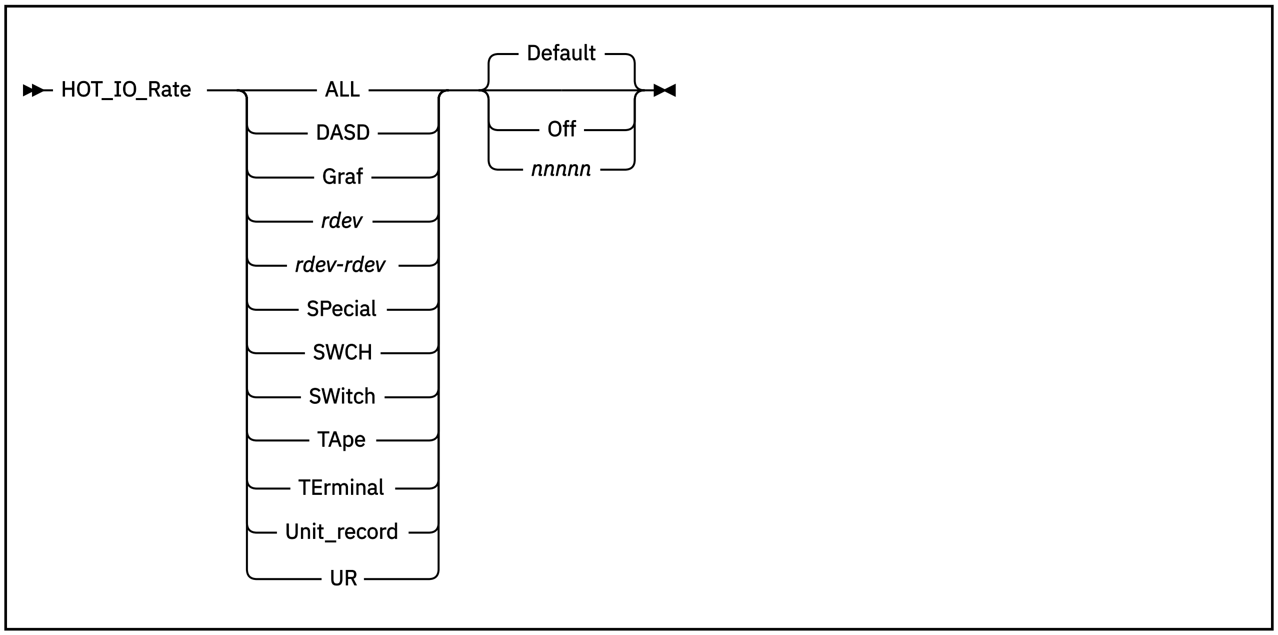
READer *formname*
specifies the default user (and, implicitly, operator) forms for files created on a real card reader. If omitted, the default is STANDARD.

Examples

1. To have CP use the form name STANDARD for the console, printer, punch, and reader user form names, use the following FORM_DEFAULT statement:

```
Form_Default,                       /* Set up user form defaults      */
   Console  STANDARD,               /*    so they are all set to      */
   Printer  STANDARD,               /*    the default ... STANDARD    */
   Punch    STANDARD,
   Reader   STANDARD
```

HOT_IO_RATE Statement



Purpose

Use the HOT_IO_RATE statement to define the maximum number of consecutive, unsolicited interrupts per second that CP should allow from an I/O device (the hot I/O rate) before refusing to accept input from it.

How to Specify

Include as many statements as needed; they are optional. You can place HOT_IO_RATE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

ALL
: tells CP to define the hot I/O rate for all devices in the active I/O configuration.

DASD
: tells CP to define the hot I/O rate for all DASDs in the active I/O configuration.

Graf
: tells CP to define the hot I/O rate for all graphic display devices in the active I/O configuration.

rdev
: tells CP to define the hot I/O rate for a device at a specific real device number in the active I/O configuration. The variable *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

rdev-rdev
: tells CP to define the hot I/O rate for devices in a range of specific real device numbers in the active I/O configuration. Each *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

SPecial
: tells CP to define the hot I/O rate for all special devices (3088s, CTCAs, 37x5s) in the active I/O configuration.

SWCH
SWitch
: tells CP to define the hot I/O rate for all switching devices in the active I/O configuration.

TApe
tells CP to define the hot I/O rate for all tape drive devices in the active I/O configuration.

TErminal
tells CP to define the hot I/O rate for all terminals in the active I/O configuration.

Unit_record
UR
tells CP to define the hot I/O rate for all unit record devices in the active I/O configuration.

Default
tells CP to use the default rate of 16 unsolicited interrupts per second for the specified device or devices.

Off
turns hot I/O detection off and tells CP to accept all unsolicited interrupts for the specified device or devices.

Attention: We do not recommend that you specify the OFF operand to turn off the hot I/O detection for a device or devices for any length of time. Hot I/O detection protects CP from broken hardware that floods the system with unsolicited interrupts. If you turn hot I/O detection off, you may experience performance degradation or a system abend.

nnnnn
tells CP how many unsolicited interrupts per second to accept for the specified device or devices before CP stops accepting input from the device or devices. The variable *nnnnn* is a decimal number from 1 to 62500.

Usage Notes

1. If an IODF statement is defined in the system configuration file with the *osconfig* parameter specified, then the software I/O configuration will be controlled by HCD. In this case, the HOT_IO_Rate statement is ignored if it is specified. For more information, see "IODF Statement" on page 181.
2. CP processes HOT_IO_RATE statements sequentially for each device before bringing it online. For example, if you specify:

   ```
       Hot_IO_Rate  DASD     100
       Hot_IO_Rate  200-300   50
   ```

   CP accepts 100 unsolicited interrupts per second for all DASD and 50 unsolicited interrupts per second for all devices with real device numbers between 200 and 300, inclusive. If there are any DASD defined in the 200-300 real device number range, those DASD will have an unsolicited interrupt rate of 50, not 100.
3. **Attention:** We do not recommend that you specify the OFF operand to turn off the hot I/O detection for a device or devices for any length of time. Hot I/O detection protects CP from broken hardware that floods the system with unsolicited interrupts. If you turn hot I/O detection off, you may experience performance degradation or a system abend.
4. If you need to change the hot I/O rate after IPL, use the CP SET HOTIO command. To check the current hot I/O rates, use the CP QUERY HOTIO command. For more information, see SET HOTIO and QUERY HOTIO in *z/VM: CP Commands and Utilities Reference*.

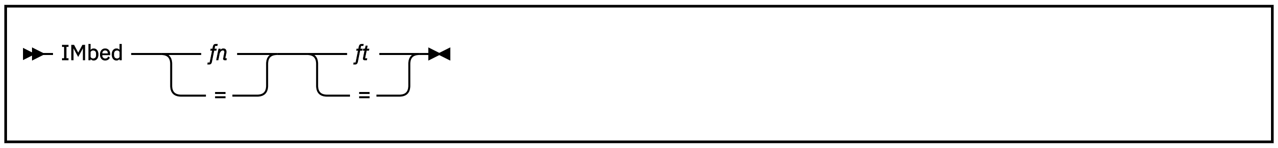
Examples

1. To specify a global unsolicited interrupt rate of 100 for all I/O devices, override the global rate for all DASDs, and override the global DASD rate for a specific range of DASD, use the following HOT_IO_RATE statements:

   ```
     Hot_IO_Rate  All        100        /* Set rate for all devices to
                                           make sure none affect the
                                           system too much.            */
   ```

```
   Hot_IO_Rate  DASD       200        /* Allow DASD a few more errors   */
   Hot_IO_Rate  0280-028f   50        /* Third party DASD string        */
```

IMBED Statement



Purpose

Use the IMBED statement to tell CP to imbed a file into the main system configuration file during an IPL. All imbedded files must reside on the same disk as the main system configuration file.

How to Specify

Include as many statements as needed; they are optional. You can place IMBED statements anywhere in the main system configuration file and in any files that are imbedded into the main system configuration file. You can nest imbedded files as deep as you like.

Operands

fn
is the file name of the file you want imbedded. An = tells CP to imbed a file with the same file name.

ft
is the file type of the file you want imbedded. An = tells CP to imbed a file with the same file type.

Usage Notes

1. If you specify a file name or file type of -SYSTEM-, CP uses the name of the system being IPLed as that file name or file type. For example, if your system name is VMSYS3 and you have the following IMBED statement in your system configuration file:

   ```
   Imbed -system- config
   ```

   CP looks for a file named VMSYS3 CONFIG on the parm disk. If found, CP imbeds the file and processes the statements within it. If CP does not find the file, the IMBED statement fails and CP continues processing with the statement after the IMBED.
2. An imbed will not succeed if the -SYSTEM- variable is included and no system name exists to replace it. This situation arises when you do not specify SYSTEM_IDENTIFIER and SYSTEM_IDENTIFIER_DEFAULT statements in the system configuration file before the IMBED statement.
3. This statement will not accept imbed loops. If the system configuration file includes an imbed of the file SAMPLE CONFIG, and the SAMPLE CONFIG file includes an imbed of the file SAMPLE2 CONFIG, then CP will generate an error message if the SAMPLE2 CONFIG file tries to imbed the SAMPLE CONFIG file.

Examples

1. If you list your RDEVICE statements in a file separate from the master system configuration file, you can use the following IMBED statement to include the RDEVICE statements:

   ```
   Imbed  rdev  config                    /* Pull in RDEVICE statements */
   ```
2. If you have several systems with similar system configuration files, you can use the IMBED statement to create a master system configuration file that contains the common statements and imbed files

that contain the system-specific statements. For example, suppose you had three systems (BOSTON, MAINE, and NEWYORK) that had very similar system configuration files with the following exceptions:

- BOSTON lets end users change their privilege classes
- MAINE wants to manually answer all the prompts during an IPL
- NEWYORK limits the number of logged on users to 1000.

You can create a main system configuration file with the following IMBED statement:

```
Imbed  -system- config                /* Pull in the system-specific
                                         information                */
```

Each of the three systems has a copy of this main system configuration file and each has a system-specific configuration file. For example, on the BOSTON system, there is a file called BOSTON CONFIG which contains the following statement:

```
Features  Enable  Set_PrivClass      /* Let the users and the system
                                        operator change command
                                        privilege classes           */
```

On the MAINE system, there is a file called MAINE CONFIG which contains the following statement:

```
Features  Disable  Auto_Warm_IPL           /* Perform manual IPLs */
```

And on the NEWYORK system, there is a file called NEWYORK CONFIG which contains the following statement:

```
Features  MaxUsers  1000                /* Do not let more than
                                           1000 users log on    */
```

Note: You can also process system-specific information without using the IMBED statement. You can use the EQUATE statement to set up nicknames for systems and use those nicknames in the main system configuration file to limit the scope of the statements they preface. For more information, see "EQUATE Statement" on page 155.

In the previous example, you could have defined nicknames for the three systems and used those nicknames to preface the FEATURES statements:

```
 .
 .
 .
Equate  Marketing   boston
Equate  University  maine
Equate  Research    newyork
 .
 .
 .
Marketing:   Features  Enable  Set_PrivClass
University:  Features  Disable  Auto_Warm_IPL
Research:    Features  MaxUsers  1000
 .
 .
 .
```

INIT_MITIME Statement

```
►►─ INIT_MITime ── ss ─►◄
```

Purpose

Use the INIT_MITIME statement to change the MITIME (the time interval at which a device is checked for missing interrupts) that is in effect during device initialization at system IPL time.

How to Specify

The INIT_MITIME statement is optional. Since there is only one MITIME value used at IPL time, there should never be a need for more than one INIT_MITIME statement. However, if you specify more than one INIT_MITIME statement, CP uses the value from the last statement. If no INIT_MITIME statement is specified, CP uses the default value of 15 seconds.

Operands

ss
is the time interval in seconds at which the device should be examined for missing interrupts. The number of seconds specified must be a value from one to 60, and the value is rounded up to the next multiple of five seconds.

Usage Notes

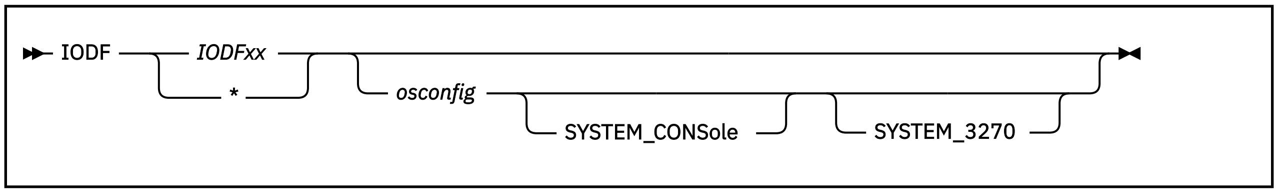
1. Lowering the initialization MITIME value below the default value of 15 seconds may cause certain devices to not come online during IPL due to timeout errors. These devices may be able to be brought online with the VARY ON command after the IPL has completed.
2. When using the INIT_MITIME statement to raise the MITIME value, take into consideration the fact that this could result in a slower IPL due to CP waiting longer for I/O interrupts to come in from devices.

Examples

1. To set the initialization MITIME to 45 seconds, use the following INIT_MITIME statement:

```
Init_Mitime  45
```

IODF Statement



Purpose

Use the IODF statement to indicate that HCD will be used to control the I/O hardware and/or software configuration.

How to Specify

The IODF statement is optional. If specified, it must contain at least the IODF*xx* operand to indicate the filename of the production IODF from which configuration information will be read.

You can place the IODF statement anywhere in the system configuration file. Only the first IODF statement that is found in the system configuration file is honored. If additional IODF statements are specified, they are ignored.

Operands

IODF*xx*
: specifies the filename of the production IODF to be used. IODF is a constant value and *xx* is a variable that consists of two hexadecimal numbers. The filetype of the specified production IODF is not specified in the statement, but must be PRODIODF. This file must be located on the PMAINT CF0 disk (the same disk where the SYSTEM CONFIG file resides) at system IPL time.

*****
: specifies that the filename of the production IODF to be used is the one currently stored in the hardware configuration token in the HSA. The filename must be in the form of IODF*xx* and the filetype that will be used with that filename is PRODIODF. This file must be located on the PMAINT CF0 disk (the same disk where the SYSTEM CONFIG file resides) at system IPL time.

  Note: In order for this to be used, the hardware configuration token must contain the name of the production IODF file (in the form of IODF*xx*) in the descriptor field 2 portion of the token.

osconfig
: specifies the OS configuration ID of the VM I/O configuration that is defined in the IODF. If specified, this configuration is used to build the software view of the I/O configuration, and the statements that are normally used in the system configuration file to build this view are ignored.

SYSTEM_CONSole
: specifies that the Operating System Messages panel on the IBM Hardware Management Console (HMC) can serve as a system operator console. This can only be specified when an OS configuration ID is specified.

SYSTEM_3270
: specifies that the integrated 3270 console on the HMC can serve as a system operator console. This can only be specified when an OS configuration ID is specified.

Usage Notes

1. If an error occurs during the processing of an IODF statement which would result in the system coming up without the I/O configuration being defined as expected, then CP will enter a disabled wait state. This wait state could occur for any of the following reasons:

   the specified IODF name is not valid

the IODF could not be opened
the IODF could not be read
the data in the IODF was not valid
an *osconfig* name was specified, but no matching OSR record was found in the IODF

2. Once an IODF statement has been processed in the system configuration file, HCD is in control of the hardware and/or software I/O configuration. Because of this, any system configuration file statements that affect the I/O configuration are nullified. Any of these statements occurring prior to the IODF statement (meaning they have already been processed) are undone, and any of these statements which occur after the IODF statement are ignored. The following list shows the I/O configuration statements that are always undone or ignored, as well as which I/O configuration statements are only undone or ignored when HCD is controlling the software configuration (i.e. when an *osconfig* name has been specified):

   - IODF
   - FEATURES DISABLE/ENABLE DYNAMIC_IO
   - FEATURES DISABLE/ENABLE SET_DYNAMIC_IO
   - RDEVICE (if osconfig name specified)
   - DEVICES ACCEPTED/NOTACCEPTED (if osconfig name specified)
   - DEVICES DYNAMIC_I/O/NOTDYNAMICI/O (if osconfig name specified)
   - DEVICES OFFLINE_AT_IPL/ONLINE_AT_IPL (if osconfig name specified)
   - DEVICES SENSED/NOTSENSED (if osconfig name specified)
   - DEVICES SHARED/NOTSHARED (if osconfig name specified)
   - HOT_IO_RATE (if osconfig name specified)
   - OPERATOR_CONSOLES (if osconfig name specified)
   - EMERGENCY_MESSAGE_CONSOLES (if osconfig name specified)
   - FEATURES DISABLE/ENABLE SET_DEVICES (if osconfig name specified)

3. Once a system is IPLed with HCD in control of the I/O configuration, there is a one-way mechanism to take that control away from HCD and give it back to VM. The DISABLE HCD command provides this one-way escape from having HCD control the I/O configuration. It will shut down HCD's capabilities for the rest of the current IPL, and will attempt to give VM the capability to dynamically change the I/O configuration through its dynamic I/O command interface.

Examples

1. To have HCD control only the hardware I/O configuration, as specified in the IODF01 PRODIODF production IODF file, use the following IODF statement in the system configuration file:

```
    IODF IODF01
```

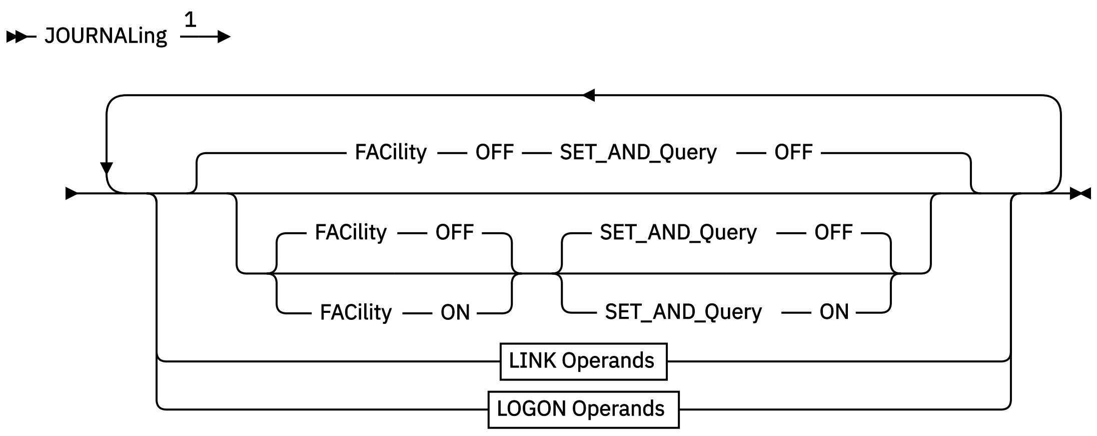
2. To have HCD control both the hardware and software I/O configuration, as specified in the IODF02 PRODIODF file (which contains the CONFIG04 operating system configuration), use the following IODF statement in the system configuration file:

```
    IODF IODF02 CONFIG04
```

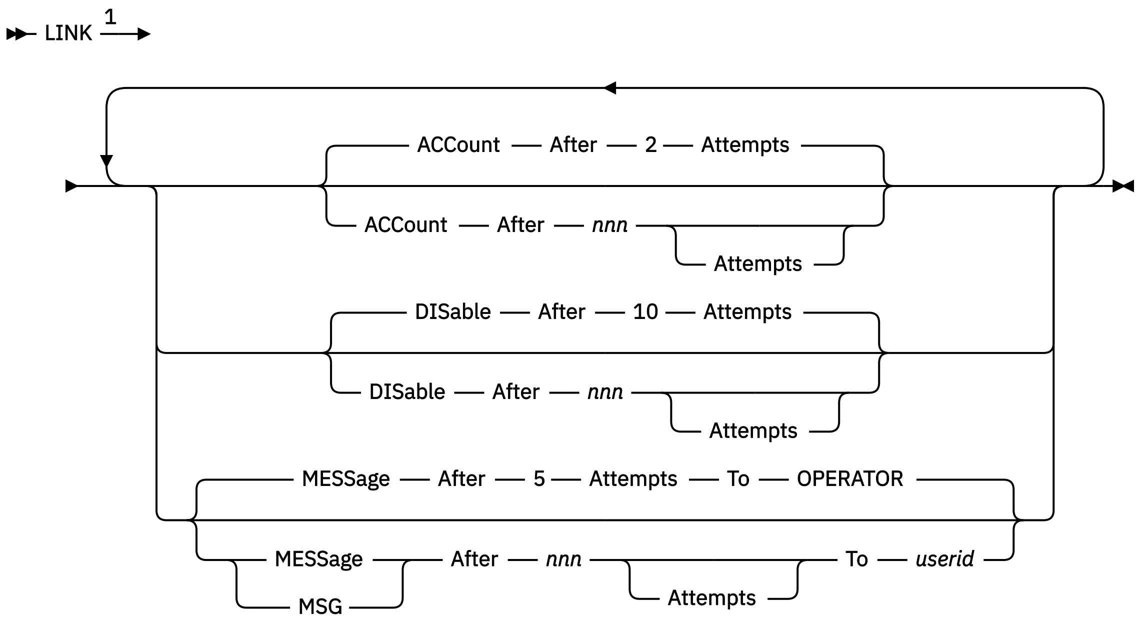
3. To have HCD control both the hardware and software I/O configuration, as specified in the IODF02 PRODIODF file (which contains the CONFIG04 operating system configuration), and to have the system console as a possible operator console, use the following IODF statement in the system configuration file:

```
    IODF IODF02 CONFIG04 SYSTEM_CONSOLE
```

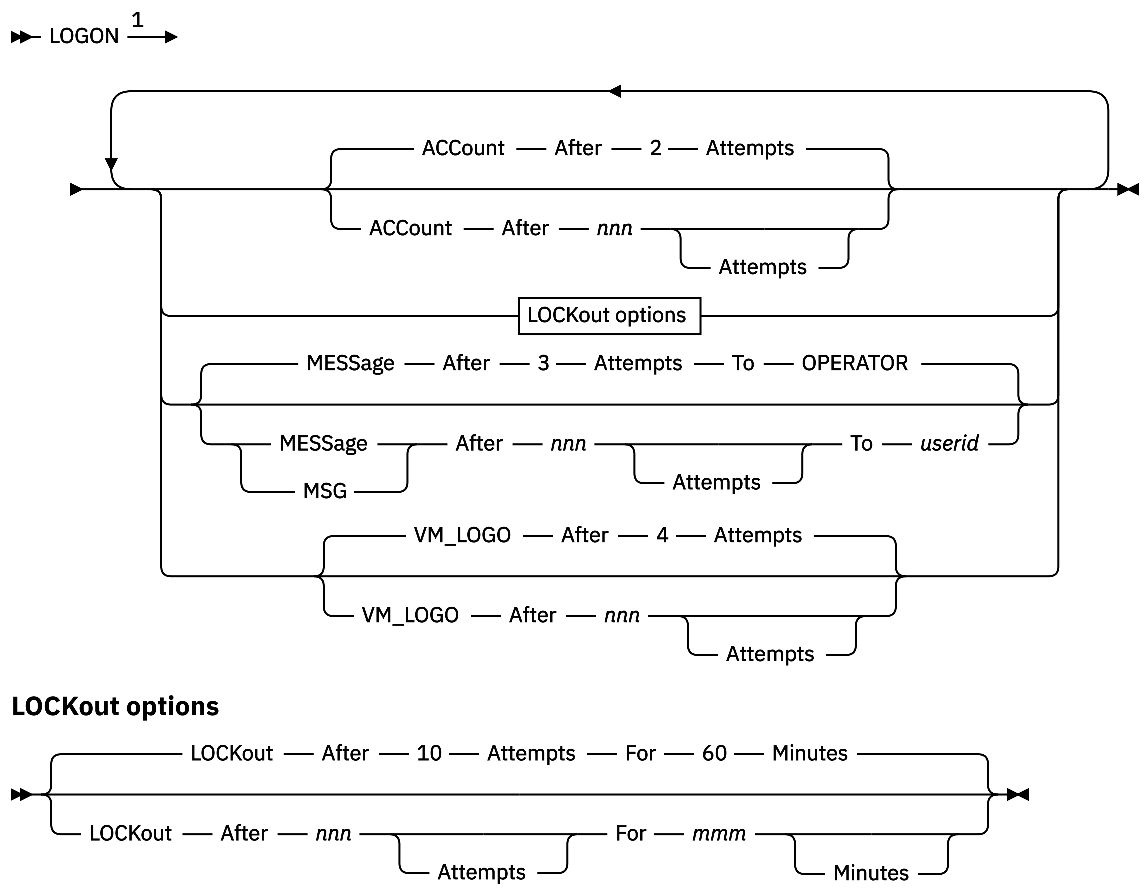
JOURNALING Statement



LINK Operands



LOGON Operands



Notes:

[1] You must specify at least one operand.

Purpose

Use the JOURNALING statement to tell CP whether to include the journaling facility, whether to enable the system being initialized to set and query the journaling facility, and what to do if someone tries to log on to the system or link to a disk without a valid password.

How to Specify

Include as many statements as needed; they are optional. You can place JOURNALING statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

FACility OFF
tells CP to disable the journaling facility for this system. This is the initial setting.

FACility ON
tells CP to enable the journaling facility for this system.

SET_AND_Query OFF
tells CP to prevent people from using the CP SET and QUERY commands to set and query the journaling function for this system.

SET_AND_Query ON
tells CP to allow people to use the CP SET and QUERY commands to set and query the journaling function for this system.

LINK

tells CP that you want to define what should happen when users try to link to a minidisk with the wrong password. These settings only apply to a single user ID for a single logon session.

ACCount After *nnn* Attempts

tells CP to generate a type 06 accounting record after someone tries *nnn* times to use a CP LINK command with an invalid password and to generate a type 06 accounting record each time that person issues a subsequent CP LINK command with an invalid password. The variable *nnn* is a decimal number from 0 to 255. If omitted, the default is ACCOUNT AFTER 2 ATTEMPTS.

DISable After *nnn* Attempts

tells CP to disable the CP LINK command for a user that tries *nnn* times to use the CP LINK command with an invalid password. The variable *nnn* is a decimal number from 1 to 255. If omitted, the default is DISABLE AFTER 10 ATTEMPTS.

MESSage After *nnn* Attempts To *userid*
MSG After *nnn* Attempts To *userid*

tells CP to send a message to a specific user ID when a user tries *nnn* times to use the CP LINK command with an invalid password. If the specified *userid* is disconnected or logged off, CP sends the message to the system operator. The variable *nnn* is a decimal number from 0 to 255. If omitted, the default is MESSAGE AFTER 5 TO OPERATOR.

LOGON

tells CP that you want to define what should happen when users try to log on with the wrong password.

ACCount After *nnn* Attempts

tells CP to generate a type 04 accounting record after someone tries *nnn* times to log on with an invalid password and to generate a type 04 accounting record each time that person issues a subsequent CP AUTOLOG, LOGON, or XAUTOLOG command with an invalid password. The variable *nnn* is a decimal number from 0 to 255. If omitted, the default is ACCOUNT AFTER 2 ATTEMPTS.

LOCKout After *nnn* Attempts For *mmm* Minutes

tells CP to issue an error message to the terminal, lock the terminal, and lock the user ID after someone tries (unsuccessfully) more than *nnn* times to log on to a specific user ID or terminal. No one can log on to that user ID or use that terminal for *mmm* minutes; anyone who tries receives another error message. The variable *nnn* is a decimal number from 1 to 255 and the variable *mmm* is a decimal number from 0 to 255. If omitted, the default is LOCKOUT AFTER 10 ATTEMPTS FOR 60 MINUTES.

MESSage After *nnn* Attempts To *userid*
MSG After *nnn* Attempts To *userid*

tells CP to send a message to a specific user ID when a user tries more than *nnn* times to log on to a user ID with an invalid password. If the specified *userid* is disconnected or logged off, CP sends the message to the system operator. The variable *nnn* is a decimal number from 0 to 255. If omitted, the default is MESSAGE AFTER 3 ATTEMPTS TO OPERATOR.

VM_LOGO After *nnn* Attempts

tells CP to display a new VM logon screen and allow a new logon sequence to be started after someone tries more than *nnn* times to log on with an invalid password from a single terminal. The variable *nnn* is a decimal number from 1 to 255. If omitted, the default is VM_LOGO AFTER 4 ATTEMPTS.

Examples

1. To have CP:
   - Enable the journaling facility
   - Allow the QUERY and SET journaling commands to be issued
   - Send a message to the operator after someone tries 3 times to log on with the wrong password and after someone tries 5 times to link to a minidisk with the wrong password

- Generate a type 04 accounting record after someone tries 5 times to log on with the wrong password (and to generate a type 04 accounting record for each subsequent attempt)
- Generate a type 06 accounting record after someone tries 6 times to link to a minidisk with the wrong password
- Display a new logo screen on a terminal after someone tries 7 times to logon with the wrong password
- Lock a terminal and user ID for 10 minutes after someone tries 9 times to log on with the wrong password
- Disable the LINK command for a specific user ID for the rest of their logon session when they try to link to a minidisk 8 times with the wrong password

use the following JOURNALING statement:

```
Journaling,              /* Set Up Journaling Facility               */
   Facility       on,    /*    Turn On Journaling                    */
   Set_and_Query  on,    /*    Allow Set & Query Journaling Commands */
   Logon,                /*    Set Up Logon Journaling Values        */
      Message  after  3  attempts  to  operator,
      Account  after  5  attempts,
      VM_Logo  after  7  attempts,
      Lockout  after  9  attempts  for  10,
   Link,                 /*    Set Up Link Journaling Values         */
      Message  after  5  attempts  to  operator,
      Account  after  6  attempts,
      Disable  after  8  attempts
```

LOGO_CONFIG Statement

```
▸▸— LOGO_CONFig — fn — ft —▸◂
```

Purpose

Use the LOGO_CONFIG statement to specify the file name and file type of a logo configuration file.

How to Specify

The LOGO_CONFIG statement is optional. Because there is only one logo configuration file, you need only one LOGO_CONFIG statement. If you specify more than one LOGO_CONFIG statement, CP uses the last one specified. You can place LOGO_CONFIG statements anywhere in the system configuration file.

If you do not specify a LOGO_CONFIG statement, CP looks for a file named LOGO CONFIG on the parm disk. If this file does not exist on the parm disk, CP uses the information in HCPBOX ASSEMBLE to build the logos.

Operands

fn
is the file name of the logo configuration file.

ft
is the file type of the logo configuration file. This file must be on the same disk as the system configuration file.

Usage Notes

1. If you specify a file name or file type of -SYSTEM-, CP uses the name of the system being IPLed as that file name or file type. For example, if your system name is VMSYS3 and you have the following LOGO_CONFIG statement in your system configuration file:

   ```
   Logo_Config  -system-  config
   ```

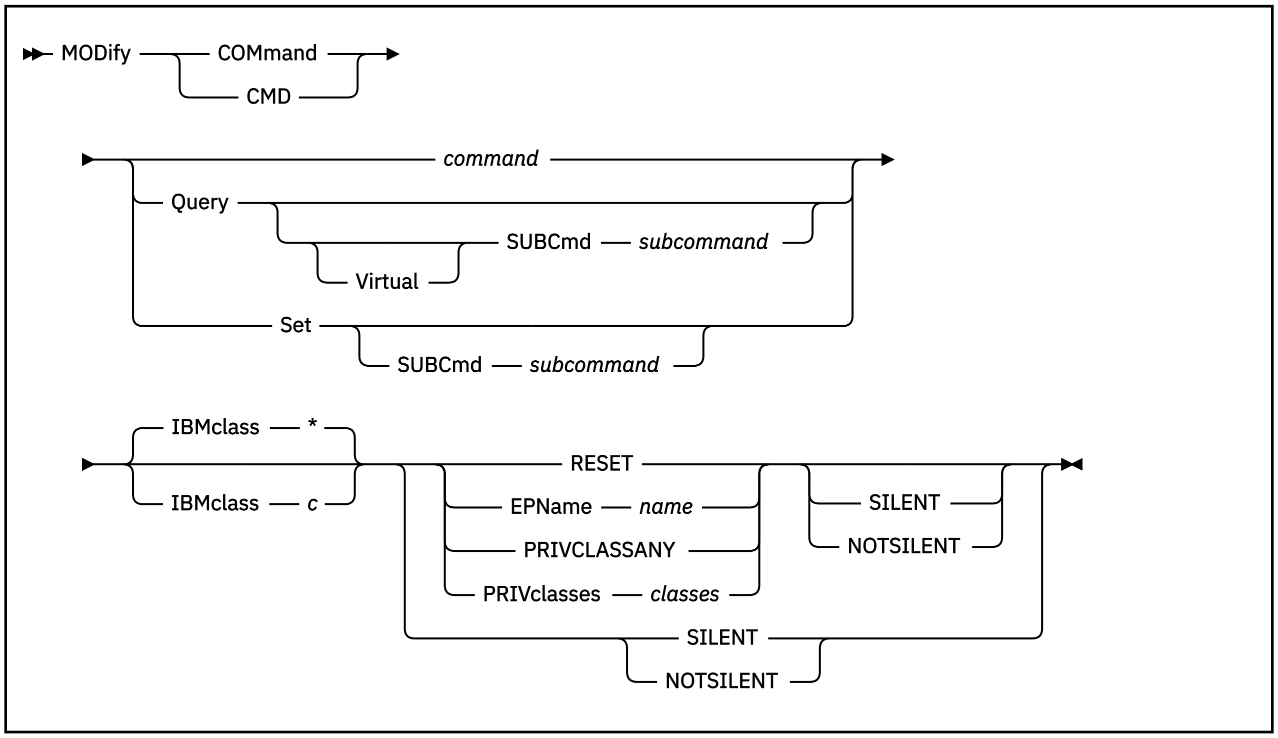
   CP looks for a file named VMSYS3 CONFIG on the parm disk.
2. CP will not read a logo configuration file if the -SYSTEM- variable is included and no system name exists to replace it. This situation arises when you do not specify SYSTEM_IDENTIFIER and SYSTEM_IDENTIFIER_DEFAULT statements in the system configuration file before the LOGO_CONFIG statement.

Examples

1. To have CP read a logo configuration file called LOGO CONFIG during IPL, use the following LOGO_CONFIG statement:

   ```
   Logo_Config  logo  config
   ```

MODIFY COMMAND / CMD Statement



Purpose

Use the MODIFY COMMAND or CMD statement to redefine an existing CP command on the system during initialization. For more information about specifying generic command names, see Usage Note "1" on page 190.

You can also redefine an existing CP command after initialization using the MODIFY COMMAND or MODIFY CMD commands. For more information, see MODIFY COMMAND / CMD in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place MODIFY COMMAND or CMD statements anywhere in the system configuration file. If you specify more than 1 statement with the same command or subcommand name, CP uses only the first statement. Subsequent MODIFY COMMAND or CMD statements do not redefine the command.

Operands

command

is the name of the existing CP command that you are overriding. The variable *command* is a 1-character to 12-character alphanumeric string.

Query

tells CP that you are overriding a CP QUERY command.

Virtual

tells CP that you are overriding a CP QUERY VIRTUAL subcommand.

SUBCmd *subcommand*

is the name of the CP QUERY subcommand that you are overriding. The variable *subcommand* is a 1-character to 12-character alphanumeric string. For more information about specifying generic subcommand names, see Usage Note "1" on page 190.

Set
tells CP that you are overriding a CP SET command.

SUBCmd *subcommand*
is the name of the CP SET subcommand that you are overriding. The variable *subcommand* is a 1-character to 12-character alphanumeric string. For more information about specifying generic subcommand names, see Usage Note "1" on page 190.

IBMclass *
tells CP to redefine all versions of the specified command or subcommand. If omitted, IBMCLASS * is the default.

IBMclass *c*
tells CP to redefine a specific version of the specified command or subcommand. The variable *c* can be any 1 of the following:

A
this is a system-control command to be used by the primary system operator.

B
this is a command for operational control of real devices.

C
this is a command to alter host storage.

D
this is a command for system-wide control of spool files.

E
this is a command to examine host storage.

F
this is a command for service control of real devices.

G
this is a general-use command used to control the functions of a virtual machine.

0
(zero) this command has no specific IBM class assigned.

RESET
tells CP to stop using the customer-written CP command and return to using the existing CP command that was shipped with the z/VM product.

EPName *name*
tells CP the name of the entry point that contains the code to process the command. The variable *name* must be a 1-character to 8-character string. The first character must be alphabetic or one of the following special characters: dollar sign ($), number sign (#), underscore (_), or at sign (@). The rest of the string can be alphanumeric characters, the four special characters ($, #, _, and @), or any combination thereof.

PRIVCLASSANY
tells CP that users with any privilege class can issue the command that you are redefining.

PRIVclasses *classes*
tells CP that only users with 1 or more of the specified privilege classes can issue the command that you are redefining. Whatever you specify on this operand will replace the current privilege classes. The variable *classes* is 1 or more privilege classes in the range A through Z, 1 through 6, or an asterisk (*). Privilege class * indicates all privilege classes (A-Z and 1-6).

Note: If you want more than one privilege class, specify your classes in one string of characters. Do not separate the classes with blank spaces. For example, specify "`privclasses abc123`", not "`privclasses a b c 1 2 3`".

SILENT
tells CP that the responses from the command you are redefining can be suppressed by invoking it using the SILENTLY command. For more information, see SILENTLY in *z/VM: CP Commands and Utilities Reference*.

Note: Response suppression is supported only for the ATTACH, DETACH, and GIVE commands.

NOTSILENT
tells CP that the responses from the command you are redefining cannot be suppressed.

Usage Notes

1. If you are making similar changes to several QUERY (or SET) subcommands that share common characters, you can use a generic name rather than issuing multiple MODIFY COMMAND or CMD commands. A generic name is a 1- to 12-character alphanumeric string with asterisks (*) in place of one or more characters and percent signs (%) in place of exactly one character. For example:

   ```
   modify command query tr*c% …
   ```

   redefines all QUERY commands that start with TR and have C as their next-to-last character.
2. To define a new CP command or a new version (by IBM class) of an existing CP command, use the DEFINE COMMAND or CMD statement or command. For more information, see "DEFINE COMMAND / CMD Statement" on page 90. See also DEFINE COMMAND / CMD command in *z/VM: CP Commands and Utilities Reference*.
3. To define a new alias for an existing CP command, use the DEFINE ALIAS statement or command. For more information, see "DEFINE ALIAS Statement" on page 87. See also DEFINE ALIAS in *z/VM: CP Commands and Utilities Reference*.
4. Once defined, COMMANDS, SUBCOMMANDS, ALIASES, and DIAGNOSE codes cannot be deleted. They can be altered in various appropriate ways but they remain in existence until a SHUTDOWN or RESTART IPL is done.
5. To load the command processing code into the system execution space, use the CPXLOAD statement or command. For more information, see "CPXLOAD Statement" on page 76. See also CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
6. To activate a CP command:
   - To activate a command while defining the command, use the ENABLE operand of the DEFINE COMMAND or CMD statement or command. For more information, see "DEFINE COMMAND / CMD Statement" on page 90. See also DEFINE COMMAND / CMD command in *z/VM: CP Commands and Utilities Reference*.
   - To activate a command after defining the command, use the ENABLE COMMAND or CMD statement or command. For more information, see "ENABLE COMMAND / CMD Statement" on page 146. See also ENABLE COMMAND / CMD in *z/VM: CP Commands and Utilities Reference*.
7. To display the address of the CP command table entry block, the current IBM class, and the current privilege class for a specified CP command, use the LOCATE CMDBK command. For more information, see LOCATE CMDBK in *z/VM: CP Commands and Utilities Reference*.
8. To deactivate a CP command:
   - To deactivate a CP command while defining it, use the DISABLE operand of the DEFINE COMMAND or CMD statement or command. For more information, see "DEFINE COMMAND / CMD Statement" on page 90. See also DEFINE COMMAND / CMD command in *z/VM: CP Commands and Utilities Reference*.
   - To deactivate a command after defining the command, use the DISABLE COMMAND or CMD statement or command. For more information, see "DISABLE COMMAND / CMD Statement" on page 128. See also DISABLE COMMAND / CMD in *z/VM: CP Commands and Utilities Reference*.
9. To remove the command processing code from the system execution space, use the CPXUNLOAD command. For more information, see CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.
10. For more information about enabling and disabling commands, see Defining and Modifying Commands and Diagnose Codes in *z/VM: CP Exit Customization*.

Examples

1. To have CP change the CP SHUTDOWN command from privilege class A to privilege class S, use the following:

```
Modify Command shutdown,           /* Put tighter controls on who can */
              privclasses s        /* shut down the system.           */
```

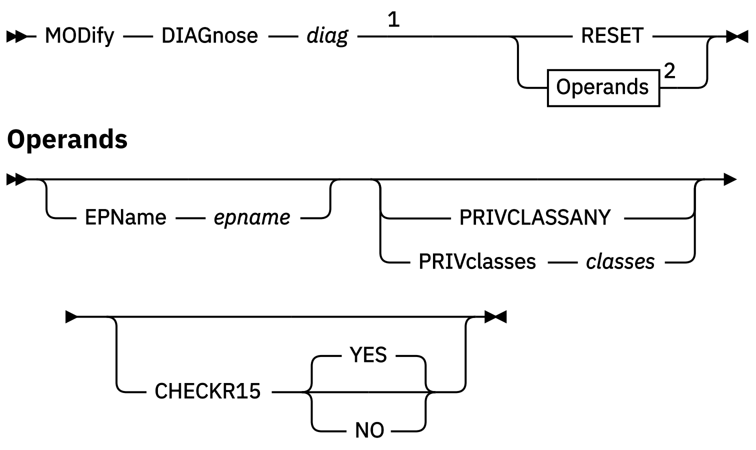
2. To have CP change the version of the CP SET PRIVCLASS command that allows users with any privilege class to issue the command to a version that allows only users with privilege class G and Z to issue the command, use the following:

```
Modify Command Set SubCmd privclass,    /* Make sure only a few can */
                 IBMclass 0,            /* issue SET PRIVCLASS      */
                 PrivClasses gz
```

3. To have CP change all IBM class E QUERY subcommands to privilege class G, use the following:

```
Modify Command Query SubCmd *,          /* Make sure anyone can use */
                 IBMclass e,            /* the Queries to check     */
                 PrivClasses g          /* system storage           */
```

MODIFY DIAGNOSE Statement



Notes:

[1] You must specify one of the following operands.

[2] If you specify more than one operand, you can specify them in any order.

Purpose

Use the MODIFY DIAGNOSE statement to redefine an existing DIAGNOSE code on the system during initialization.

You can also redefine an existing DIAGNOSE code after initialization using the MODIFY DIAGNOSE command. For more information, see MODIFY DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place MODIFY DIAGNOSE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

diag
is the number of the DIAGNOSE code that you are redefining. The variable *diag* must be a hexadecimal number between X'0100' and X'01FC' and must be a multiple of 4. All other DIAGNOSE code numbers are reserved for IBM use only.

RESET
tells CP to stop using the customer-written DIAGNOSE code and return to using the existing DIAGNOSE code that was shipped with the z/VM product.

EPName *name*
tells CP the name of the entry point that contains the code to process the DIAGNOSE code. The variable *name* must be a 1- to 8-character string. The first character must be alphabetic or one of the following special characters: dollar sign ($), number sign (#), underscore (_), or at sign (@). The rest of the string can be alphanumeric characters, the 4 special characters ($, #, _, and @), or any combination thereof.

PRIVCLASSANY
tells CP that users with any privilege class can issue the DIAGNOSE code that you are redefining.

PRIVclasses *classes*
tells CP that only users with 1 or more of the specified privilege classes can issue the DIAGNOSE code that you are redefining. Whatever you specify on this operand will replace the current privilege classes. The variable *classes* is 1 or more privilege classes in the range A through Z, 1 through 6, or an asterisk (*). Privilege class * indicates all privilege classes (A-Z and 1-6).

Note: If you want more than one privilege class, specify your classes in one string of characters. Do not separate the classes with blank spaces. For example, specify "`privclasses abc123`", not "`privclasses a b c 1 2 3`".

CHECKR15 YES
CHECKR15 NO
indicates whether the diagnose router should check register 15 upon return from the diagnose handler.

Usage Notes

1. To define a new DIAGNOSE code, use the DEFINE DIAGNOSE statement or CP command. For more information, see "DEFINE DIAGNOSE Statement" on page 96. See also DEFINE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.

   Note: Unless you specify the ENABLE operand of the DEFINE DIAGNOSE statement or command, the new DIAGNOSE code is initially in a disabled state after being defined.
2. To load the DIAGNOSE processing code into the system execution space, use the CPXLOAD statement or CPXLOAD CP command. For more information, see "CPXLOAD Statement" on page 76. See also CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
3. If you do not specify the ENABLE operand, a new DIAGNOSE code is initially in a disabled state after being defined. CP treats disabled DIAGNOSE codes as if they were never defined. If you try to use a disabled DIAGNOSE code in a program, CP will give you a program check specification exception.
4. To activate a DIAGNOSE code while defining it, use the ENABLE operand of the DEFINE DIAGNOSE statement or command. For more information, see "DEFINE DIAGNOSE Statement" on page 96. See also DEFINE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
5. To activate a new DIAGNOSE code after defining it, use the ENABLE DIAGNOSE statement or ENABLE DIAGNOSE CP command. For more information, see "ENABLE DIAGNOSE Statement" on page 148. See also ENABLE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
6. To display information about a DIAGNOSE code (status, entry point name, and privilege class) after initialization, use the QUERY DIAGNOSE command. For more information, see QUERY DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
7. To display the address of the CP DIAGNOSE code table block for a DIAGNOSE code after initialization, use the LOCATE DGNBK command. For more information, see LOCATE DGNBK in *z/VM: CP Commands and Utilities Reference*.
8. To deactivate a DIAGNOSE code after defining it, use the DISABLE DIAGNOSE statement or command. For more information, see "DISABLE DIAGNOSE Statement" on page 130. See also DISABLE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
9. To deactivate a DIAGNOSE code while defining it, use the DISABLE operand of the DEFINE DIAGNOSE statement or command. For more information, see "DEFINE DIAGNOSE Statement" on page 96. See also DEFINE DIAGNOSE in *z/VM: CP Commands and Utilities Reference*.
10. To remove the DIAGNOSE processing code from the system execution space, use the CPXUNLOAD command. For more information, see CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.
11. Many external security managers (ESMs) do not support DIAGNOSE codes above X'03FC'. For this reason, CP does not support DIAGNOSE codes above X'03FC'. The DIAGNOSE codes between X'0000' and X'03FC' are divided as follows:

    X'0000' to X'00FC'
    Reserved for IBM use

X'0100' to X'01FC'
Reserved for customer use

X'0200' to X'03FC'
Reserved for IBM use.

12. When CHECKR15 YES is specified, the diagnose router will check register 15 on return from the diagnose handler. If register 15 contains:

    RC = 0
    Processing was successful. Complete the guest instruction.

    RC = 4
    Processing failed due to a condition which would cause a guest program check. Simulate guest program interruption passed in R0.

    RC = 8
    Nullify the instruction.

    RC = 12
    Present the machine check then nullify the instruction. R2 will contain the address of the MCRBK which will contain the machine check information.

    RC = 16
    Generate machine check for processing damage, then go to HCPENDOP to terminate the instruction.

    RC = 20
    Present the machine check, then go to HCPENDOP to terminate the instruction. R2 will contain the address of the MCRBK, which contains machine check information.

    RC = 24
    Issue error message or soft abend for paging I/O error, then nullify the instruction. R1 has the message or abend number.

    If a return code is invalid (negative, not a multiple of 4 or too big (RC > 24)), then a soft abend will occur.

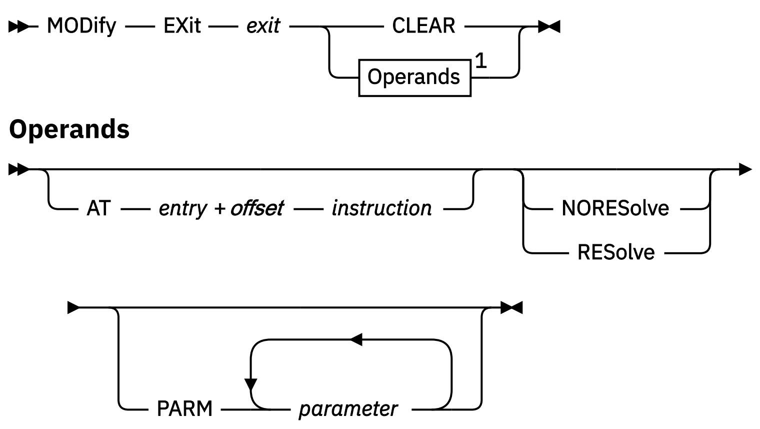
13. For more information about user-defined DIAGNOSE codes, see *z/VM: CP Exit Customization*.

Examples

1. To have CP change the privilege class for DIAGNOSE code X'7C' from "any" to R, use the following:

```
Modify Diagnose 7c PrivClasses r
```

MODIFY EXIT Statement



Notes:

[1] You must specify at least one of the listed operands. If you specify more than one, you can specify them in any order.

Purpose

Use MODIFY EXIT to redefine or remove an existing dynamic exit point during initialization.

You can also redefine or remove an existing dynamic exit point after initialization by using the MODIFY EXIT command. For more information, see MODIFY EXIT in *z/VM: CP Commands and Utilities Reference*.

How to Specify

Include as many statements as needed; they are optional. You can place MODIFY EXIT statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

exit
is the number of the exit point you are redefining or removing. This value must be a hexadecimal number between X'0' and X'FFFF'. The recommended value is between X'F000' and X'FFFF', because that range is reserved for customer use. All other exit numbers are reserved for IBM, vendor, or general use.

CLEAR
tells CP to remove the exit.

AT *entry + offset instruction*
identifies the new location for the exit point you are redefining and the instruction that is located at the exit point. The variable *entry* must be a 1-to-8-character string. The first character must be alphabetic or one of the following special characters: $ (dollar sign), # (number sign), _ (underscore), or @ (at sign). The rest of the string can be alphabetic or numeric characters, the four special characters ($, #, _, or @), or any combination. The variable *offset* must be a 1-to-4-character even hexadecimal number between X'0' and X'FFFE'. The variable *instruction* must be a 2-, 4-, or 6-character hexadecimal number.

NORESolve
tells CP to resolve the entry points associated with this exit number the first time they are called.

RESolve
tells CP to resolve the entry points associated with this exit number when the association is first established. Any existing associated entry points are resolved immediately.

PARM *parameter*
is a list of one or more parameters to be supplied to the exit. Five kinds of tokens can be used to define a parameter:

1. Addresses: strings up to eight characters long, consisting of the hexadecimal digits 0 through 9 and A through F.
2. General Registers: strings beginning with G or R, followed by a decimal number between 0 and 15 or a hexadecimal digit, designating the contents of a general register.
3. Indirection: a percent sign (%), which causes the contents of an address or the contents of an address in a register to be used instead of the address or register contents itself.
4. Arithmetic: a plus sign (+) or minus sign (-).
5. Displacement: strings of up to four hexadecimal digits.

Each parameter string specifies how to combine these tokens to generate a parameter value to be passed to an exit routine. The following is a Backus-Naur definition of the syntax of a parameter:

```
<parameter> ::= <anchor> | <anchor><vector>
   <anchor> ::= <reg> | 0...FFFFFFFF | <anchor>%
   <vector> ::= <modifier> | <vector>% | <vector><modif ier>
 <modifier> ::= +<disp> | -<disp> | +<reg> | -<reg>
      <reg> ::= G<digit> | R<digit>
    <digit> ::= 0...15 | 0...9, A...F
     <disp> ::= 0...7FFF
```

Usage Notes

1. To define a new dynamic exit point, use the DEFINE EXIT statement or command. For more information, see "DEFINE EXIT Statement" on page 100. See also DEFINE EXIT in *z/VM: CP Commands and Utilities Reference*.
2. To load the exit point code into the system execution space, use the CPXLOAD statement or command. For more information, see "CPXLOAD Statement" on page 76. See also CPXLOAD in *z/VM: CP Commands and Utilities Reference*.
3. To activate a new exit after defining it, use the ENABLE EXITS statement or command. For more information, see "ENABLE EXITS Statement" on page 150. See also DEFINE EXIT in *z/VM: CP Commands and Utilities Reference*.
4. To display status and usage statistics information about a specific exit point after initialization, use the QUERY EXITS command. For more information, see QUERY EXITS in *z/VM: CP Commands and Utilities Reference*.
5. To stop CP from calling the entry points and external symbols associated with one or more exit points after defining those exit points, use the DISABLE EXITS command. For more information, see DISABLE EXITS in *z/VM: CP Commands and Utilities Reference*.
6. To remove the customer-written CP routines from the system execution space:
   a. Use the DISASSOCIATE command to revoke all entry point and external symbol assignments that were made with the ASSOCIATE EXIT statement or command. For more information, see DISASSOCIATE in *z/VM: CP Commands and Utilities Reference*.
   b. Use the CPXUNLOAD command to unload the customer-written CP routines. For more information, see CPXUNLOAD in *z/VM: CP Commands and Utilities Reference*.
7. Exit numbers are allocated as follows:
   - X'0000' to X'7FFF' are reserved for IBM use.
   - X'8000' to X'EFFF' are reserved for vendor and general use.
   - X'F000' to X'FFFF' are reserved for private customer use.

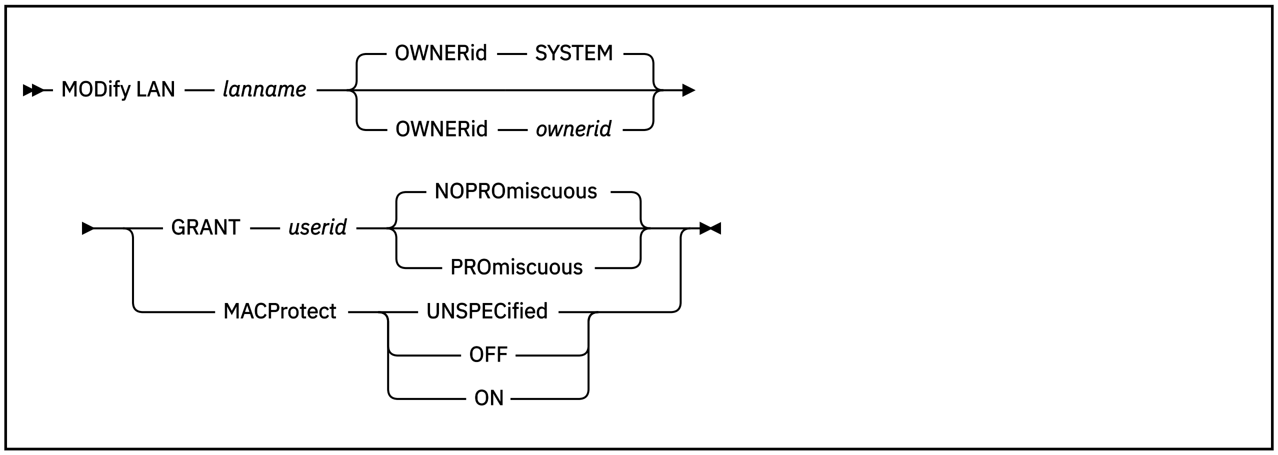
8. The RESOLVE option ensures that the entry names associated with an exit point are defined.
9. Each exit is passed a parameter list that begins with three standard parameters, as described in *z/VM: CP Exit Customization*. Additional parameters, specified by the PARM operand, are optional and follow the first three in the order in which they are specified.
10. Errors (for example, addressing exceptions) during evaluation of user-defined parameters when an exit is being invoked cause CP to abend.
11. For more information about user-defined exit points, see Benefits of using CP exits in *z/VM: CP Exit Customization*.

Examples

1. To change the location of exit point X'F422' to HCPXGMSM + 4, enter the following:

```
Modify Exit f422 At hcpxmgsm + 4 41700001
```

MODIFY LAN Statement



Purpose

Use the MODIFY LAN statement to modify the attributes of an existing guest LAN during initialization.

How to Specify

Include as many statements as needed; they are optional. You can place MODIFY LAN statements anywhere in the system configuration file following a corresponding DEFINE LAN statement.

Operands

lanname
: is the name of the guest LAN to be modified. The *lanname* is a single token (1–8 alphanumeric characters). The combination of *ownerid* and *lanname* will identify the LAN. This LAN must be created by a DEFINE LAN statement in the System Configuration file.

OWNERid *ownerid*
OWNERid SYSTEM
: specifies the owner of the LAN.

GRANT *userid*
: specifies *userid* is added to the access list for this LAN. This allows *userid* to connect an adapter to LAN *lanname* if it is a RESTRICTED LAN. If an External Security Manager (ESM) is in control of the LAN, it may override the CP access list.

PROmiscuous
: specifies that the userid from the grant is authorized for PROMISCUOUS Mode.

NOPROmiscuous
: specifies that the userid from the grant is NOT authorized for PROMISCUOUS Mode.

MACProtect
: modifies the MAC address protection for devices coupled to this guest LAN. Turning MAC address protection on prevents a guest from using a MAC address that is not assigned to the user's network device. MACPROTECT is valid only for an ETHERNET guest LAN.

  There are three levels of inheritance used when determining the MAC address protection level for a MAC address assigned to a network device. The highest is the system level, followed by the guest LAN (or virtual switch) level. The lowest level is the protection set for a specific network data device.

  The MAC address protection level assigned is determined first at the device, then guest LAN and lastly the system level. If MAC address protection is set at the device level using the SET NIC CP command, then its current setting is used. When the device is set to UNSPECIFIED, it will inherit the guest LAN's MACPROTECT setting. When the guest LAN MACPROTECT setting is set to UNSPECIFIED, the

SYSTEM's MAC address protection specified by the SET VMLAN CP command or VMLAN configuration statement is assigned.

UNSPECified

allow the MAC address protection to be determined by the SYSTEM level setting. UNSPECIFIED is the default setting for a guest LAN.

If MAC address protection is set to UNSPECIFIED by the SET NIC CP command for a specific device, then the device's MAC address protection will be set to the SYTEM level defined by the SET VMLAN CP command or the VMLAN configuration statement.

OFF

set the MAC address protection level off for coupled devices defined with a MAC address protection set to UNSPECIFIED.

If MAC address protection is set to UNSPECIFIED by the SET NIC CP command for a specific device, then the device's MAC address protection will be set to off.

ON

set the MAC address protection level on for coupled devices defined with a MAC address protection set to UNSPECIFIED.

If MAC address protection is set to UNSPECIFIED by the SET NIC CP command for a specific device, then the device's MAC address protection will be set on.

Usage Notes

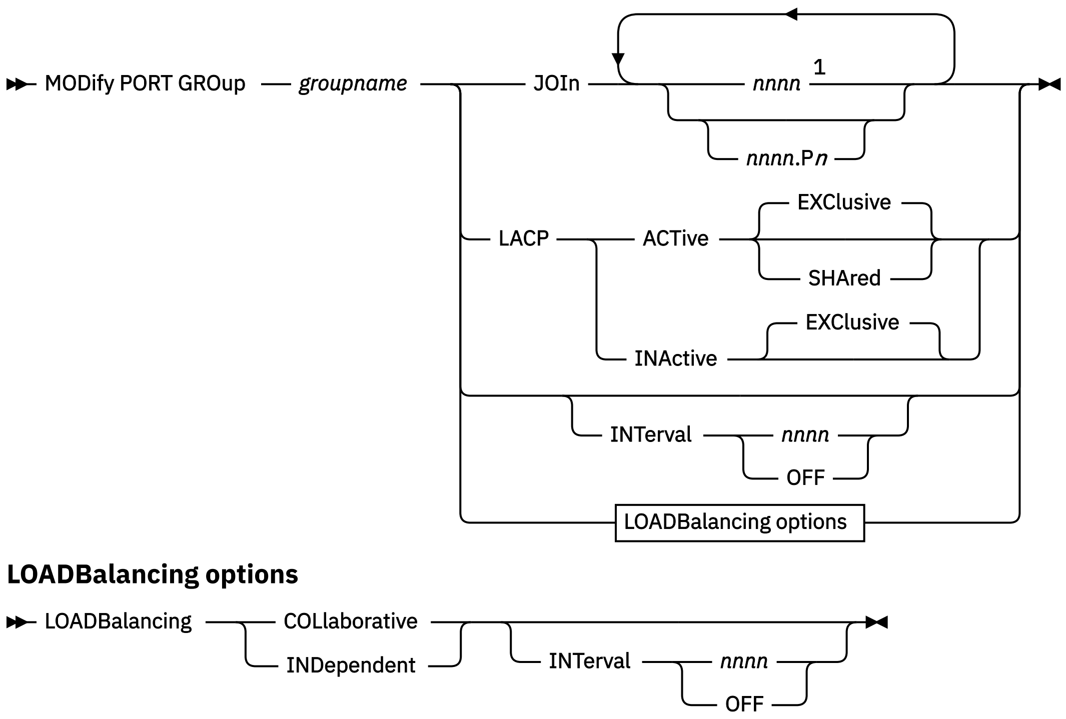
1. If the guest LAN is defined with the UNRESTRICTED attribute, any user can connect a virtual adapter to that LAN. The CP access list has no meaning for an UNRESTRICTED LAN. This will be overridden if an External Security Manager (ESM) is in control of the virtual switch.
2. When a RESTRICTED LAN is defined, the owner is automatically added to the access list.

Examples

1. To modify a SYSTEM LAN named TECHINFO by adding user REYNOLDS to the access list, specify the following:

```
modify lan techinfo grant reynolds
```

MODIFY PORT Statement



Notes:

[1] You can specify a maximum of 8 real device numbers.

Purpose

Use the MODIFY PORT statement to define or change the OSA-Express (QDIO) or Network-Express (EQDIO) devices that make up a link aggregation group and to set the attributes of a link aggregation group.

If the port group specified on the MODIFY PORT GROUP statement does not exist and it is an EXClusive port group, then the group is created. The group persists until the next time that z/VM is IPLed or until the group name is deleted with a SET PORT *groupname* DELETE command.

If the port group specified on the MODIFY PORT GROUP statement does not exist and it is a shared port group, then creation of the shared port group is deferred until the IVL virtual switch UPLINK port connects the host to the IVL domain. Message HCP3178I is displayed for each deferred shared port group. Subsequent statements with the same *groupname* are also deferred.

For more information about virtual networking options in VM, see *z/VM: Connectivity*

How to Specify

Include as many statements as needed; they are optional. You can place MODIFY PORT statements anywhere in the system configuration file. If you specify more than one statement with the same *groupname* with the same JOIN operand, the first definition is accepted and subsequent statements receive an error. For the INTerval operand, the last statement definition overrides any previous specifications.

Operands

groupname

specifies the name of the link aggregation group that is affected by the command. The *groupname* value is a single token (1-8 alphanumeric characters) that identifies a link aggregation group. This group name is associated with a virtual switch via the SET VSWITCH or DEFINE VSWITCH command or the DEFINE VSWITCH statement in the System Configuration file.

JOIn

specifies that the devices (*rdev*) specified join the specified group. If the group does not exist, it is created and persists until the next time that z/VM is IPLed or the group name is deleted with a SET PORT *groupname* DELETE command.

nnnn
nnnn*.P*n

specifies a real networking device within the specified group that is affected by the command.

Specify each real device as a hexadecimal number. The specification can include a hexadecimal port number that is prefixed by lower case or upper case letter P. If no port number is included, it defaults to port 0. The port number follows the device number and is separated by a period.

For example, to specify port 1 for real device 300, specify 300.P1. The port number value is limited by how many ports the adapter hardware feature supports.

Each OSA-Express (QDIO) real device number represents a trio of devices that cooperate to provide a complete network interface. In contrast, each Network Express (EQDIO) real device number represents a complete network interface.

For example, a specification of a QDIO device as RDEV 100 actually uses device numbers 100, 101, and 102 to provide a connection to the real hardware LAN segment. In contrast, a specification of an EQDIO device as RDEV 200 uses only device number 200 to provide the real hardware LAN connection.

You can specify a maximum of 8 real device numbers. If you specify more than one device number, then each must be separated from the others by at least one blank.

If the port group already has at least one operational device, any new device must be compatible with the current attributes (see usage note “3” on page 203.

When the port group is defined with the SHARED option, some devices on the adapters that contain the specified devices are reserved for system use. See Usage Note “3” on page 203 for information about device number assignment for shared port groups.

See Usage Note “1” on page 203 for more information about the requirements for the OSA devices.

LACP ACTive
LACP INActive

LACP ACTive mode specifies that the virtual switch initiates negotiations with the physical switch via the Link Aggregation Control Protocol (LACP) and responds to LACP packets that are sent by the physical switch. LACP INActive specifies that aggregation is performed, but without LACP. It is only useful when the physical switch is also operating without LACP. In this mode, there is no negotiation between the physical switch and the virtual switch.

Ensure that the same LACP setting exists on both the virtual switch and the physical switch.

LACP ACTIVE is the default for a groupname until changed by a MODIFY PORT GROUP LACP command.

SHARED cannot be specified with LACP INACTIVE.

EXClusive

an exclusive port group is a set of OSA adapters that are configured so that each adapter supports only a single VSWITCH QDIO connection. In other words, no sharing of the OSA adapter is permitted.

Once the port group is defined with the EXClusive option, it cannot be changed to a SHAred port group without deleting the port group.

SHAred

when LACP ACTIVE SHARED is specified, it defines a shared port group. A shared port group is a set of OSA adapters that are configured so that one or more global virtual switches and global virtual switch members share the OSA adapter(s) that make up the port group. This defines a Multi-VSwitch LAG (Link Aggregation Group) configuration.CE

A port group that is defined with the SHARED option can be used only with virtual switches that are defined with the GLOBAL option. The virtual switches can be in the same or different LPARs that reside in the same CPC.

The port group has a global scope which means it provides connectivity across multiple virtual switches and multiple systems that run z/VM. It is propagated to all other systems in the same IVL domain. In other words, the Control Program creates a copy of the shared port group with the same name on all other systems that run z/VM and are in the domain (unless the shared port group member is already defined with that name). A change to the shared port group on one system affects all copies of the shared port group on all other systems in the IVL domain.

A port group defined with the SHARED option provides a single point of control for all virtual switches that use the shared port group. For example, a SET PORT GROUP *groupname* LEAVE command for a shared port group causes the specified port to be removed from **all** virtual switches on **all** IVL members that use a copy of the port group with that name.

Because of the global scope, once the port group is defined with the SHARED option, it cannot be changed to an EXClusive port group without deleting the port group. In addition, the LACP mode cannot be changed to INACTIVE.

See *z/VM: Connectivity* for more information.

INTerval *nnnn*
INTerval OFF

indicates the interval to be used by the Control Program (CP) when doing load balancing of conversations across multiple links in the group. It is specified as the number of seconds between load balancing operations across the link aggregation group. It must be a decimal number between 1 and 9990 or the keyword OFF. The interval value is rounded up to the next multiple of thirty seconds. INTERVAL OFF indicates that no load balancing is done by CP.

INTERVAL 300 is the default for a groupname until changed by a MODIFY PORT GROUP INTERVAL command.

When the port group is configured to use the Independent Load Balance management method, the scope of the SET PORT GROUP INTERVAL setting is local to the system it is issued on.

When the port group is configured to use the Collaborative Load Balance management method, the scope of the SET PORT GROUP INTERVAL setting is global (affects all systems within the IVL).

LOADBalancing `INDependent`

Each VSwitch that uses a Link Aggregated Group (LAG) to provide connectivity to an external network is responsible for balancing its own bandwidth across all the network adapters within the group. LOADBALANCING INDEPENDENT is the only allowed load balancing option on an exclusive LAG.By default, exclusive port groups will use the Independent Load Balance management method.

LOADBalancing `COLlaborative`

The Collaborative Load Balance management method will consider and serialize its operation with all the sharing VSwitches. By default, a shared port group will use the Collaborative Load Balance management method.

The Collaborative Load Balance management method can only be used when all active systems connected to the same IVL Domain have support for Collaborative Load Balance. If any z/VM System is added or already exists in the domain which doesn't have support for Collaborative Load Balance, then all Shared Port Groups within the domain will be automatically forced to use the Independent Load Balance management method.

Usage Notes

1. The following restrictions pertain to the network devices in the specified group:
   - The network device must support IEEE802.3ad Link Aggregation.
   - To aggregate multiple network devices within a group, the following criteria must be met:
     - All devices must be the same type: OSA-Express (QDIO) or Network-Express (EQDIO).
     - All devices must connect to the same physical LAG group.
     - All devices must run in full-duplex mode.
     - All devices must operate at the same speed.
2. The following pertain to the OSA devices in an exclusive port group, that is, one that is configured with the EXClusive option:
   - When an OSA port is active in a link aggregated group, it cannot have more than a single QDIO/EQDIO connection established on the adapter (CHPID). Any requests to establish a QDIO/EQDIO connection using another device number on the same CHPID in any logical partition fail until the OSA port is no longer active within the group. Likewise, an OSA port with an active QDIO/EQDIO connection on any device number on the same CHPID cannot join a link aggregated group until there are no active QDIO/EQDIO connections on the port.
   - When the port group is configured as LACP Inactive, all ports in the group must support the network disablement function. Cards that do not support the network disablement function can be used only in a port group that is configured with LACP Active.
3. The following notes pertain to the OSA devices in a shared port group, which is a group that is configured with the SHARED option:
   - Another shared port group with the same name that is defined anywhere in the IVL domain must have a matching configuration. The Control Program propagates the shared port group to all other members of the domain if a copy of the port group does not already exist.
   - When a specific network adapter is configured in a shared port group, CP reserves device numbers for system use on each z/VM system that has an active IVL connection. CP reserves three device numbers for a QDIO adapter or one device number for an EQDIO adapter. Devices that are reserved for system use are shown with "SYSTEM" status in the QUERY OSA display.
   - A shared port group can be shared by multiple global virtual switches that reside in the same system. Each shared port group is assigned an instance (0,1,2,3) when configured to a global virtual switch.
   - The administrator can use either port that is associated with a multiple-port network adapter. Once a port is activated in a shared port group, it is no longer available for guest usage and the ATTACH command cannot be used to specify devices that are reserved for system use. Likewise, a network connection established by a guest on a specific port prevents its use in a shared port group.
   - When a shared port group is in use on a global virtual switch, an RDEV list cannot be used for the virtual switch to provide back up connectivity. Instead, back up connectivity is provided by a different global virtual switch member using the shared port group.
   - A shared port group can be used by up to 64 global virtual switch members across the IVL domain. The members can be in the same or different global virtual switches.
4. The maximum number of groups that can be defined to CP is 128.
5. It is invalid to create a group name of ALL.
6. The number of devices that are identified by the RDEV keyword on the DEFINE or SET VSWITCH command combined with the number of devices in an associated GROUP cannot exceed eight devices.
7. When the virtual switch has been defined with the GROUP attribute (using exclusive port group), any devices that are identified by the RDEV keyword (on the DEFINE or SET VSWITCH command) are used for failover in the event of a real switch failure of the link aggregation group. Failover in this environment is to a single OSA device that is connected to a second real switch. The virtual switch recognizes when the port group is available again and will initiate recovery to the port group

member(s). When the virtual switch is defined with GROUP and RDEV NONE, it means that there is no link aggregation group failover in the event that the real switch should fail.

8. The port number that is specified by *nnnn* .***P****nn* is used to initialize device *nnnn*. If the port number is not specified, it defaults to port 0. If the port number is not supported, device initialization is terminated.

Examples

Example 1: To define a group named LINGROUP that contains multiple devices, specify the following:

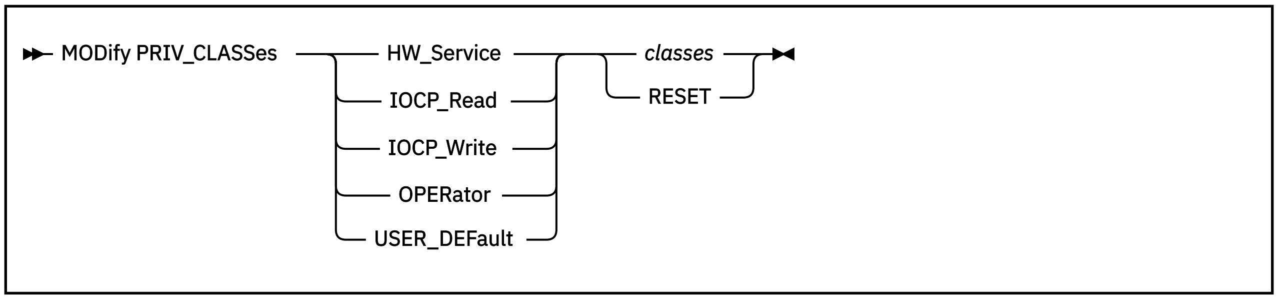
```
modify port group lingroup join B00 C00 D00
```

Example 2: To define a shared port group with the IVL VSwitch and global VSwitch, specify the following:

```
define vswitch ivlvsw type ivl domain B vlan 11 native none rdev 900 A00
modify port group prodlag1 lacp active shared
modify port group prodlag1 join B00 C00 D00
define vswitch prodvsw1 global ethernet vlan aware native none group prodlag1
```

A00 and 900 are OSA adapters that must not have LACP configured on their external network switch ports. They should be on the same LAN segment. B00, C00, and D00 are to be aggregated and must have LACP configured on their external network switch ports.

MODIFY PRIV_CLASSES Statement



Purpose

Use MODIFY PRIV_CLASSES to change the privilege classes authorizing the following CP functions:

- Logging on as the primary system operator
- Intensive error recording
- Using the read function of the CP IOCP utility
- Using the write function of the CP IOCP utility
- Specifying the default user class.

Parameters

HW_Service
tells CP to change the privilege classes authorized to perform intensive error recording.

IOCP_Read
tells CP to change the privilege classes authorized to use the read function of the IOCP utility.

IOCP_Write
tells CP to change the privilege classes authorized to use the write function of the IOCP utility. For more information, see IOCP in *z/VM: CP Commands and Utilities Reference*.

OPERator
tells CP to change the privilege classes which are for the primary system operator.

USER_DEFault
tells CP to change the privilege classes that are defined for users who do not have a class specified in their virtual machine definitions.

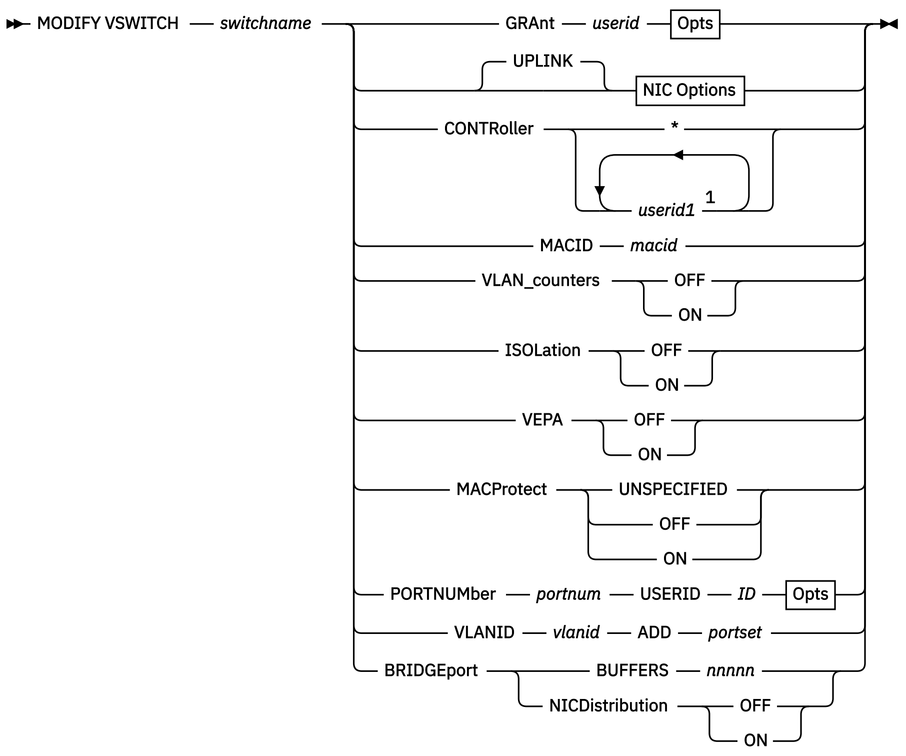
RESET
tells CP to reset the privilege classes to their original default settings.

classes
tells CP which class or classes are to be set. The variable *classes* is a 1- to 8-character alphanumeric string from A through Z and from 0 to 6.

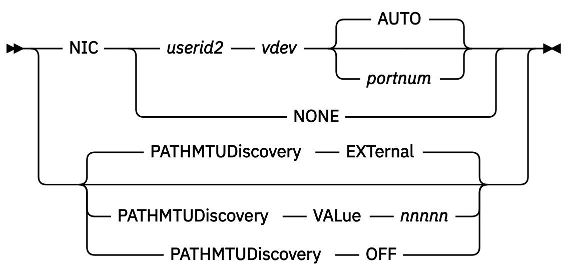
Usage Notes

1. It is suggested that you simply use the PRIV_CLASSES statement to establish initial privilege class values.
2. If you previously used the OVERRIDE utility (no longer supported) to change the privilege classes, you can use the CVTOVRID utility to convert your override file into a set of MODIFY statements. For more information, see CVTOVRID in *z/VM: CP Commands and Utilities Reference*.

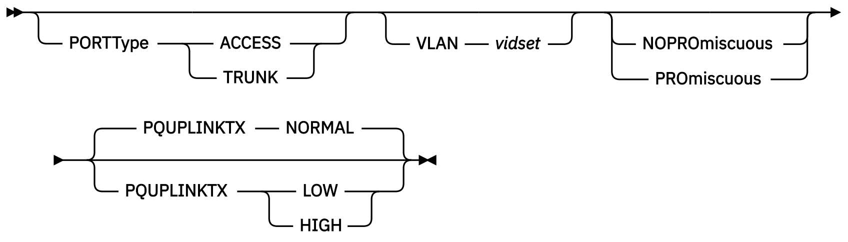
MODIFY VSWITCH Statement



NIC Options



Opts (Grant and Portnumber Options)



Notes:

[1] You can specify a maximum of 8 user IDs.

Purpose

Use the MODIFY VSWITCH statement to modify the attributes of an existing virtual switch.

How to Specify

Include as many statements as needed; they are optional. You can place MODIFY VSWITCH statements anywhere in the system configuration file following a corresponding DEFINE VSWITCH statement. If you specify more than one statement with the same *switchname* with the same operands, the last statement definition overrides any previous specifications.

Operands

switchname

is the name of the virtual switch to be modified. The *switchname* is a single token (1–8 alphanumeric characters) that identifies the virtual switch. This virtual switch was created by a DEFINE VSWITCH command or statement in the System Configuration file.

GRANT *userid*

specifies the user ID (*userid*) to be either added or modified in the access list for this virtual switch. This allows the user ID to connect an adapter to *switchname*. If an External Security Manager (ESM) is in control of the virtual switch, it may override the CP access list.

If the user ID is not currently in the access list, it will be added with the specified network characteristics. Any network characteristics not specified will use the defaults from the virtual switch.

If the user ID already exists in the access list, then only the network characteristics specified on this command will be changed.

The MODIFY VSWITCH statement with GRANT creates a z/VM assigned port (in the range of 2176-4095 inclusive) representing the intended configuration if a port in this range does not already exist. The port defined by this command will persist until the VSWITCH is destroyed or the GRANT is removed by the SET VSWITCH command with the REVOKE option. If any ports are already configured for this user, update each port as specified by this new GRANT specification.

PORTType

defines the type of VSwitch connection. If the GRANT or PORTNUMBER operand is being used to add a new user or port to the access list, and the virtual switch has been defined as VLAN aware and no PORTTYPE is specified on the GRANT or PORTNUMBER, the porttype assigned to the guest or port will be the default porttype as specified on the DEFINE VSWITCH command or statement.

If the GRANT or PORTNUMBER operand is issued for a user or port that has already been added to the access list, and the virtual switch has been defined as VLAN aware and no PORTTYPE is specified on the GRANT or PORTNUMBER, the current setting of porttype for the guest or port will be used.

PORTType ACCESS

defines the type of connections that are established to be an access port. The guest is unaware of VLAN IDs and sends and receives only untagged traffic.

PORTType TRUNK

defines the type of connections that are established to be a trunk port. The guest is VLAN aware and sends and receives only tagged traffic for those VLANs to which the guest is authorized. If the guest is also authorized to the *natvid*, untagged traffic sent or received by the guest is associated with the native VLAN ID (*natvid*) of the virtual switch.

VLAN *vidset*

identifies the VLAN ID (or set of VLAN IDs) to which this user is restricted while connected to *switchname*. If VLAN is not specified, the default VLAN for this user is the default VLAN ID as specified on the DEFINE VSWITCH command or statement. Note that when a virtual switch is defined as VLAN AWARE, a default VLAN ID is not assigned. If a default VLAN ID is not assigned when the virtual switch is defined, then all inbound or outbound frames are discarded until a VLAN ID is assigned.

If VLAN is specified, each item in the *vidset* must be a valid VLAN number (between 1 and 4094, inclusive). The *vidset* may contain ranges:

Table 10. vidset Ranges

Examples:
VLAN 100-199 VLAN 100-199 1000-1099
VLAN 1 101-199
VLAN 1 101-199 4094
VLAN 1 2 3
VLAN 1 101 201 301

If an External Security Manager (ESM) is in control of the virtual switch, it can override the CP access list.

PROmiscuous
specifies that the user ID or port is authorized for PROMISCUOUS Mode.

NOPROmiscuous
specifies that the user ID or port is NOT authorized for PROMISCUOUS Mode.

PQUPLINKTX
specifies the priority for all packets sent from a virtual NIC's network connection to an external network. The PQUPLINKTX value will be used when priority queuing is enabled on the virtual switch (PRIQueuing ON). If PQUPLINKTX is configured for a virtual switch that does not have priority queuing enabled, the setting will be saved and used if priority queuing is enabled at a later time. If PQUPLINKTX is not specified, the default is that all outbound traffic to the external network will be sent at Normal priority. See PRIQueuing operand on the DEFINE VSWITCH statement for additional information.

This option is not allowed for IVL virtual switches. For an IVL virtual switch, z/VM handles the priority of outbound transmissions.

Outbound traffic from the Hipersockets Bridgeport will be sent at Normal priority.

LOW
specifies that outbound traffic to the external network will be sent at a low priority. This traffic will use the low priority queue which is serviced less frequently than the normal or high priority queues.

NORMAL
specifies that outbound traffic to the external network will be sent at a normal priority. This traffic will use the normal priority queue which is serviced less frequently than the high priority queue but more frequently than the low priority queue.

HIGH
specifies that outbound traffic to the external network will be sent at a high priority. This traffic will use the high priority queue which is serviced more frequently than the normal or low priority queues.

UPLINK
enables and specifies connectivity for a virtual switch UPLINK port. The UPLINK port is a special port that typically is used to connect the virtual switch to a physical switch -- essentially bridging the virtual switch's simulated network to a physical network.

NIC
causes the UPLINK port for a virtual switch to be connected to, or disconnect from, a virtual switch guest port. The NIC operand can be specified only when a virtual switch is in the DISCONNECT state. Setting the UPLINK port to a guest port also will set RDEV NONE and NOGROUP. Changing either of the RDEV or GROUP settings causes UPLINK NIC NONE to be set.

***userid2 vdev* (AUTO/*portnum*)**

specifies that the UPLINK port is to be connected to a guest port, and that the virtual switch is not to be connected to a physical switch. In this configuration the virtual switch routes all broadcasts and destination MAC or IP Address unicast frames that cannot be resolved within the simulated LAN segment to the specified guest port. If your virtual switch is VLAN AWARE, the guest port should be configured as a TRUNK port.

- AUTO - CP determines the VSWITCH port number to be used.

  If a single user port was defined for this user by the MODIFY VSWITCH PORTNUMBER statement, then that port will be automatically used by COUPLE or CONNECT, and its port settings will be used. If there are multiple user ports defined, then the CONNECT or COUPLE command will fail.

- *portnum*

  The guest portnum may be specified. When the uplink NIC is COUPLEd or CONNECTed, it is connected to the specified *portnum* on the virtual switch. The portnumber must be a valid user-defined port number (between 1 and 2048, inclusive), and must be configured with the MODIFY VSWITCH PORTNUMBER statement. If the *portnum* is not a configured port, then the command will fail.

NONE

disconnects the currently connected guest port from the special virtual switch UPLINK port. Until another UPLINK port is specified, all destination MAC or IP Address unicast frames that cannot be resolved within the simulated LAN segment are discarded

PATHMTUDiscovery

indicates that CP will, by default, participate in the Path MTU Discovery process in order to minimize packet loss and to maximize the size of packets sent through a defined UPLINK RDEV port device. The Path MTU Discovery process performed by CP examines and discards any IP datagrams destined for the UPLINK port that are larger than the Path MTU Discovery value allowed by the UPLINK RDEV port device. CP also sends to the datagram source an appropriate notification in the form of an ICMP or ICMPv6 message.

The MODIFY VSWITCH PATHMTUDISCOVERY option can be used to configure a user-defined Path MTU Discovery value, to indicate that CP should determine the Path MTU Discovery value (the default), or to indicate to CP that the Path MTU Discovery process is to be disabled.

These options are specified as follows:

EXTernal

indicates that CP should determine the Path MTU Discovery value to assign. The value to be assigned is determined as follows:

- If the virtual switch is configured with an active UPLINK RDEV port, CP determines the Path MTU Discovery value to use from the maximum MTU value supported by the connected OSA UPLINK RDEV port device. Typically, this value is 8992 bytes.
- If the virtual switch is configured with one or more UPLINK RDEV ports but none are currently active, CP disables the Path MTU Discovery process, effectively the same as specifying the MODIFY VSWITCH PATHMTUDISCOVERY OFF option. Later, when one of the UPLINK RDEV port devices is activated, CP determines the Path MTU Discovery value to use based on the maximum MTU value supported by the active OSA UPLINK RDEV port device. Typically, this value is 8992 bytes.
- If the virtual switch is configured without an UPLINK RDEV port device, CP disables the Path MTU Discovery process, effectively the same as specifying the MODIFY VSWITCH PATHMTUDISCOVERY OFF option. Later, when the UPLINK RDEV port device is defined and activated for the virtual switch, CP determines the Path MTU Discovery value to use based on the maximum MTU value supported by the active OSA UPLINK RDEV port device. Typically, this value is 8992 bytes.

PATHMTUDISCOVERY EXTERNAL is the default setting for the virtual switch.

VALue *nnnnn*
configures a user-defined value of *nnnnn* to be used by CP as the Path MTU Discovery value. Specifying this option can be useful if the Path MTU Discovery value of the physical network needs to be different from the Path MTU Discovery value used by CP when MODIFY VSWITCH PATHMTUDISCOVERY EXTERNAL is specified. To determine the recommended Path MTU Discovery value, see the Maximum Transmission Unit (MTU) information in the hardware documentation associated with the OSA Uplink RDEV port device. The range of values for *nnnnn* must be a minimum of 512 bytes and a maximum of 65535 bytes.

OFF
indicates that the Path MTU Discovery process for this virtual switch is to be disabled by CP. This setting is not recommended when the virtual switch is defined with a Bridge Port.

See usage note "8" on page 214 for more information about the PATHMTUDISCOVERY option.

CONTroller * | *userid1*
identifies the z/VM user ID that controls the OSA-Express device connected to the virtual switch. You can specify a maximum of eight user IDs. If you specify more than one user ID, each must be separated from the others by a least one blank. CONTROLLER * means CP selects from any of the eligible z/VM TCP/IP stacks. See usage note "4" on page 213 for more information about the function of a controller.

Note: A controller virtual machine is not required and is not used to manage a Network-Express (EQDIO) device interface. An EQDIO device is controlled by VSwitch internal logic. If, however, a BRIDGEPORT HiperSockets connection is configured, a controller virtual machine is required to manage the HiperSockets device interface.

If you specify multiple real devices on the RDEV keyword or through the GROUP keyword, then specify CONTROLLER * or allow it to default or specify a list of user IDs. The controller functions are then spread across multiple z/VM TCP/IP stacks, providing more flexibility in case of a failure. You can specify a pool of specific controllers to be chosen from by specifying a list of user IDs after the CONTROLLER keyword with the SET VSWITCH command or the MODIFY VSWITCH statement.

MACID *macid*
is a unique identifier (up to six hexadecimal digits) used as part of the virtual switch MAC address. This MACID (three bytes) is appended to the system MACPREFIX or USERPREFIX (three bytes) to form a unique MAC address for this virtual switch. If a specified MACID is already in use the virtual switch is not changed. You must eliminate the conflict or select a different MACID. If no MACID is set for the virtual switch, CP generates a unique identifier for this virtual switch.

VLAN_counters
indicates whether or not detailed support is provided for VLAN counters. This affects the level of detail that should be maintained for the virtual switch network activity.

OFF
indicates that the counters required are not needed at the VLAN level. That is, transmission counters should be kept at the port level. Packet counters are kept for each port to show the number of frames received, transmitted or discarded.

ON
indicates that transmission counters should be kept at the VLAN level. This is, for each port, packet counters are maintained to show the number of frames received, transmitted or discarded for each VLAN.

VLAN_counters OFF is the default.

ISOLation
determines if guests on the virtual switch can communicate between themselves and other hosts and/or LPARs that share the same OSA port with the virtual switch. ISOLATION OFF is set by default for all virtual switch types to allow unrestricted network communications.

OFF
allows guest ports to communicate with each other and with any hosts and/or LPARs that share the same OSA port.

ON
: prohibits guests from sending traffic to other guests on the same virtual switch by discarding traffic that is destined for another guest port on the virtual switch. In addition, no direct LPAR communications sharing the same OSA port is permitted with the guest ports of the virtual switch. All traffic from the virtual switch destined for any sharing hosts/LPARs on the same OSA port will be dropped. Any traffic destined for the virtual switch guest ports from hosts/LPARs sharing the same OSA port will also be dropped.

VEPA
: determines the operational mode of the virtual switch with regard to forwarding guest-to-guest and guest-to-external destination communications.

OFF
: allows guest ports to communicate with each other and with any hosts or LPARs or both that share the same OSA port. VEPA OFF is the default setting for a QDIO virtual switch.

ON
: prohibits guests from sending traffic to other guests on the same virtual switch without going through an external entity by forwarding all traffic from the guest through the OSA uplink to an adjacent switch. In addition, no direct LPAR communications sharing the same OSA port is permitted with the guest ports of the virtual switch. All traffic from the virtual switch destined for any sharing hosts/LPARs on the same OSA port will be forwarded as well. Any traffic destined for the virtual switch guest ports from hosts/LPARs sharing the same OSA port will also be forwarded to the adjacent switch. VEPA ON requires an ETHERNET virtual switch (without a Bridge Port) with OSA uplink(s) that supports VEPA as a well as the partner switch must support Reflective Relay. VEPA ON may not be specified if ISOLATION is ON.

MACProtect
: turns on or off MAC address protection for devices coupled to this virtual switch. Turning MAC address protection on prevents a guest from using a MAC address that is not assigned to the user's network device. MACPROTECT is valid only for an ETHERNET virtual switch.

There are three levels of inheritance used when determining the MAC address protection level for a MAC address assigned to a network device. The highest is the system level, followed by the virtual switch (or guest LAN) level. The lowest level is the protection set for a specific network data device.

The MAC address protection level assigned is determined first at the device, then virtual switch and lastly the system level. If MAC address protection is set at the device level using the SET NIC CP command, then its current setting is used. When the device is set to UNSPECIFIED, it will inherit the MACPROTECT setting of the virtual switch. When the virtual switch MACPROTECT setting is set to UNSPECIFIED, the SYSTEMs MAC address protection specified by the SET VMLAN CP command or VMLAN configuration statement is assigned.

UNSPECified
: allow the MAC address protection to be determined by the SYSTEM level setting. UNSPECIFIED is the default setting for a virtual switch.

If MAC address protection is set to UNSPECIFIED by the SET NIC CP command for a specific device, then the device's MAC address protection will be set to the SYSTEM level defined by the SET VMLAN CP command or the VMLAN configuration statement.

OFF
: set the MAC address protection level off for coupled devices defined with a MAC address protection set to UNSPECIFIED.

If MAC address protection is set to UNSPECIFIED by the SET NIC CP command for a specific device, then the device's MAC address protection will be set off.

ON
: set the MAC address protection level on for coupled devices defined with a MAC address protection set to UNSPECIFIED.

If MAC address protection is set to UNSPECIFIED by the SET NIC CP command for a specific device, then the device's MAC address protection will be set on.

PORTNUMber *portnum* USERID *ID opts*

The PORTNUMber operand defines a port for a virtual switch and associates the specified user ID (*ID*) with a virtual switch port number (*portnum*). This operand will also be used to specify port options including porttype, promiscuous and VLAN ids. The user ID will be added to the access list for this virtual switch. This allows the user ID to connect an adapter to *portnum* on virtual switch *switchname*. If an External Security Manager (ESM) is in control of the virtual switch, the user ID must be authorized by the ESM, and the VLANs configured with the MODIFY VSWITCH PORTNUMBER statement must be included in the VLAN list provided by the ESM. The *portnum* must be a valid port number (between 1 and 2048, inclusive).

This option is not allowed for a TYPE IVL virtual switch.

VLANID *vlanid* ADD *portset*

The ADD operand adds the port set to the specified virtual lan segment. The *vlanid* must be a valid VLAN number (between 1 and 4094, inclusive). Each item in the *portset* must be a valid port number (between 1 and 2048, inclusive). Each port must be defined with the MODIFY VSWITCH PORTNUMBER statement before being added to a VLAN. The *portset* may contain ranges:

Table 11. portset Ranges

Examples:
1-25
1 25-50
1 25-50 2048
1 2 3
1 101 201 301

This option is not allowed for a TYPE IVL virtual switch.

If an External Security Manager (ESM) is in control of the virtual switch, the VLAN ids defined with VLANID command must be included in the VLAN list provided by the ESM.

BRIDGEport

specifies buffering and NIC distribution properties of the Bridge Port.

BUFFERS *nnnnn*

indicates the maximum number of asynchronous requests that are allowed to be outstanding on the Bridge Port. The allowable range is 0 to 65535. The default value is 2048.

NICDistribution

specifies how Ethernet frames flow from HiperSockets bridge capable ports on the HiperSockets CHPID into the virtual switch uplink or other simulated guests that are coupled to the virtual switch.

OFF

The virtual switch does not examine each Ethernet frame to identify the HiperSockets guest port that sent the frame. The virtual switch distributes all Ethernet frames from the HiperSockets CHPID as a single source. OFF is the default setting if you do not specify the NICDISTRIBUTION parameter.

ON

The virtual switch examines each Ethernet frame to identify the HiperSockets guest port that sent the frame. The virtual switch uses the HiperSockets guest port information to distribute the frames among the virtual switch uplinks or other simulated guests that are coupled to the virtual switch. The information is used to balance the load, especially when links are in a link-aggregation port group configuration and when most guests are on the HiperSockets CHPID.

When NICDISTRIBUTION is ON, additional information that is related to the HiperSockets logical guest ports is displayed by the QUERY VSWITCH command, in monitor records, and by

Diagnose x'26C'. For example, packet counters and IP addresses of the guests are displayed. For more information, see QUERY VSWITCH in *z/VM: CP Commands and Utilities Reference*.

Usage Notes

1. For more information, see Planning for Guest LANs and Virtual Switches in *z/VM: Connectivity*.
2. If the virtual switch has been defined as VLAN UNAWARE, the following are not allowed:
   - the PORTTYPE and the VLAN option on GRANT
   - the PORTTYPE and the VLAN option on PORTNUMBER
   - the VLANID option
3. If the virtual switch has been defined as VLAN aware and no VLAN membership is given with the GRANT, PORTNUMBER, or the VLANID (or provided by an external security manger), the guest is a member of the virtual switch's default VLAN ID (*defvid*) only if a defvid was assigned as part of the virtual switch definition.
4. With a QDIO uplink device, a virtual switch's connection to a real hardware LAN segment is not operational until an eligible TCP/IP stack is selected to be the controller for the OSA-Express device. CP selects an eligible TCP/IP stack to be the controller by either:
   - It is recommended that the VSWITCH controller be at the same release level as CP, although all supported releases are allowed.
   - If CONTROLLER *userid1* is specified on the DEFINE or SET VSWITCH commands or the DEFINE or MODIFY VSWITCH system configuration statements, with either a single user ID or a list of user IDs, only those user IDs are selected.
   - If CONTROLLER * is specified or allowed to default, CP selects from any eligible TCP/IP stack.

   A TCP/IP stack becomes eligible when:
   - The TCP/IP MODULE running in the controller is at a release level that supports the function required for the virtual switch.
   - An IUCV *VSWITCH statement is included in its virtual machine definition.
   - The TCP/IP VSWITCH CONTROLLER statement is coded, and has defaulted to be ON or is explicitly set to ON in the TCP/IP configuration file or through an OBEYFILE command.
   - The stack has completed initialization.
   - The stack has virtual device numbers available for CP to attach the control device.

     The virtual device range used by CP is specified in the VSWITCH CONTROLLER TCP/IP configuration statement. If no VDEV range is specified, CP uses the virtual device number (*vdev*) that matches the *rdev* number specified on the DEFINE VSWITCH or SET VSWITCH command. For more information, see VSWITCH CONTROLLER Statement in *z/VM: TCP/IP Planning and Customization*.

     Note: Do not code DEVICE and LINK TCP/IP configuration statements for the device. Do not attach the device to a TCP/IP controller virtual machine. These steps are handled by DEFINE VSWITCH processing when a controller is selected.

   If an eligible stack is not found, or none of the *rdevs* are operational, you receive a message, and the virtual switch operates in a local LAN environment.

   With an EQDIO uplink device, a controller is not required because the virtual switch logic controls the queue. The operational status of the virtual switch's connection to a real hardware LAN segment does not depend on controller selection or eligibility.
5. MODIFY VSWITCH VLAN_counters ON can significantly affect the rate of transmission for a virtual switch. Use this option only if there is a need to understand traffic patterns at the level of VLAN IDs. One example of when you might want VLAN_counters ON is when you are using a network management tool to analyze VLAN network traffic.
6. Detailed transmission counters are available only for a VLAN AWARE Virtual Switch with the VLAN_counters option set to ON. These values may be obtained using the Diagnose X'26C' interface.

7. If the virtual switch is VLAN aware, the VLANs need to be configured with the GRANT, VLANID or PORTNUMBER command or via the NICDEF statement in the user directory. If an External Security Manager (ESM) is in control of the virtual switch, the VLANs configured (with GRANT, VLANID, PORTNUMBER or NICDEF) must be included in the VLAN list provided by the ESM.
8. The following notes are related to the MODIFY VSWITCH PATHMTUDISCOVERY option:
   - When a PORT GROUP is defined using the SET PORT command, the PATHMTUDISCOVERY option applies to all devices in the group.
   - If a virtual switch is configured with an Uplink NIC, the MODIFY VSWITCH PATHMTUDISCOVERY option is not applicable. However, it can be set for use later when an Uplink RDEV port device is defined.
   - If a virtual switch is configured with the NOUPLINK option, the MODIFY VSWITCH PATHMTUDISCOVERY option is not allowed.

     Note: When MODIFY VSWITCH statements are used in conjunction with NICDEF statements (and Directory Network Authorization is enabled) to configure the network attributes, the following rules apply:

     – No MODIFY VSWITCH configuration is required if the NICDEF statement provides all necessary network configuration.

     – NICDEF attributes override any prior MODIFY VSWITCH configuration (and this is reflected in subsequent QUERY VSWITCH output).

Examples

The following examples apply when CP, not an ESM, is protecting the guest LAN:

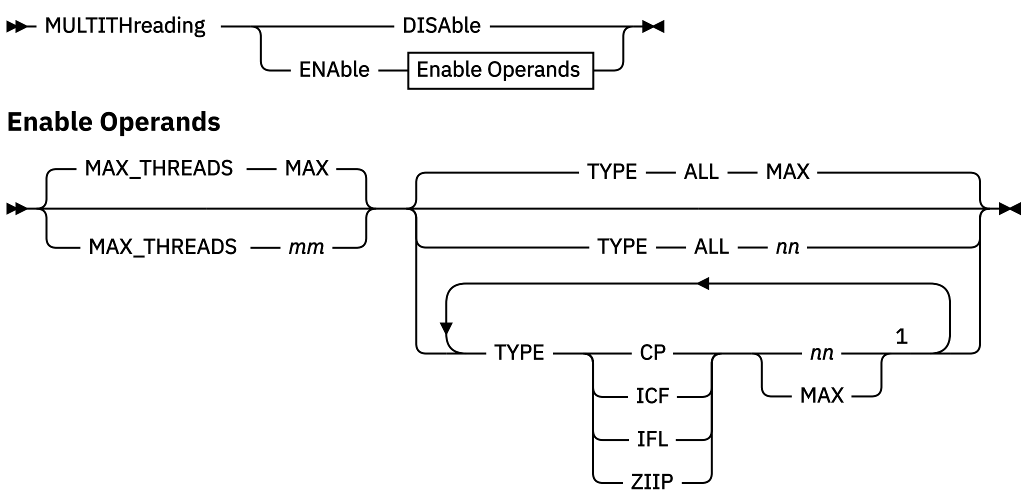
1. To allow user VMTCPIP to connect to virtual switch BIGANG and filter any incoming traffic using VLAN 9, specify the following:

```
modify vswitch bigang grant vmtcpip vlan 9
```

2. To define a port based virtual switch VSWITCH1 with multiple access ports 10 and 20 for user1, specify the following. Port 10 will be added to VLAN 11 and port 20 will be added to VLAN 21.

```
define vswitch vswitch1 portbased vlan aware
modify vswitch vswitch1 portnumber 10 userid user1
modify vswitch vswitch1 portnumber 20 userid user1
modify vswitch vswitch1 vlanid 11 add 10
modify vswitch vswitch1 vlanid 21 add 20
```

MULTITHREADING Statement



Notes:

[1] You can specify each CPU type only once.

Purpose

Use the MULTITHREADING statement to define the multithreading characteristics of the system. The statement also defines system options that can be customized if multithreading is enabled. Multithreading support is available on the system only if the simultaneous multithreading (SMT) facility is installed on the hardware.

How to Specify

The MULTITHREADING statement is optional. Only one statement is allowed. Subsequent MULTITHREADING statements cause an error message to be displayed and are not processed. If the MULTITHREADING statement is not specified, multithreading will be disabled on the system.

Operands

DISAble
ENAble

specifies whether multithreading is enabled or disabled on the system.

Multithreading will be enabled only if z/VM is configured to run with the HiperDispatch vertical polarization mode enabled and with the dispatcher work distribution mode set to reshuffle. For more information, see "SRM Statement" on page 269.

MAX_THREADS MAX
MAX_THREADS *mm*

specifies the upper limit of the number of threads to activate on each core. The value *mm* must be a decimal number 2 - 32.

If MAX_THREADS is not specified, or if MAX_THREADS MAX is specified, or if *mm* is greater than the number of threads per core supported by the hardware and z/VM, the maximum supported threads value is used. This value is the lesser of the highest number of threads per core available on the hardware and the highest number of threads per core supported by z/VM.

The value used for MAX_THREADS directly affects the processor numbers assigned to each thread. This value, rounded up to a power of 2, is the number of CPU addresses assigned to each core.

TYPE ALL {*nn*|MAX}
TYPE CP {*nn*|MAX}
TYPE ICF {*nn*|MAX}
TYPE IFL {*nn*|MAX}
TYPE ZIIP {*nn*|MAX}

specifies the CPU type and the number of threads for that type. If ALL is specified, no individual CPU types may be specified.

The value specified by this operand, and the maximum supported threads as defined in the description of the MAX_THREADS operand, tell z/VM the number of threads to activate on each CPU type. The value *nn* must be a decimal number 1 - 32 and cannot be greater than the value specified for the MAX_THREADS operand. If ALL or an individual CPU type is not specified, or if MAX is specified, the maximum supported threads value for the CPU type is used.

Usage Notes

1. The number of threads activated per core can be set only at system IPL, using this configuration statement, and modified after IPL using the SET MULTITHREAD command. However the SET MULTITHREAD command can be used only if multithreading is enabled using the MULTITHREADING statement.
2. This level of z/VM supports up to 2 threads for IFL CPUs and 1 thread for all other CPU types.
3. If MAX is specified or allowed to default for MAX_THREADS, or for ALL, or for any of the CPU type operands, the multithreading level might change when the system is migrated to a new machine or is upgraded to a new level of z/VM.
4. When multithreading is enabled, each core in the configuration comprises multiple processors, each with its own processor address. The processor address then consists of the bits comprising the core ID, concatenated with bits comprising a thread ID that ranges from zero to MAX_THREADS minus 1. The number of bits in the thread ID is the smallest number that can accommodate this range. For example, when MAX_THREADS is 2, one bit is needed to represent thread IDs 0 and 1; thus core ID 0003 is shifted left one bit, so that core ID 0003 comprises processor addresses 0006 and 0007. In this example, any processors defined above 0027 would, when multithreading is enabled, become cores with processor addresses greater than 004F. z/VM supports up to 80 (address 004F) processors. The processors with addresses above 004F will be offline at IPL.

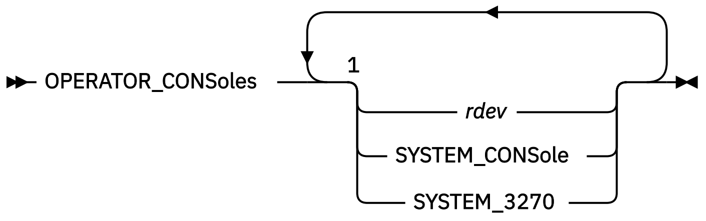
   For more information, see "CPU-Address Expansion" in z/Architecture Principles of Operation (https://publibfp.dhe.ibm.com/epubs/pdf/a227832d.pdf).
5. To display the system's multithreading status, use the QUERY MULTITHREAD command. To display processor core utilization information, use the INDICATE MULTITHREAD command. For more information, see QUERY MULTITHREAD and INDICATE MULTITHREAD in *z/VM: CP Commands and Utilities Reference*.

Example

To enable multithreading at the maximum values on the system, use the following statement:

```
Multithreading Enable
```

OPERATOR_CONSOLES Statement



Notes:

[1] The maximum number of device numbers that you can specify is 500.

Purpose

Use the OPERATOR_CONSOLES statement to define a list of device numbers from which CP is to choose the system operator console. CP will log on the system operator virtual machine at the first available address or location specified with the command.

How to Specify

Include as many statements as needed; they are optional. You can place OPERATOR_CONSOLES statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

rdev
is the real hexadecimal device numbers of possible system operator consoles. These devices must be locally-attached 3270-type supported displays.

The maximum number of device numbers that you can specify is 500. When you initialize the system, CP goes sequentially through the list defined with the OPERATOR_CONSOLES statement and looks for the first operational console. When CP finds an operational console, it uses it as the system operator's console. If CP cannot find an operational console, it enters a disabled wait state with a wait state code of X'1010' in the PSW.

SYSTEM_CONSole
specifies that the Operating System Messages panel on the IBM Hardware Management Console (HMC) can serve as a system operator console.

SYSTEM_3270
specifies that the integrated 3270 console on the HMC can serve as a system operator console.

Usage Notes

1. If an IODF statement is defined in the system configuration file with the *osconfig* parameter specified, then the software I/O configuration will be controlled by HCD. In this case, the OPERATOR_CONSoles statement is ignored if it is specified. For more information, see "IODF Statement" on page 181.
2. Consoles that CP is to notify when there is a system emergency (for example, an impending abend, shutdown, or dump) should be listed on the EMERGENCY_MESSAGE_CONSOLES statement. For more information, see "EMERGENCY_MESSAGE_CONSOLES Statement" on page 144.
3. 2250 and 3250 displays are not valid operator consoles.
4. During a software re-IPL caused by either a system incident or the CP SHUTDOWN REIPL command, CP determines whether the operator virtual machine is connected to the primary system console. If

the operator virtual machine is disconnected, logged off, or logged on to a console other than the primary console, CP disconnects the operator virtual machine when the system is reinitialized, unless you specified the NODISCONNECT operand on a SYSTEM_USERIDS statement. This disconnection is a security measure to reduce the risk of CP's logging the operator virtual machine on to an unattended console.

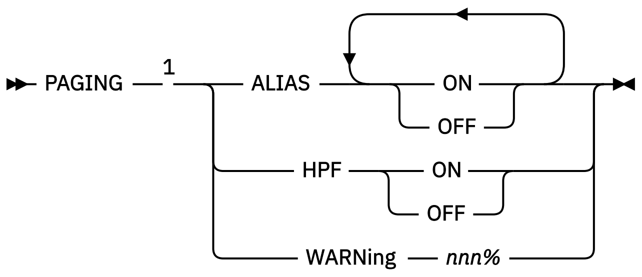
5. Specifying CONS=*addr* or CON=*addr* in the IPL parameters portion of the Stand-Alone Program Loader (SAPL) window overrides any consoles specified with the OPERATOR_CONSOLES statement. For more information, see Using the Stand-Alone Program Loader in *z/VM: System Operation*.

Examples

1. The following example shows three possible device addresses for the system operator console.

```
Operator_Consoles  8e6  8e0  bc0          /* Operator Consoles in the
                                             Machine Room            */
```

PAGING Statement



Notes:

[1] You can specify the operands in any order, but you can specify an operand only once per PAGING statement.

Purpose

Use the PAGING statement to specify settings for the paging subsystem.

How to Specify

Include as many statements as needed; they are optional. You can place PAGING statements anywhere in the system configuration file. If you specify more than one statement with the same operand, the last operand definition overrides any previous specifications for that operand.

Operands

ALIAS ON
: indicates that system-attached HyperPAV aliases should be used for paging I/O. HyperPAV aliases allow concurrent I/O requests to the same physical volume by using alias devices. Aliases do not have associated storage, and can be bound dynamically to base volumes. Using HyperPAV can increase throughput by reducing queuing at the device level.

ALIAS OFF
: indicates that system-attached HyperPAV aliases should be not be used for paging I/O. This is the default.

HPF ON
: indicates that transport-mode channel program format should be used for paging I/O. With transport mode, less overhead occurs between the channel subsystem and the FICON adapter than with traditional command-mode I/O. This results in higher I/O rates and less CPU overhead. Specifying PAGING HPF ON is not recommended unless all paths to paging devices are High Performance FICON for System z® (zHPF) capable.

HPF OFF
: indicates that command-mode channel program format should be used for paging I/O. This is the default.

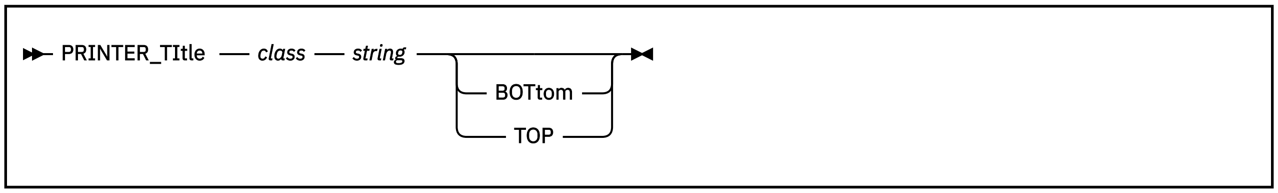
WARNing *nnn*%
: defines the allocation warning level for paging volumes. The *nnn* value is a decimal integer in the range of 0 to 100. If the number of allocated slots reaches this percentage of total slots on all paging volumes, message HCP401I is issued to alert the system operator.

  This warning level is in addition to the HCP401I 90% warning message issued by default. A WARNING value of 90% does not result in a second warning message. This value also serves as the default MAXPAGEFULL setting for a SET STORAGE command to ensure that a storage removal does not cause paging volumes to exceed this level of allocation. For more information, see SET STORAGE in *z/VM: CP Commands and Utilities Reference*.

Usage Notes

1. The paging subsystem's settings can be changed after IPL using the SET PAGING command.

PRINTER_TITLE Statement



Purpose

Use the PRINTER_TITLE statement to specify the printed output classes that are to contain classification titles. CP prints classification titles on the output separator page and, optionally, at the top or bottom of each page of output. You can specify a different classification title for each class of output (A through Z and 0 through 9).

How to Specify

Include as many statements as needed; they are optional. You can place PRINTER_TITLE statements anywhere in the system configuration file. If you specify more than one statement with the same class, the last statement overrides any previous specifications.

If you omit the PRINTER_TITLE statement, CP does not print any classification titles.

Operands

class
is a 1-character alphanumeric string specifying the spool file class.

string
is a classification title for this class. It may be up to 46 characters long, and must be enclosed in single or double quotation marks unless it is a single word entered with no imbedded blanks, in which case the entire word will be capitalized.

If you need to include a single quotation mark in the title, enclose the whole string with double quotation marks. If you need to include a double quotation mark in the title, enclose the whole string with single quotation marks. Also, you may use two consecutive double quotation marks ("") to represent a " character within a string delimited by double quotation marks. Similarly, you may use two consecutive single quotation marks ('') to represent a ' character within a string delimited by single quotation marks.

BOTtom
tells CP that this title is to be printed both on the separator page and at the bottom of each page of output.

TOP
tells CP that this title is to be printed both on the separator page and at the top of each page of output.

Note: If you do not specify the TOP or BOTTOM operands, CP only prints the title on the separator pages.

Usage Notes

1. CP inserts the title line at the top or bottom of each page. To do this, CP inserts a CCW to print one space, followed by the title line before or after each skip to channel 1 CCW issued by the operating system running in the virtual machine printing the file. Inserting the title lines this way means that:

   - CP inserts the titles as the file is created. Later, if you change the file to a different class, CP does not change the titles in the file. However, the title on the separator page always reflects the true class of the file.

- Inserting the title on the output page adds a line of output to the printed page. If the virtual machine application is counting lines, its count will be incorrect. Thus, the output listing may contain blank pages.
- If the operating system running in the virtual machine printing the file does not use skip to channel 1, the title lines are never printed.

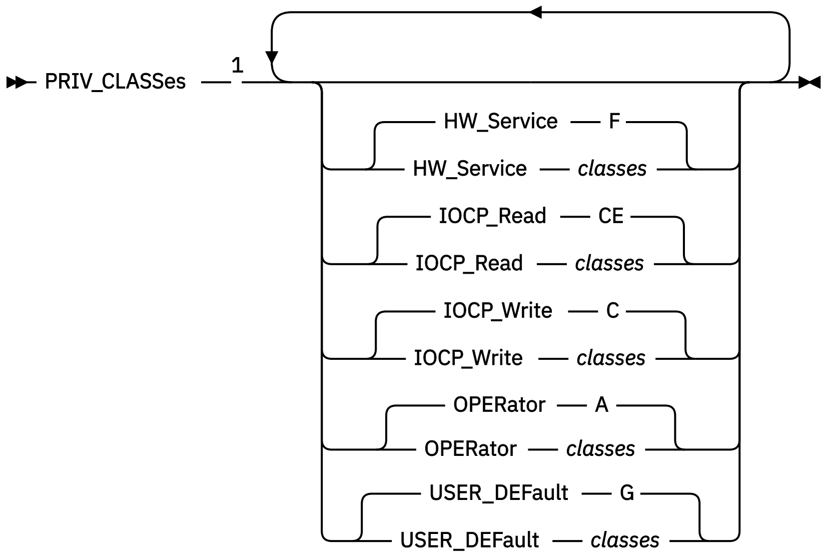
Examples

1. To define four printer classes as follows:
   - Class A has the title "No Security Rating" printed at the bottom of each page and on the separator pages
   - Class C has the title "COMPANY XYZ CONFIDENTIAL" printed at the top of each page and on the separator pages
   - Class Q has the title "It's not Confidential" printed on the separator pages
   - Class X has the title "Contents aren't Confidential" printed on the separator pages

   use the following PRINTER_TITLE statements:

```
/*-------------------------------------------------------------------*/
/*    Setting up Printer Titles                                      */
/*    -------------------------                                      */
/*
/*    Class A printer files get "No Security Rating" printed on the    */
/*            bottom of each page and on the separator pages           */
/*    Class C printer files get "COMPANY XYZ CONFIDENTIAL" printed at  */
/*            the top of each page and on the separator pages          */
/*    Class Q printer files get "It's Not Confidential" printed at     */
/*            the top of each page and on the separator pages          */
/*    Class X printer files get "Contents aren't Confidential" printed */
/*            at the top of each page and on the separator pages       */
/*-------------------------------------------------------------------*/
    Printer_Title  A  'No Security Rating'       Bottom

    Printer_Title  C  'COMPANY XYZ CONFIDENTIAL' Top

    Printer_Title  Q  'It''s Not Confidential'

    Printer_Title  X  "Contents aren't Confidential"
```

PRIV_CLASSES Statement



Notes:

[1] You must specify at least one of the following operands.

Purpose

Use the PRIV_CLASSES statement to change the privilege classes authorizing the following CP functions:

- Logging on as the primary system operator
- Intensive error recording
- Using the read function of the CP IOCP utility
- Using the write function of the CP IOCP utility
- Specifying the default user class.

How to Specify

Include as many statements as needed; they are optional. You can place PRIV_CLASSES statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

HW_Service *classes*
: tells CP which class or classes are authorized to perform intensive error recording. The variable *classes* is a 1- to 8-character alphanumeric string from A through Z and from 0 to 9. If omitted, the default is class F.

IOCP_Read *classes*
: tells CP which class or classes are authorized to use the read function of the IOCP utility. The variable *classes* is a 1- to 8-character alphanumeric string from A through Z and from 0 to 9. If omitted, the default is classes CE.

For more information, see IOCP in *z/VM: CP Commands and Utilities Reference*.

IOCP_Write *classes*
tells CP which class or classes are authorized to use the write function of the IOCP utility. The variable *classes* is a 1- to 8-character alphanumeric string from A through Z and from 0 to 9. If omitted, the default is class C.

For more information, see IOCP in *z/VM: CP Commands and Utilities Reference*.

OPERator *classes*
tells CP which class or classes are for the primary system operator. The variable *classes* is a 1- to 8-character alphanumeric string from A through Z and from 0 to 9. If omitted, the default is class A.

USER_DEFault *classes*
tells CP which class or classes to define for users who do not have a class specified in their virtual machine definitions. The variable *classes* is a 1- to 8-character alphanumeric string from A through Z and from 0 to 9. If omitted, the default is class G.

Examples

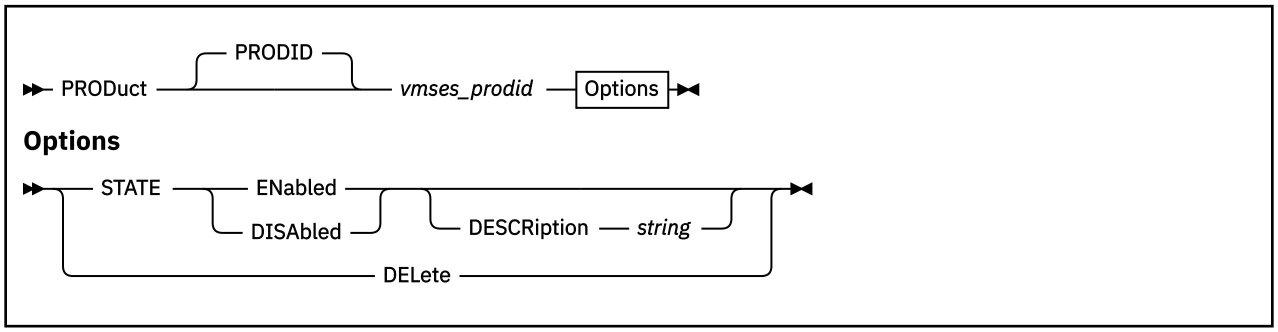
1. To authorize:

   - The primary system operator for class A, C, and E commands
   - Class C users to issue the IOCP READ and WRITE
   - Class D users to perform intensive error recording
   - All users without classes defined in their virtual machine definitions for class G commands

   use the following PRIV_CLASSES statement:

```
Priv_Classes,
     Operator      ACE,          /* Classes needed to be operator    */
     IOCP_Write    C,            /* Class needed for IOCDS read      */
     IOCP_Read     C,            /* Class needed for IOCDS write     */
     HW_Service    D,            /* Class needed for intensive error */
                                 /*      recording                   */
     User_Default  G             /* Default classes for general user */
```

PRODUCT Statement



Purpose

Use the PRODUCT statement to define a product or feature to the system.

How to Specify

Include as many statements as needed; they are optional. You can place PRODUCT statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

PRODID *vmses_prodid*
is the identifier used by VMSES/E to install and service the given product. *vmses_prodid* must be a 7 or 8 character identifier.

STATE ENABLED
identifies this product or feature as being licensed to run on this system.

When a product is defined with STATE ENABLED, it indicates that the product is authorized by the installation to run on this system. It does not mean that the product is installed or is being used.

STATE DISABLED
identifies this product or feature as not being licensed to run on this system. The product may or may not be physically installed on the system.

DESCRIPTION *string*
is a string of data that identifies additional information regarding the product.

string is a 1-255 character string (including any imbedded blanks and special characters) associated with the *vmses_prodid* entered. The string may be enclosed in single or double quotation marks. The text inside the quotation marks is not translated to uppercase. The string is stored unchanged and without the delimiting quotation marks.

If string is not delimited by quotation marks, it is translated to uppercase and multiple blanks are compressed to a single blank.

If you need to include a single quotation mark in the string, enclose the whole string with double quotation marks. If you need to include a double quotation mark in the string, enclose the whole string with single quotation marks. Also, you may use two consecutive double quotation marks ("") to represent a " character within a string delimited by double quotation marks. Similarly, you may use two consecutive single quotation marks ('') to represent a ' character within a string delimited by single quotation marks.

DELETE
removes from the system any definition information for the product ID, *vmses_prodid*.

Usage Notes

1. Each product is identified by a unique *vmses_prodid*. If a subsequent PRODUCT statement or SET PRODUCT command specifying an existing *vmses_prodid* is issued, it overwrites all of the related data including the description data.
2. The DELETE operand is included for those installations that may choose to have a common imbed for multiple systems with specific imbed statements following for each specific system.

Examples

1. To define and enable RSCS Function Level 720 (7VMRSC20) to the system with a comment that describes what the specified product ID identifies, use the following PRODUCT statement:

   ```
   PRODUCT 7VMRSC20 STATE ENABLED DESCRIPTION 'RSCS Networking Function
   Level 720'
   ```

2. To define PVM and explicitly disable PVM (5684100E) to the system with a comment that identifies why the product is disabled, use the following PRODUCT statement:

   ```
   PRODUCT 5684100E STATE DISABLED DESCRIPTION 'PVM FACILITY BASE - Disabled
   23-AUG-1997 after license expired.'
   ```

3. To delete the entry for PVM (5684100E), use the following PRODUCT statement:

   ```
   PRODUCT 5684100E DELETE
   ```

   A description is not allowed because all data for the product is discarded from the system when the DELETE operand is processed.
4. To define and enable DITTO (5688025A) to the system, use the following PRODUCT statement:

   ```
   PRODUCT 5688025A STATE ENABLED
   ```

   A description was not entered, therefore, there is no tie between the enablement statement for product ID 5688025A and DITTO.

If the above PRODUCT statements were processed in the given order, you would receive the following results after entering the `Q PRODUCT` command:

```
Q PRODUCT
Product  State    Description
5VMRSC30 Enabled  RSCS Networking Function Level 530
5688025A Enabled
5735FALQ Enabled  TCP/IP LEVEL 310 - TCP/IP     FEATURE (BASE)
```

RDEVICE Statement

Purpose

Use the RDEVICE statement to add to the system's definition of a set of real I/O devices. The sections below describe the syntax of the RDEVICE statements for the following devices:

- Dedicated advanced function printers. See "RDEVICE Statement (Advanced Function Printers)" on page 228.
- Card punches. See "RDEVICE Statement (Card Punches)" on page 230.
- Card readers. See "RDEVICE Statement (Card Readers)" on page 232.
- Communication controllers and line adapters. See "RDEVICE Statement (Communication Controllers)" on page 234.
- DASD. See "RDEVICE Statement (DASD)" on page 236.
- Graphic display devices. See "RDEVICE Statement (Graphic Display Devices)" on page 239.
- Impact printers. See "RDEVICE Statement (Impact Printers)" on page 243.
- Special devices. See "RDEVICE Statement (Special Devices)" on page 247.
- Tape units. See "RDEVICE Statement (Tape Units)" on page 249.
- Terminals. See "RDEVICE Statement (Terminals)" on page 251.
- Unsupported devices. See "RDEVICE Statement (Unsupported Devices)" on page 252.
- 3800 printers. See "RDEVICE Statement (3800 Printers)" on page 255.

How to Specify

Include as many statements as needed; they are optional. You can place RDEVICE statements anywhere in the system configuration file.

If you specify more than one statement with the real device number, CP uses the last statement. For example, if you specify:

```
Rdevice  0500  Type  3800  Model  1  AFP  yes  Chars  gf10
 .
 .
 .
Rdevice  0500  Type  AFP
```

CP defines a dedicated advanced function printer at real device number 0500, not a 3800 printer.

Usage Notes

1. If an IODF statement is defined in the system configuration file with the *osconfig* parameter specified, then the software I/O configuration will be controlled by HCD. In this case, the RDEVice statement is ignored if it is specified. For more information, see "IODF Statement" on page 181.
2. If you try to redefine the system residence or parm disk volume, CP will issue the following message:

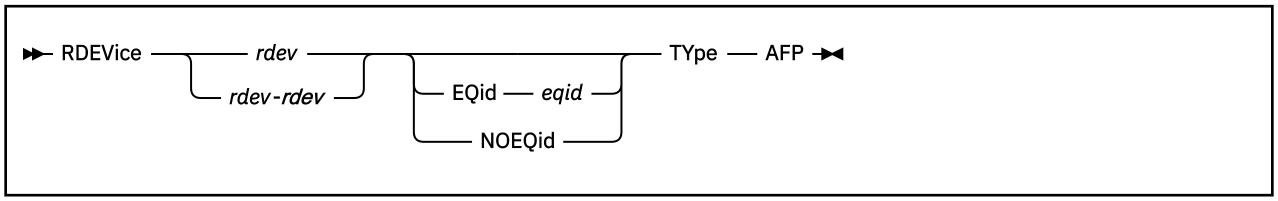
   HCP6751E

   Device *rdev* cannot be defined because it is the {SYSRES|parm disk} volume.
3. The RDEVICE statement cannot be used to change a current static device definition that exists in the HCPRIO ASSEMBLE file. When this redefinition is detected, you will see the following message:

   HCP6895E

   RDEV *rdev* cannot be changed or deleted because it is static device definition from HCPRIO

RDEVICE Statement (Advanced Function Printers)



Purpose

Use this RDEVICE statement to define one or more dedicated advanced function printers to CP. CP supports the following advanced function printers using Advanced Function Printing licensed programs:

3800 (Supported only for models 3, 6, and 8.)
3820 (Supported only when the 3820 printer is channel-attached.)
3825
3827
3828
3835
3900

Note: Printers defined as advanced function printers that follow the CCU specifications can be used only as dedicated devices.

Operands

rdev
rdev-rdev
is the real device number (or numbers) of the dedicated advanced function printer that you want defined. The maximum number of devices allowed within a range is 256. Each *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

EQid *eqid*
assigns the device equivalency ID (EQID) *eqid* to the RDEV. The *eqid* is a string of 1–8 alphanumeric characters. Note that for CTCA, FCP, HiperSocket, and OSA devices, this EQID must be unique or be shared only by other devices of the same type. For all other devices, the EQID must be unique across all devices on the system.

NOEQid
removes a previously assigned EQID from this RDEV. After the device is varied online, it reverts back to a system-generated EQID.

TYpe AFP
tells CP that the real device (or devices) that you are defining are dedicated advanced function printers (AFP) that follow the CCU specification. Note that printers defined as TYPE AFP can only be used as dedicated devices.

For more information about advanced function printers, see the *PSF/VM System Programming Guide*.

Usage Notes

1. If the specified EQID has already been assigned to one or more other devices, CP will issue the following message:

   HCP048E
   Specified EQID already assigned to a different device.

2. If the issued command attempts to assign an EQID to multiple devices and the input device type requires each device to have a unique EQID, CP will issue the following message:

HCP048E
EQID must be unique for the specified device class.

Examples

1. To define a dedicated advanced function printer at real device number 0110, use the following RDEVICE statement:

```
Rdevice  0110  Type  AFP
```

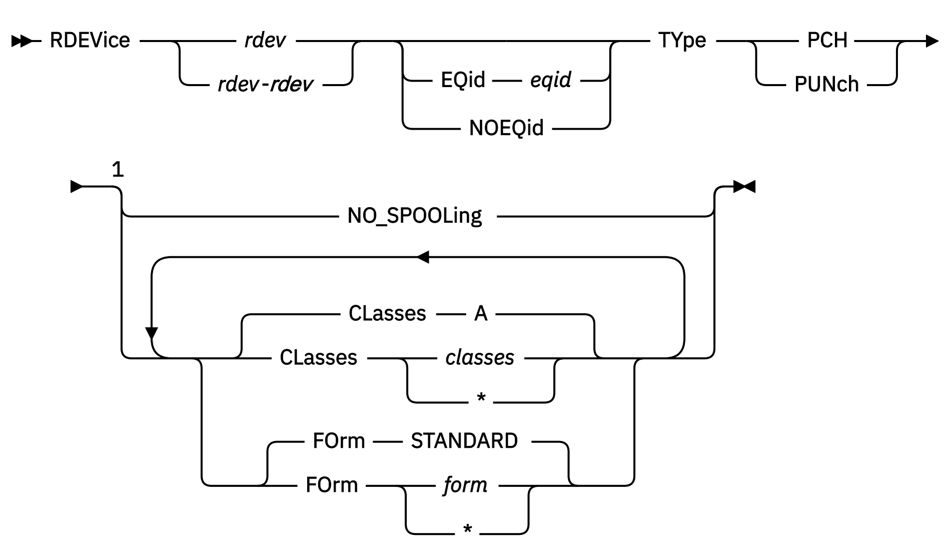
2. To define three dedicated advanced function printers at real device numbers 0111, 0112, and 0113, use one of the following RDEVICE statements:

```
Rdevice  0111-0113       Type  AFP
```

3. To define three dedicated advanced function printers at real device numbers E20, F10, and 1F00, use the following RDEVICE statements:

```
Rdevice   e20  Type  AFP
Rdevice   f10  Type  AFP
Rdevice  1f00  Type  AFP
```

RDEVICE Statement (Card Punches)



Notes:

[1] You must specify at least one of the following operands.

Purpose

Use this RDEVICE statement to define one or more card punches to CP. CP supports the following card punch:

3525 (Supported only for punching; the printing features are not supported.)

Operands

rdev
rdev-rdev
: is the real device number (or numbers) of the card punch that you want defined. The maximum number of devices allowed within a range is 256. Each *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

EQid *eqid*
: assigns the device equivalency ID (EQID) *eqid* to the RDEV. The *eqid* is a string of 1–8 alphanumeric characters. Note that for CTCA, FCP, HiperSocket, and OSA devices, this EQID must be unique or be shared only by other devices of the same type. For all other devices, the EQID must be unique across all devices on the system.

NOEQid
: removes a previously assigned EQID from this RDEV. After the device is varied online, it reverts back to a system-generated EQID.

TYpe PCH
TYpe PUNch
: tells CP that the real device (or devices) that you are defining are card punches. Note that CP only supports the 3525 for punching; the printing features are not supported.

NO_SPOOLing
: tells CP not to use the card punch for spooling.

CLasses *classes*
CLasses *
defines the output spooling class or classes that the card punch may process. Each class is one alphanumeric character and you can specify up to eight classes. If you specify more than one class, do not include any blanks between classes (for example, CLASSES ABC). If omitted, the default is A. If specified as an asterisk (*), the card punch can process any file, regardless of class.

To change the spooling class without having to re-IPL, use the CP START UR command. For more information, see START UR in *z/VM: CP Commands and Utilities Reference*.

FOrm *form*
FOrm *
defines the 1- to 8-character spooling form number that the card punch can process. When the operator starts the device after a cold start and does not specify a form, CP uses the form specified on this RDEVICE statement. If omitted, the default is STANDARD. If specified as an asterisk (*), the card punch can process any file, regardless of form number.

Usage Notes

1. If the specified EQID has already been assigned to one or more other devices, CP will issue the following message:

   HCP048E
   Specified EQID already assigned to a different device.

2. If the issued command attempts to assign an EQID to multiple devices and the input device type requires each device to have a unique EQID, CP will issue the following message:

   HCP048E
   EQID must be unique for the specified device class.

Examples

1. To define a card punch at real device number 0120, use the following RDEVICE statement:

```
Rdevice  0120  Type  Punch
```

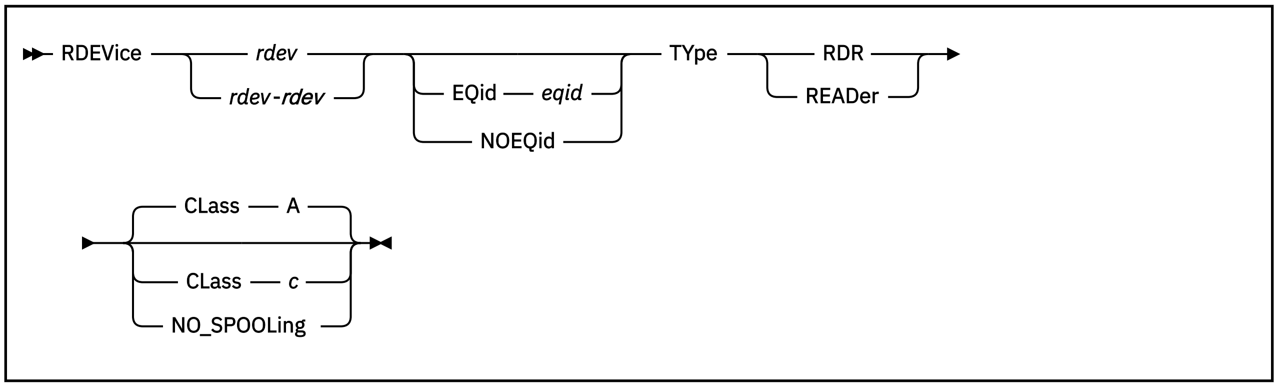
2. To define three card punches at device numbers 0121, 0122, and 0123, and to indicate that they can process any file regardless of form number, use one of the following RDEVICE statements:

```
Rdevice  0121-0123       Type  Punch  Form *
```

3. To define a card punch at device number 124 that processes class A files and four card punches that process files with spooling form number XYZ and class Z, use the following RDEVICE statements:

```
Rdevice  124      Type  Punch             Class  a
Rdevice  125-128  Type  Punch  Form  xyz  Class  z
```

RDEVICE Statement (Card Readers)



Purpose

Use this RDEVICE statement to define one or more real card readers to CP. CP supports the following card readers:

3505

Operands

rdev
rdev-rdev
is the real device number (or numbers) of the card reader that you want defined. The maximum number of devices allowed within a range is 256. Each *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

EQid *eqid*
assigns the device equivalency ID (EQID) *eqid* to the RDEV. The *eqid* is a string of 1–8 alphanumeric characters. Note that for CTCA, FCP, HiperSocket, and OSA devices, this EQID must be unique or be shared only by other devices of the same type. For all other devices, the EQID must be unique across all devices on the system.

NOEQid
removes a previously assigned EQID from this RDEV. After the device is varied online, it reverts back to a system-generated EQID.

TYpe RDR
TYpe READer
tells CP that the real device (or devices) that you are defining are card readers.

CLass *c*
defines the 1-character alphanumeric spooling class that the card reader assigns to spool files it creates. If omitted, the default is A.

To change the spooling class without having to re-IPL, use the CP START UR command. For more information, see START UR in *z/VM: CP Commands and Utilities Reference*.

NO_SPOOLing
tells CP not to use the card reader for spooling.

Usage Notes

1. If the specified EQID has already been assigned to one or more other devices, CP will issue the following message:

   HCP048E
   Specified EQID already assigned to a different device.

2. If the issued command attempts to assign an EQID to multiple devices and the input device type requires each device to have a unique EQID, CP will issue the following message:

   HCP048E

   EQID must be unique for the specified device class.

Examples

1. To define a card reader at device number 0130, use the following RDEVICE statement:

   ```
   Rdevice  0130  Type  Reader
   ```

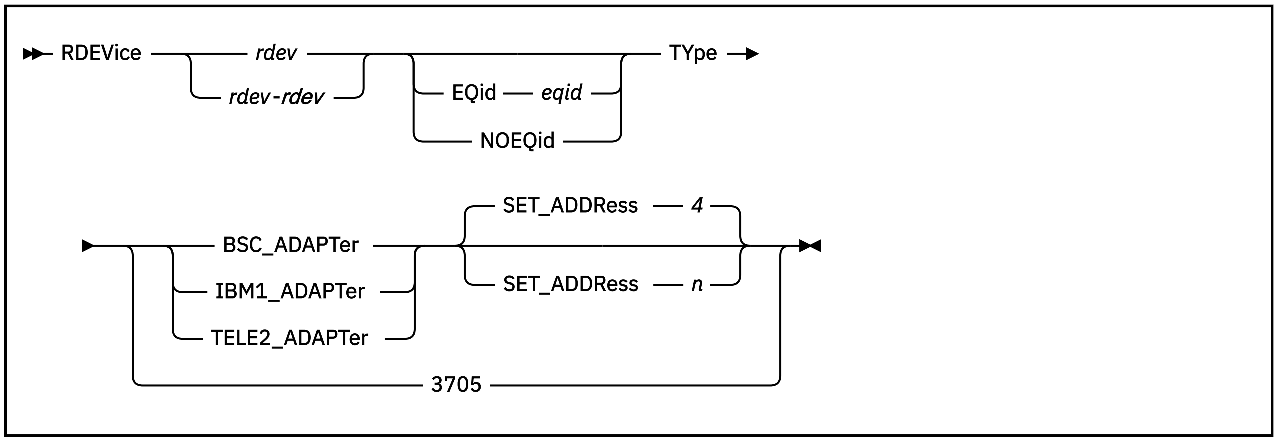
2. To define three card readers at device numbers 0131, 0132, and 0133, and to indicate that CP should not use them for spooling, use one of the following RDEVICE statements:

   ```
   Rdevice  0131-0133       Type  Reader  No_Spooling
   ```

3. To define a card reader at device number 134 that creates class W spool files, use the following RDEVICE statement:

   ```
   Rdevice  134  Type  Reader  Class  w
   ```

RDEVICE Statement (Communication Controllers)



Purpose

Use this RDEVICE statement to define one or more real communication controllers or line adapters to CP. CP supports the 3745 communication controllers and line adapters.

Note: CP supports the BSC adapter as dedicated only.

Operands

rdev
rdev-rdev
: is the real device number (or numbers) of the communication controller or line adapter that you want defined. The maximum number of devices allowed within a range is 256. Each *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

EQid *eqid*
: assigns the device equivalency ID (EQID) *eqid* to the RDEV. The *eqid* is a string of 1–8 alphanumeric characters. Note that for CTCA, FCP, HiperSocket, and OSA devices, this EQID must be unique or be shared only by other devices of the same type. For all other devices, the EQID must be unique across all devices on the system.

NOEQid
: removes a previously assigned EQID from this RDEV. After the device is varied online, it reverts back to a system-generated EQID.

TYpe
: tells CP that the real devices you are defining are:

  BSC_ADAPTer
  : specifies an IBM Binary Synchronous Terminal Control Type II for a 3745. (The BSC adapter is supported as dedicated-only.)

  IBM1_ADAPTer
  : specifies that an IBM Terminal Adapter Type I attaches a 2741 to a 3745.

  TELE2_ADAPTer
  : specifies that a CPT-TWX (Models 33/35) Terminal, 3101, 3151, 3161, 3162, or 3163 attaches to a Line Interface Base Type I in a 3745.

  SET_ADDRess *n*
  : tells CP which set address (SAD) command to enter for a telecommunication line attached to a control unit. The variable *n* is a 1-character decimal number between 0 and 4, with a default of 4. The variable *n* can be:

0
: tells CP to enter a SADZERO command.

1
: tells CP to enter a SADONE command.

2
: tells CP to enter a SADTWO command.

3
: tells CP to enter a SADTHREE command.

4
: (the default) tells CP not to enter a SAD command.

3705
: defines the base address used to load the 3745 communications controller.

Usage Notes

1. For all emulated lines running EP or NCP (PEP), you must have an RDEVICE statement for every line and the lines must be in the DEVICES NOT_SENSED list.

   If you do not want to use RDEVICE statements in your system configuration file, you can use the CP SET RDEVICE command. However, the emulated lines must still be in the DEVICES NOT_SENSED list. If you fail to do this, the lines may not come online and you will not be able to correct this situation without IPLing the system again.
2. Because emulator lines cannot be sensed dynamically, you should define all EP or NCP (PEP) emulator lines using either an RDEVICE statement or the CP SET RDEVICE command. The valid line types are:
   - BSC_ADAPTER
   - IBM1_ADAPTER
   - TELE2_ADAPTER
3. If the specified EQID has already been assigned to one or more other devices, CP will issue the following message:

   HCP048E
   Specified EQID already assigned to a different device.
4. If the issued command attempts to assign an EQID to multiple devices and the input device type requires each device to have a unique EQID, CP will issue the following message:

   HCP048E
   EQID must be unique for the specified device class.

Examples

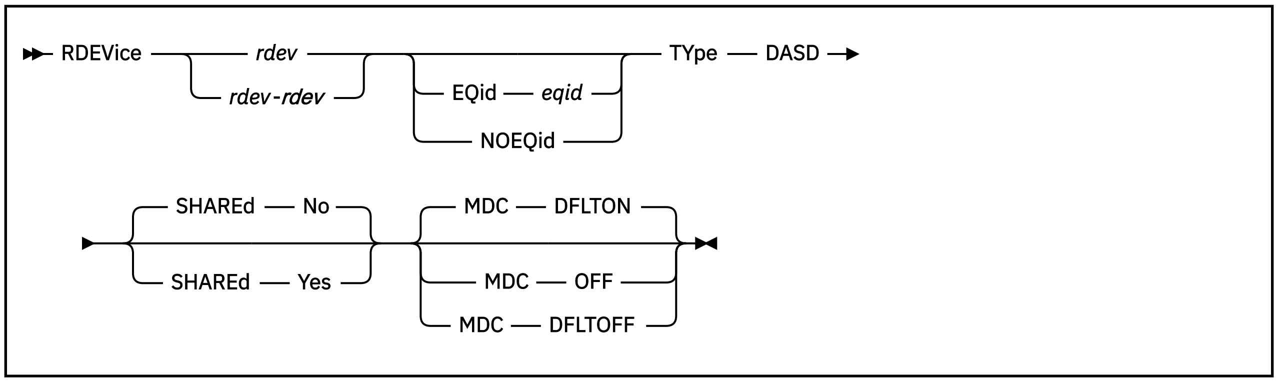
1. To define an IBM Terminal Adapter Type 1 at device number 01F0, use the following RDEVICE statement:

   ```
   Rdevice  01F0  Type  IBM1_Adapter
   ```
2. To define three BSC Terminal Control Type II adapters at device numbers 01F1, 01F2, and 01F3, use one of the following RDEVICE statements:

   ```
   Rdevice  01F1-01F3  Type  BSC_Adapter
   ```
3. To define six 3161, 3162, and 3163 lines at device addresses 030 through 035, use the following RDEVICE statement:

   ```
   Rdevice  030-035  Type  TELE2_Adapter        /* 316x ASCII lines     */
   ```

RDEVICE Statement (DASD)



Purpose

Use this RDEVICE statement to define one or more real direct access storage devices (DASD) to CP. CP supports the following DASD:

3390
9336

These DASD types do respond to the sense ID request instruction; you need to specify RDEVICE statements for them only if you want to specify the SHARED YES, the MDC OFF, or the MDC DFLTOFF options.

A Note About the 3850 Mass Storage System
VM/ESA release 1.1 was the last release to support the IBM 3850 Mass Storage System. Therefore, the system configuration file does not support the 3850. When devices are no longer supported, you may find that the existing code has not be removed from the product. If you have a 3850, you may still be able to use it by coding an RDEVICE macroinstruction in the HCPRIO ASSEMBLE file. For information about the RDEVICE macroinstruction, see the z/VM 6.2 edition of this document.

Operands

rdev
rdev-rdev

is the real device number (or numbers) of the DASD that you want defined. A real device defined in the active configuration can be entered as a 4-digit hexadecimal number between X'0000' and X'FFFF'. A real device defined in the active or standby configuration can be a 5-digit hexadecimal number between X'00000' and X'3FFFF' with the leading digit specifying the subchannel set where the device is defined. A range of devices cannot span subchannel sets. The maximum number of devices allowed within a range is 256.

EQid *eqid*

assigns the device equivalency ID (EQID) *eqid* to the RDEV. The *eqid* is a string of 1–8 alphanumeric characters. Note that for CTCA, FCP, HiperSocket, and OSA devices, this EQID must be unique or be shared only by other devices of the same type. For all other devices, the EQID must be unique across all devices on the system.

NOEQid

removes a previously assigned EQID from this RDEV. After the device is varied online, it reverts back to a system-generated EQID.

TYpe DASD

tells CP that the real device (or devices) that you are defining are DASD.

SHAREd

tells CP whether you want the device or devices to be shared concurrently among multiple real systems.

In an SSI cluster, this setting specifies whether the device or devices are to be shared outside the SSI cluster. For more information, see usage note "4" on page 237.

No

(the default) tells CP not to share the device or devices among multiple real systems.

Yes

tells CP to share the device or devices among multiple real systems. This setting tells CP not to cache data on the real device in the minidisk cache. For more information on sharing DASDs, see Chapter 25, "DASD Sharing," on page 667.

MDC

tells CP whether you want the device or devices to be cached in the minidisk cache.

DFLTON

(the default if the SHARED option is not specified or the SHARED NO option is specified) tells CP to cache data on the real device in the minidisk cache except for minidisks that have caching specifically disabled by the MINIOPT NOMDC directory statement or the SET MDCACHE MDISK OFF command.

OFF

(the default if the SHARED YES option is specified) tells CP not to cache data on the real device in the minidisk cache.

DFLTOFF

tells CP to not cache data on the real device in the minidisk cache except for minidisks that have caching specifically enabled by the MINIOPT MDC directory statement or the SET MDCACHE MDISK ON command.

Usage Notes

1. Not all DASD types are eligible for caching in the minidisk cache. If you specify MDC DFLTON or MDC DFLTOFF for a device that is not cache eligible, data on that device will not be cached.
2. If the specified EQID has already been assigned to one or more other devices, CP will issue the following message:

   HCP048E

   Specified EQID already assigned to a different device.
3. If the issued command attempts to assign an EQID to multiple devices and the input device type requires each device to have a unique EQID, CP will issue the following message:

   HCP048E

   EQID must be unique for the specified device class.
4. Table 12 on page 238 shows how the RDEV SHARED setting affects reserve/release, MDCACHE, and link in an SSI cluster.

Table 12. Effects of RDEV SHARED in an SSI cluster

CONDITION	RESERVE/RELEASE	MDCACHE	LINK
RDEV SHARED YES	Real reserve/release issued for fullpack minidisks with V mode suffix in an SSI cluster and a non-SSI system. Virtual reserve/release issued for non-fullpack minidisks in an SSI cluster and a non-SSI system, but it affects only the system where the reserve/release is issued.	Minidisks not eligible for cache.	Always do link conflict checking with all systems in the SSI cluster, regardless of the SHARED setting.
RDEV SHARED NO	Real reserve/release issued for fullpack minidisks with V mode suffix in an SSI cluster. Virtual reserve/release issued for fullpack minidisks with V mode suffix in a non-SSI system, and for non-fullpack minidisks with V mode suffix in an SSI cluster and a non-SSI system, but it affects only the system where the reserve/release is issued.	Minidisks are eligible for cache. Cache will be automatically turned OFF/ON when write links are granted/released on other systems in the SSI cluster.	Always do link conflict checking with all systems in the SSI cluster, regardless of the SHARED setting.

Examples

1. To define a shared DASD with minidisk caching off at device number 0410, use the following RDEVICE statement:

```
Rdevice  0410  Type  DASD  Shared  yes
```

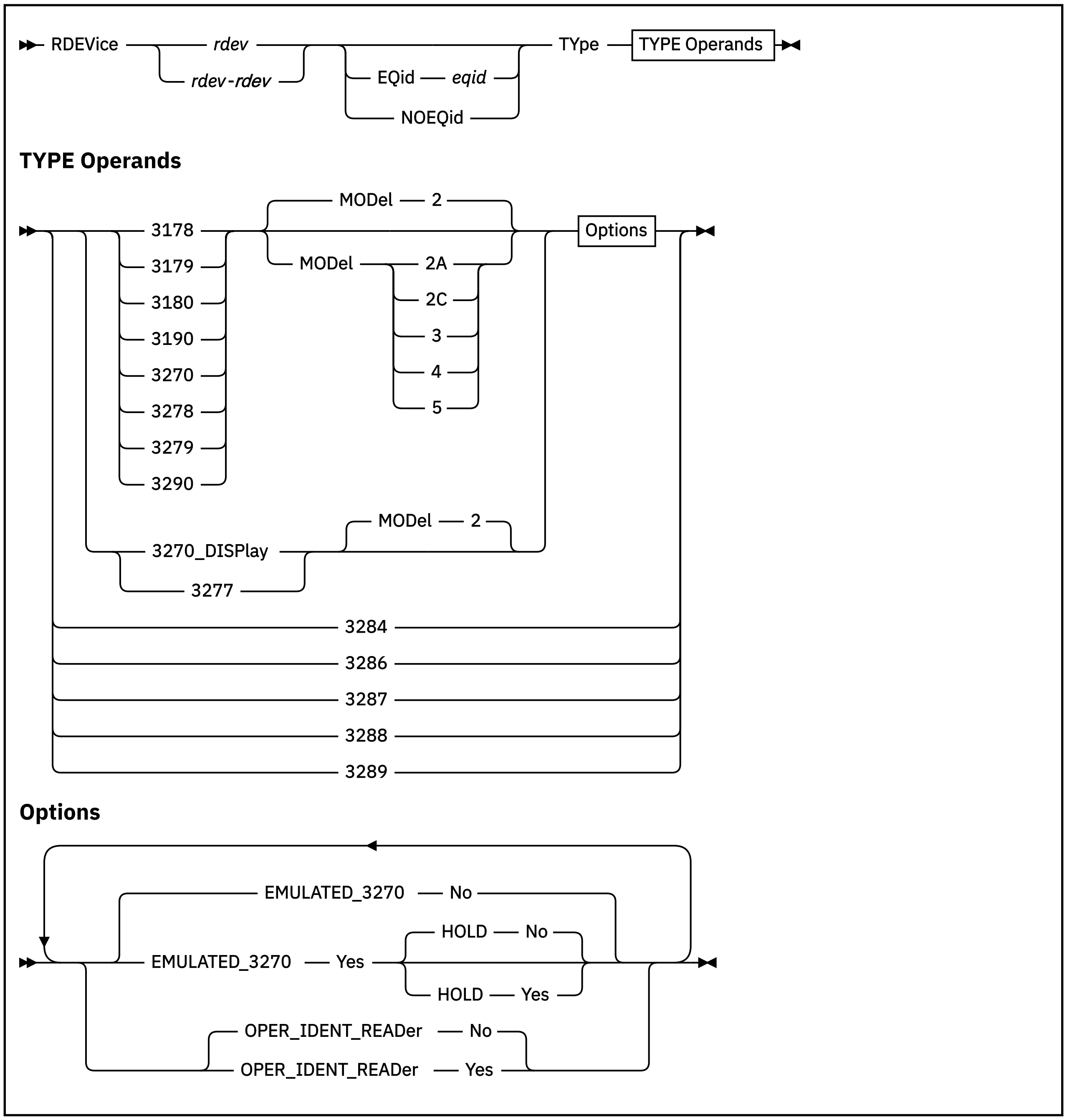
2. To define 64 shared DASDs with minidisk caching off at device numbers 100 through 13F and 32 shared DASDs at device numbers 140 through 15F, use the following RDEVICE statements:

```
Rdevice  100-13f  Type  Dasd  Shared  yes   /* 3390 Dasd shared with */
                                            /*  other 2064 …       */
Rdevice  140-15f  Type  Dasd  Shared  yes   /* 3380 Dasd shared with */
                                            /*  2064 …             */
```

3. To define a nonshared DASD with minidisk caching on, no RDEVICE statement is necessary.
4. To define a nonshared DASD at device number 500 with minidisk caching enabled on a device only for those minidisks that have specifically enabled caching, use the following RDEVICE statement:

```
Rdevice  500  Type  Dasd  MDC DFLTOFF
```

RDEVICE Statement (Graphic Display Devices)



Purpose

Use this RDEVICE statement to define one or more graphic display devices to CP.

Operands

rdev
rdev-rdev
is the real device number (or numbers) of the graphic display devices that you want defined. The maximum number of devices allowed within a range is 256. Each *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

EQid *eqid*
assigns the device equivalency ID (EQID) *eqid* to the RDEV. The *eqid* is a string of 1–8 alphanumeric characters. Note that for CTCA, FCP, HiperSocket, and OSA devices, this EQID must be unique or be shared only by other devices of the same type. For all other devices, the EQID must be unique across all devices on the system.

NOEQid
removes a previously assigned EQID from this RDEV. After the device is varied online, it reverts back to a system-generated EQID.

TYpe 3178, 3179, 3180, 3190, 3270, 3278, 3279, or 3290
tells CP that the real device (or devices) that you are defining are graphic display devices. CP can determine dynamically all of these graphic display devices during initialization or when you vary on the device. You only need to specify these graphic display devices if you want to specify the extended attributes.

MODel 2, 2A, 2C, 3, 4, or 5
is the model number of the graphic display device. If omitted, the default is Model 2.

TYpe 3270_DISPlay or 3277
tells CP that the real device (or devices) that you are defining are graphic display devices. CP can determine dynamically all of the 3270 graphic display devices during initialization or when you vary on the device. You only need to specify the 3270 graphic display devices if you want to specify the extended attributes.

Note: The 3270-PC is supported in control-unit terminal mode and distributed-function terminal (DTF) mode when defined as a 3270.

You must specify the 3277 display because its device type cannot be determined.

MODel 2
is the model number of the graphic display device. If omitted, the default is Model 2.

TYpe 3284, 3286, 3287, 3288, or 3289
tells CP that the real device (or devices) that you are defining are graphic printer devices. CP can determine dynamically all of the graphic printer devices during initialization or when you vary on the device; therefore, you do not need to specify them.

Note: CP cannot sense these devices when they are attached either to a 3174 or a 3274 control unit. If these devices are attached to a 3174 or a 3274 control unit, each device must be defined in the system configuration file using the RDEVICE statement.

EMULATED_3270 No
(the default) tells CP that this device (or devices) is not a TTY ASCII display terminal connected to the system through a 7171 ASCII DACU or ASCII Subsystem (an emulated 3270).

EMULATED_3270 Yes
tells CP that this device (or devices) is a TTY ASCII display terminal connected to the system through a 7171 ASCII DACU or ASCII Subsystem (an emulated 3270).

Note: The 3101 must be connected to a 7171 Control Unit. You must also specify EMULATED_3270 YES.

HOLD No
(the default) tells CP not to keep the display terminal telecommunications connection to the 7171 ASCII DACU or ASCII Subsystem after the virtual machine logs off or disconnects.

HOLD Yes
tells CP keep the display terminal telecommunications connection to the 7171 ASCII DACU or ASCII Subsystem after the virtual machine logs off or disconnects.

OPER_IDENT_READer No
(the default) tells CP that the 327*x* display does not have an operator identification card reader.

OPER_IDENT_READer Yes
tells CP that the 327*x* display has an operator identification card reader. If you specify OPER_IDENT_READER YES, the virtual machine operator can gain access to the system (log on) only

by inserting a magnetically encoded card. Use of a badge reader is optional for each user ID and not required for any of them. Use of a badge reader cannot replace entering a correct password but may follow a correct password as an additional security measure. If you do not want to have a card reader authorize access, do not specify OPER_IDENT_READER YES, or specify OPER_IDENT_READER NO.

Usage Notes

1. For a complete list of all graphic displays and printers that are supported for this release, see Chapter 4, “Defining I/O Devices,” on page 27.
2. You must ensure that the device numbers specified on the statement correspond to the device numbers required for the category of display terminal in the controller.
3. Refer to the *7171 Device Attachment Control Unit Reference Manual and Programming Guide* to determine which type of device (TYPE) to specify for a TTY ASCII terminal attached to a 7171 DACU or ASCII Subsystem.
4. The 3178, 3179, 3180, 3190, 3278, 3279, 3284, 3486, 3287, 3288, 3289, and 3290 devices are defined primarily for use with second-level systems. In a second-level system, you can define a printer or display to be at a particular device number and then you can do a first-level define of the device number. If you define the device in the system configuration file, the second-level system brings that device online immediately rather than having you define a device number first-level, issue the CP SET RDEVICE command second-level, and vary on the device second-level.
5. If the specified EQID has already been assigned to one or more other devices, CP will issue the following message:

   HCP048E
   Specified EQID already assigned to a different device.
6. If the issued command attempts to assign an EQID to multiple devices and the input device type requires each device to have a unique EQID, CP will issue the following message:

   HCP048E
   EQID must be unique for the specified device class.

Examples

1. To define a 3277 display at device number 0210, use the following RDEVICE statement:

   ```
   Rdevice 0210 Type 3277
   ```
2. To define three 3270 displays at device numbers 0211, 0212, and 0213, and to indicate that users should not be able to log on to them without an operator identification card, use the following RDEVICE statement:

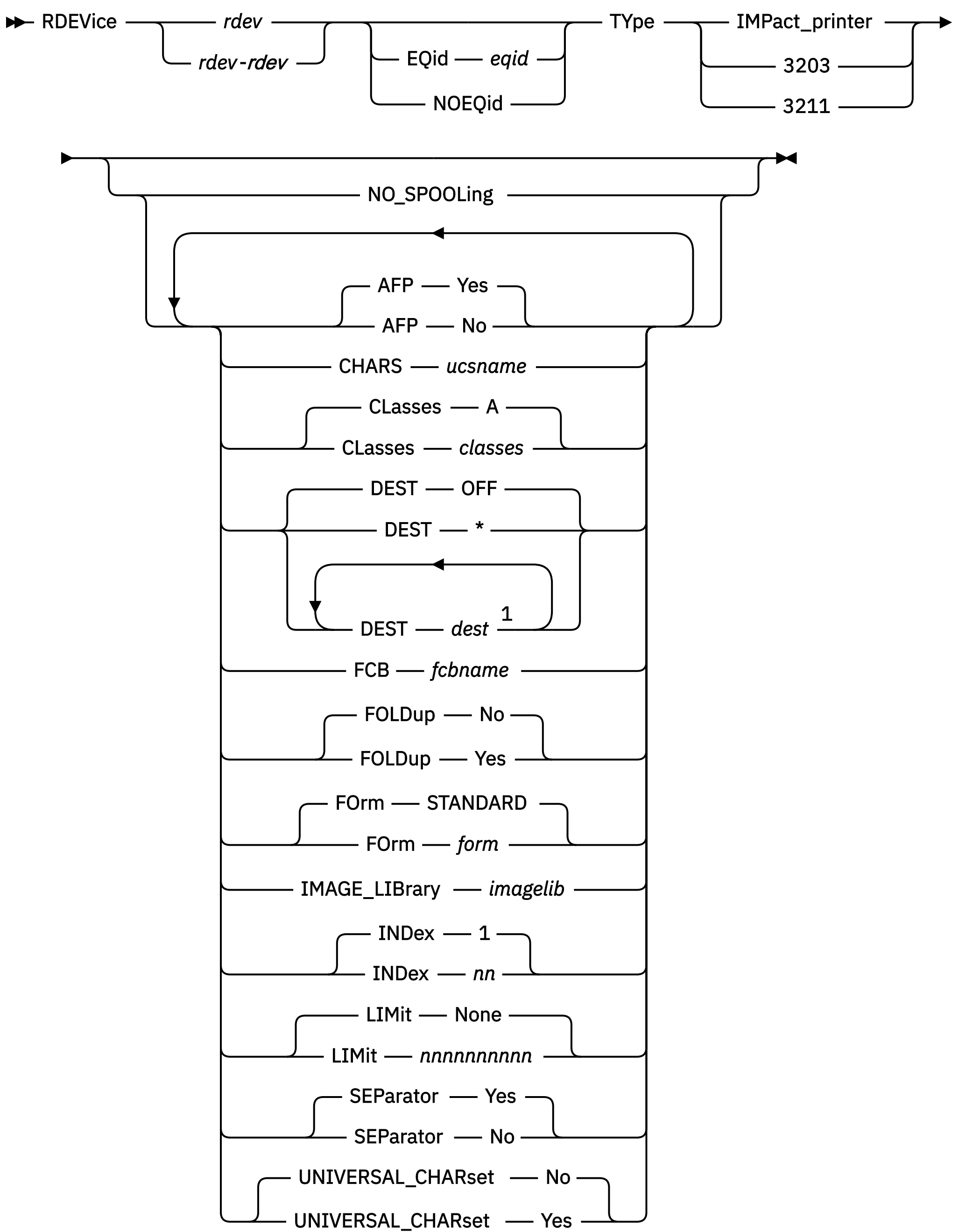
   ```
   Rdevice 0211-0213 Type 3270_Display Oper_Ident_Reader yes
   ```
3. To define:

   - 48 3270 displays at device numbers 800 through 82F
   - 16 3277 displays at device numbers 830 through 83F
   - 31 3278 displays at device numbers 970 through 98E
   - One 3287 display at device number 98F
   - One 3290 display at device number 1010
   - 15 3190 displays at device numbers 1011 through 101F
   - 61 3278 displays at device numbers 200 through 23C, which are TTY ASCII display terminals connected to the system through 7171 ASCII DACU or ASCII Subsystems, and which retain their connections after logging off or disconnecting

   use the following RDEVICE statements:

```
Rdevice 800-82f   Type 3270_Display   /* 3174 controller             */
                                      /* ... with 32 local ports     */
Rdevice 830-83f   Type 3277           /* 3272 controller             */
                                      /* ... with 16 3277 displays   */
Rdevice 970-98e   Type 3278           /* 3274 controller             */
                                      /* ... with 31 3178s and 3278s */
Rdevice 98f       Type 3287           /* 3287 printer                */
                                      /* ... hung off 3274 port 32   */
Rdevice 1010      Type 3290           /* 3290 information display on */
                                      /* ... port 0 of 3174 control  */
                                      /* ... unit                    */
Rdevice 1011-101f Type 3190           /* 15 3190s on                 */
                                      /* ... 3174 control unit       */
Rdevice 200-23c   Type 3278,          /* 7171 ports looking like     */
                  Emulated_3270 yes,  /* ... 3278s to CP             */
                  Hold yes
```

RDEVICE Statement (Impact Printers)



Notes:

[1] You cannot specify more than four output destinations.

Purpose

Use this RDEVICE statement to define the following system-managed, channel-attached impact printers to CP:

3203 (Only the 3203 Model 5 is supported.)
3211 (3211 printers are only supported as emulated by other printers.)

3262
4245
4248
6262

Operands

rdev
rdev-rdev
is the real device number (or numbers) of the impact printer that you want defined. The maximum number of devices allowed within a range is 256. Each *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

EQid *eqid*
assigns the device equivalency ID (EQID) *eqid* to the RDEV. The *eqid* is a string of 1–8 alphanumeric characters. Note that for CTCA, FCP, HiperSocket, and OSA devices, this EQID must be unique or be shared only by other devices of the same type. For all other devices, the EQID must be unique across all devices on the system.

NOEQid
removes a previously assigned EQID from this RDEV. After the device is varied online, it reverts back to a system-generated EQID.

TYpe IMPact_printer
TYpe 3202
TYpe 3211
tells CP that the real device (or devices) that you are defining is one of the impact printers listed above.

NO_SPOOLing
tells CP not to use the impact printer for spooling.

AFP Yes
(the default) tells CP that the impact printer can process files with advanced function printer (AFP) characteristics, XABs, or X'5A' CCWs.

AFP No
tells CP that the impact printer cannot process files with advanced function printer (AFP) characteristics, XABs, or X'5A' CCWs.

CHARS *ucsname*
For a printer with the UNIVERSAL_CHARSET feature, CHARS *ucsname* specifies the 1- to 4-character suffix of the name of the default universal character set (UCS) buffer image. The variable *ucsname* must correspond to one of the UCS images stored in the image library. For example, if you specify:

```
Rdevice 0003 Type 3203 Chars an
```

The image library must contain a member named 3203AN.

CLasses *classes*
defines the output spooling class or classes that the impact printer can print. Each class is 1 alphanumeric character and you can specify up to 8 classes. If you specify more than one class, do not include any blanks between the classes (for example, CLASSES ABC). If omitted, the default is A. If specified as an asterisk (*), the printer can process any file, regardless of class.

To change the spooling classes without having to re-IPL, use the CP START UR command. For more information, see START UR in *z/VM: CP Commands and Utilities Reference*.

DEST OFF
DEST *
DEST *dest*
specifies the output destination values the printer can print. Specify DEST in one of the following ways:

- By default — if you omit the DEST operand, DEST OFF is assumed.

- You can specify as many as four destination values by entering four different DEST operands, as follows:

  ```
  Dest printer1 Dest printer2 Dest printer3 Dest printer4
  ```

- DEST * specifies that the printer should process files regardless of destination.

To change the spooling destinations without having to re-IPL, use the CP START UR command. For more information, see START UR in *z/VM: CP Commands and Utilities Reference*.

FCB *fcbname*
specifies the 1- to 4-character alphanumeric suffix of the name of the default forms control buffer (FCB) CP should use after a cold start or force start. This must correspond to one of the FCB images added to an image library. For example, if you specify:

```
Rdevice 0003 Type 3203 FCB fcb8
```

The image library must contain a member named 3203FCB8.

FOLDup No
(the default) tells CP not to fold (translate) lowercase into uppercase.

FOLDup Yes
tells CP to fold (translate) lowercase into uppercase.

FOrm STANDARD
FOrm *form*
is the current spooling form number that the printer can process. This form is the default operator form for the real printer when the operator starts the device after a cold start without specifying a form. Specify FORM in one of the following ways:

- FORM *form* (*form* is a 1- to 8-character operator form number for the files the printer can process)
- FORM * indicates that the printer can process files regardless of form number
- FORM STANDARD indicates that the type of paper to be mounted on the printer is the type of paper that the installation has assigned the name STANDARD. Each installation establishes its own set of form names, assigns those form names to types of paper, and informs the operations staff and end users of the correlation between form names and types of paper.

If omitted, the default is STANDARD.

IMAGE_LIBrary *imagelib*
tells CP which image library to use after a cold start. If omitted, the default is IMAGE_LIBRARY *nnnn* (*nnnn* is the device type number).

INDex *nn*
specifies the position at which to start printing. The variable *nn* is a decimal number from 1 to 31. If omitted, the default is 1.

Note: The INDEX operand is only valid for 3211 printers. If you specify it for another impact printer, CP ignores the operand.

LIMit None
(the default) tells CP that this impact printer can print files with an unlimited number of records.

LIMit *nnnnnnnnnn*
tells CP that this impact printer limits the size of files it can print. The variable *nnnnnnnnnn* is a 1- to 10-digit decimal number that indicates the maximum number of records per file that this printer can print.

SEParator Yes
(the default) tells CP to print separator pages between output files.

SEParator No
tells CP not to print separator pages between output files.

UNIVERSAL_CHARset No
(the default) tells CP that this impact printer is not a universal character set printer.

UNIVERSAL_CHARset Yes
tells CP that this impact printer is a universal character set printer.

Note: Although UNIVERSAL_CHARset Yes is accepted for TYpe IMPact printer, it is only meaningful for TYpe 3203 and TYpe 3211 printers.

Usage Notes

1. If the specified EQID has already been assigned to one or more other devices, CP will issue the following message:

   HCP048E
   Specified EQID already assigned to a different device.

2. If the issued command attempts to assign an EQID to multiple devices and the input device type requires each device to have a unique EQID, CP will issue the following message:

   HCP048E
   EQID must be unique for the specified device class.

Examples

1. To define a 3203 printer at device number 0110 that prints only class A spool files (the default value), use the following RDEVICE statement:

```
Rdevice  0110  Type  Impact_Printer
```

2. To define a 3262 printer at device number 0110 that will be dedicated to a guest, use the following RDEVICE statement:

```
Rdevice  0110  Type  Impact_Printer  No_Spooling
```

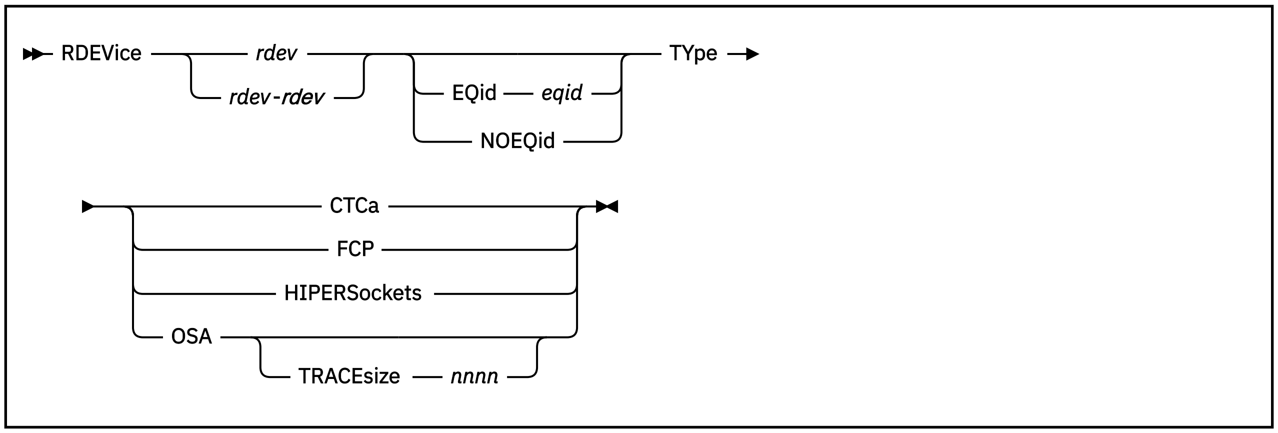
3. To define an impact printer at device number:

   - E0E that can print class 1, process "standard" forms, and use the normal output destination, forms control buffer FCB8, and image library 4245IMAG
   - 00E that can print class A files, use forms control buffer FCB8, and process FORM8 forms
   - 01E that can print class A files, use forms control buffer FCB8, and process FORM1 forms

   use the following RDEVICE statements:

```
Rdevice  e0e  Type  Impact_Printer,
              Class 1,
              Dest normal,
              FCB fcb8,
              Image_Library 4245IMAG,
              Form standard

Rdevice  00e  Type 3203,
              Class a,
              FCB fcb8,
              Form form8

Rdevice  01e  Type 3211,             /* 4248 printer running in 3211 */
              Class a,               /* compatibility mode           */
              FCB fcb8,
              Form form1
```

RDEVICE Statement (Special Devices)



Purpose

Use this RDEVICE statement to define device addresses of certain types of devices to CP.

Operands

rdev
rdev-rdev
is the real device number (or numbers) of the devices that you want defined. The maximum number of devices allowed within a range is 256. Each *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

EQid *eqid*
assigns the device equivalency ID (EQID) *eqid* to the RDEV. The *eqid* is a string of 1–8 alphanumeric characters. Note that for CTCA, FCP, HiperSocket, and OSA devices, this EQID must be unique or be shared only by other devices of the same type. For all other devices, the EQID must be unique across all devices on the system.

NOEQid
removes a previously assigned EQID from this RDEV. After the device is varied online, it reverts back to a system-generated EQID.

TYpe CTCa
tells CP that the real device (or devices) that you are defining is a channel-to-channel device.

Note: Specify a 3737 Remote Channel-to-Channel Unit Model 2 as a CTCA.

TYpe FCP
tells CP that the real device (or devices) that you are defining is a Fiber-Channel-Protocol device.

TYpe HIPERSockets
tells CP that the real device (or devices) that you are defining is a HiperSockets device.

TYpe OSA
tells CP that the real device (or devices) that you are defining is an Open-Systems-Adapter device.

TRACEsize *nnnn*
specifies the number of pages to be allocated for an internal trace table to track trace events pertaining to EQDIO on this OSA device. The allowable range is 0-4095. The default value is 8, which increases the chance of capturing sufficient documentation in the event of an EQDIO problem. Increase the page size to capture a greater history of traces. The captured documentation assists IBM in diagnosing an EQDIO-related problem. Setting a value of 0 turns off this data capture function.

Note: The TRACESIZE parameter applies to OSA devices that operate in EQDIO mode; it does not apply to OSA devices that operate in QDIO mode.

Usage Notes

1. CP can sense some devices. Therefore you do not need to specify them in the system configuration file, unless you are running programs in the device that do not respond to sense ID requests.
2. If the specified EQID has already been assigned to a different class of devices, CP will issue the following message:

   HCP048E
   Specified EQID already assigned to a different class of devices.
3. If the specified EQID has already been assigned to one or more other devices, CP will issue the following message:

   HCP048E
   Specified EQID already assigned to a different device.

Examples

1. To define a channel-to-channel adapter at device number 0250, use the following RDEVICE statement:

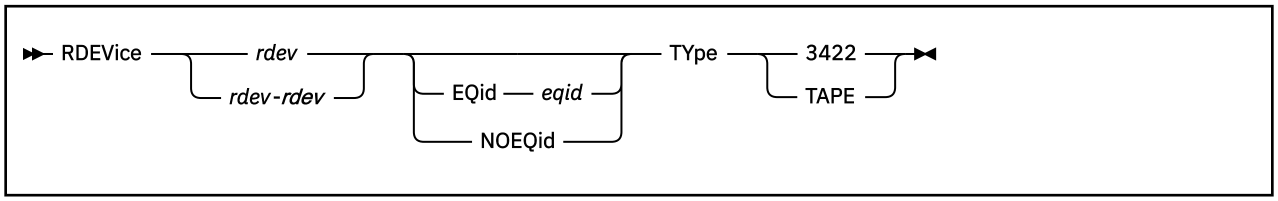
   ```
   Rdevice  0250  Type  CTCA
   ```
2. To define an Open-Systems-Adapter device at device number 0252, use the following RDEVICE statement:

   ```
   Rdevice  0252  Type  OSA
   ```
3. The following RDEVICE statement defines:

   - One channel-to-channel adapter at device address 250

   ```
   /*-----------------------------------------------------------------*/
   /* Real CTCA to 4381, 4381 owns CTCA                               */
   /*-----------------------------------------------------------------*/
      Rdevice  250  Type  CTCA
   ```

RDEVICE Statement (Tape Units)



Purpose

Use this RDEVICE statement to define one or more tape units to CP.

Operands

rdev
rdev-rdev

is the real device number (or numbers) of the tape unit that you want defined. The maximum number of devices allowed within a range is 256. Each *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

EQid *eqid*

assigns the device equivalency ID (EQID) *eqid* to the RDEV. The *eqid* is a string of 1–8 alphanumeric characters. Note that for CTCA, FCP, HiperSocket, and OSA devices, this EQID must be unique or be shared only by other devices of the same type. For all other devices, the EQID must be unique across all devices on the system.

NOEQid

removes a previously assigned EQID from this RDEV. After the device is varied online, it reverts back to a system-generated EQID.

TYpe 3422

tells CP that the real device (or devices) that you are defining is an IBM 3422 magnetic tape subsystem.

TYpe TAPE

specifies an IBM tape unit that can be dynamically sensed by CP (such as an IBM 3424, 3480, 3490 or 3590 tape unit).

Usage Notes

1. CP can dynamically sense most tape units at IPL time (such as the IBM 3480 and 3490 tape units), so there is no need for an RDEVICE statement for those units. However, if you decide to define one of these type of tape units to CP without it being connected to the system at IPL time, you can use the RDEVICE statement with the TYPE TAPE operand.
2. If the specified EQID has already been assigned to one or more other devices, CP will issue the following message:

   HCP048E

   Specified EQID already assigned to a different device.
3. If the issued command attempts to assign an EQID to multiple devices and the input device type requires each device to have a unique EQID, CP will issue the following message:

   HCP048E

   EQID must be unique for the specified device class.

Examples

1. To define a 3422 tape unit at device number 0310, use the following RDEVICE statement:

```
Rdevice  0310  Type  3422
```

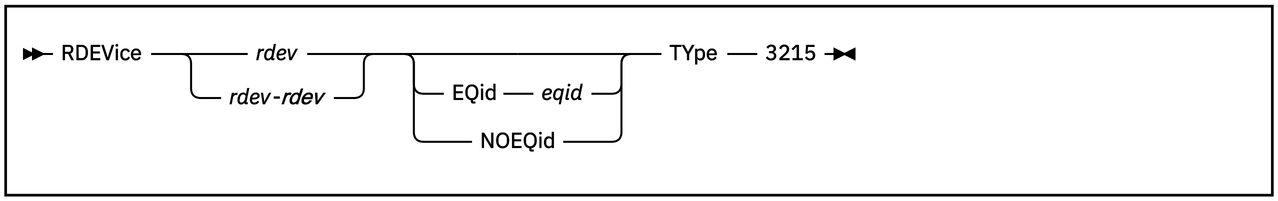
2. To define eight 3422 tape drives at device numbers F10 through F17, use the following RDEVICE statements:

```
/*---------------------------------------------------------------------*/
/* We do not have to specify RDEVICE statements for our:              */
/*                                                                     */
/*    3490s at 100-10F because they answer with their device type     */
/*          when asked (sensed).                                       */
/*                                                                     */
/*    3480s at 500-50F because they answer with their device type     */
/*          when asked (sensed).                                       */
/*                                                                     */
/*---------------------------------------------------------------------*/

/*-------------------------*/
/* Define our eight 3422s: */
/*-------------------------*/

Rdevice  f10-f17  Type  3422
```

3. To define a tape unit which can be dynamically sensed by CP at device number 0F20, use the following RDEVICE statement:

```
Rdevice 0F20 Type Tape
```

RDEVICE Statement (Terminals)



Purpose

Use this RDEVICE statement to define one or more terminals to CP.

Operands

rdev
rdev-rdev
is the real device number (or numbers) of the terminal(s) that you want defined. The maximum number of devices allowed within a range is 256. Each *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

EQid *eqid*
assigns the device equivalency ID (EQID) *eqid* to the RDEV. The *eqid* is a string of 1–8 alphanumeric characters. Note that for CTCA, FCP, HiperSocket, and OSA devices, this EQID must be unique or be shared only by other devices of the same type. For all other devices, the EQID must be unique across all devices on the system.

NOEQid
removes a previously assigned EQID from this RDEV. After the device is varied online, it reverts back to a system-generated EQID.

TYpe 3215
tells CP that the real device (or devices) that you are defining is a 3215 terminal.

Usage Notes

1. If the specified EQID has already been assigned to one or more other devices, CP will issue the following message:

   HCP048E
   Specified EQID already assigned to a different device.

2. If the issued command attempts to assign an EQID to multiple devices and the input device type requires each device to have a unique EQID, CP will issue the following message:

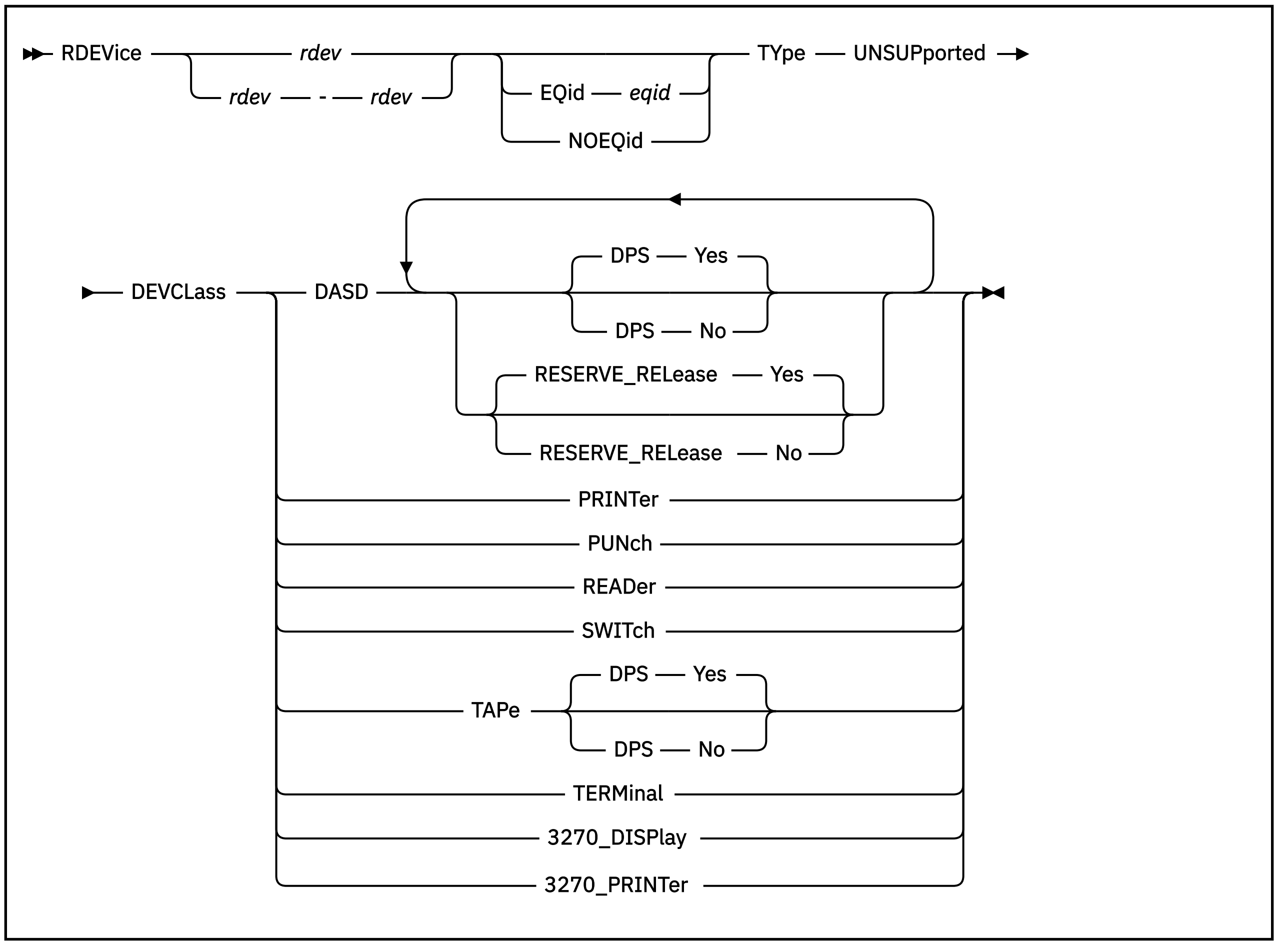
   HCP048E
   EQID must be unique for the specified device class.

Examples

1. To define a 3215 terminal at device number 0253, use the following RDEVICE statement:

```
Rdevice  0253  Type  3215
```

RDEVICE Statement (Unsupported Devices)



Purpose

Use this RDEVICE statement to define one or more unsupported devices to CP.

Note: When you define an unsupported device, you must dedicate the device to a virtual machine. To do this, specify the DEDICATE directory statement in the virtual machine's directory statement or issue the CP ATTACH command. For more information, see "DEDICATE Directory Statement" on page 500. See also ATTACH in *z/VM: CP Commands and Utilities Reference*.

Operands

rdev
rdev-rdev
is the real device number (or numbers) of the unsupported device that you want defined. The maximum number of devices allowed within a range is 256. Each *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

EQid *eqid*
assigns the device equivalency ID (EQID) *eqid* to the RDEV. The *eqid* is a string of 1–8 alphanumeric characters. Note that for CTCA, FCP, HiperSocket, and OSA devices, this EQID must be unique or be shared only by other devices of the same type. For all other devices, the EQID must be unique across all devices on the system.

NOEQid
removes a previously assigned EQID from this RDEV. After the device is varied online, it reverts back to a system-generated EQID.

TYpe UNSUPported
tells CP that this is an unsupported device type.

DEVCLass
is the device class of the unsupported device. Valid classes are:

Class	**Unsupported Device Type**
DASD	Direct access storage devices
PRINTer	Unit record printer devices
PUNch	Unit record punch devices
READer	Unit record input devices
SWITch	Dynamic switching devices.
TAPe	Tape devices
TERMinal	Terminals
3270_DISPlay	3270 display devices
3270_PRINTer	3270 printer devices

DPS Yes
(the default) tells CP that the unsupported DASD or tape device supports the dynamic path selection (DPS) function: CCW command codes X'34', Sense Path Group Identifier (SNID), and X'AF', Set Path Group Identifier (SPID).

When you specify DPS YES, CP places VM's path group identifier (PGID) in the device's control unit for each path to the control unit. This placement enables the use of a single PGID for all devices connected to the control unit, regardless of the number of guests that might be using the different devices on the control unit, how many times a given guest is re-IPLed, or how a given device might be shifted from one guest to another over time. A guest operating system, such as z/OS, has an alternate-PGID capability so it can deal with the VM PGID on these channel paths.

DPS No
tells CP that the unsupported DASD or tape device does not support the dynamic path selection (DPS) function.

When you specify DPS NO, VM takes no action with regard to path groups or PGIDs. It is the responsibility of the guest to which the unsupported DASD or tape device is dedicated or attached to form and maintain path groups, if desired, using the guest's own PGID. The use of a guest's PGID is often cumbersome if not unworkable for DPS devices, because a PGID on a channel path to a control unit applies to all devices connected to the control unit, and a PGID can only be changed after clearing any previous PGID with a system-reset of the channel path or paths.

If the guest operating system to which the device will be dedicated or attached does not contain alternate PGID support, specify DPS NO.

RESERVE_RELease Yes
(the default) tells CP that the unsupported DASD supports the reserve/release function: CCW command codes X'B4', Device Reserve (RES), X'94', Device Release (REL), and X'14', Unconditional Reserve (UR).

When you specify RESERVE_RELEASE YES for an unsupported DASD, CP issues the DEVICE RELEASE CCW command to the device whenever the virtual machine to which the device is dedicated or attached is reset by the CP SYSTEM CLEAR, SYSTEM RESET, or IPL commands.

RESERVE_RELease No
tells CP that the unsupported DASD does not support the reserve/release function. If you specify RESERVE_RELEASE NO for an unsupported DASD that contains support for the reserve/release function, a malfunction of the guest to which the device is dedicated or attached might leave a device reservation held by the guest, preventing the device from being accessed by other sharing systems.

Usage Notes

1. If the specified EQID has already been assigned to one or more other devices, CP will issue the following message:

   HCP048E
   Specified EQID already assigned to a different device.
2. If the issued command attempts to assign an EQID to multiple devices and the input device type requires each device to have a unique EQID, CP will issue the following message:

   HCP048E
   EQID must be unique for the specified device class.

Examples

1. To define an unsupported DASD, use the following RDEVICE statement:

```
Rdevice  0fff  Type  UnSupported  DevClass  DASD
```

2. To define an unsupported DASD that supports dynamic path selection, use the following RDEVICE statement:

```
Rdevice  0fe0  Type  UnSupported  DevClass  DASD  DPS  yes
```

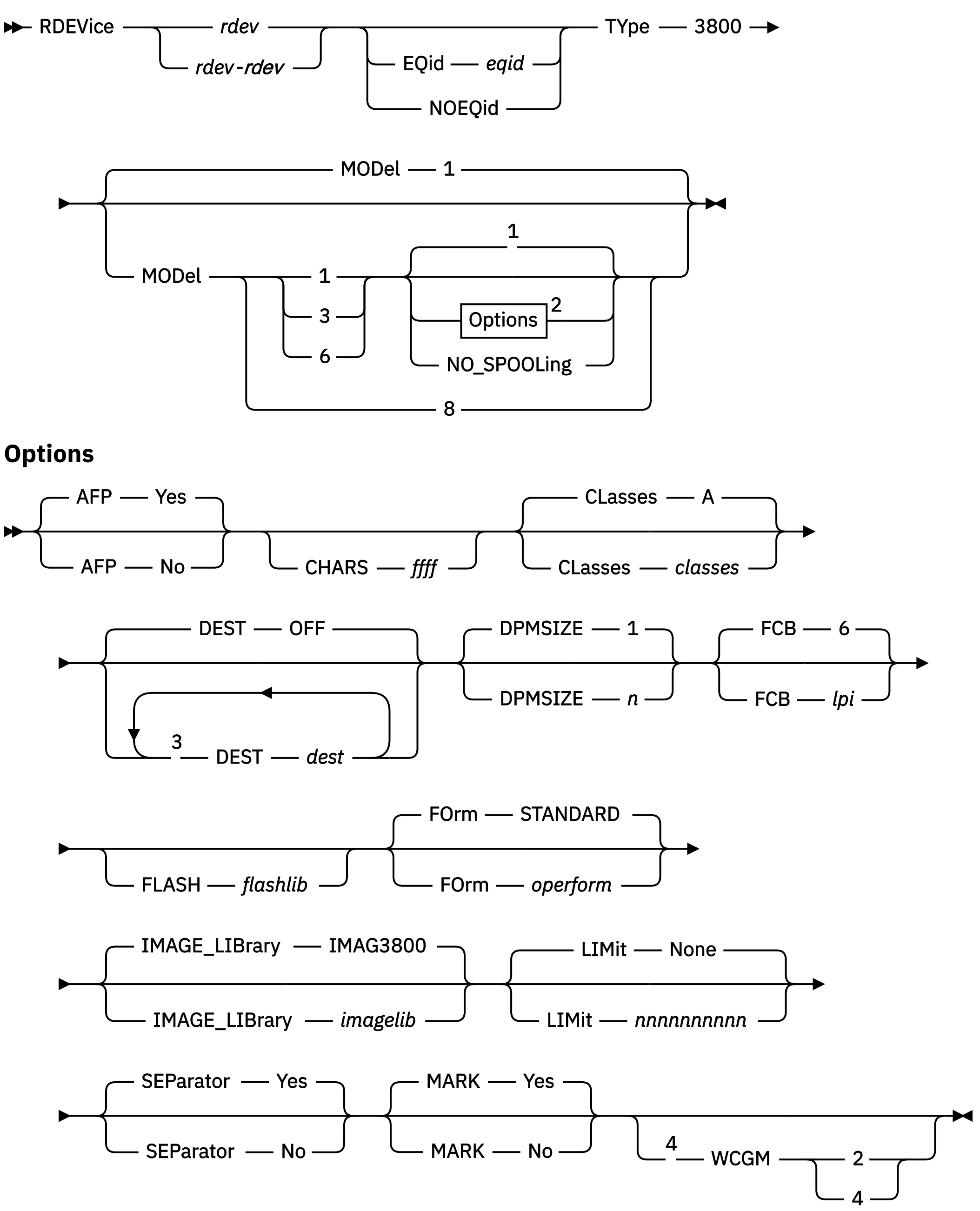
3. To define an unsupported DASD that supports both dynamic path selection and reserve/release, use the following RDEVICE statement:

```
Rdevice  0fd0  Type  UnSupported,
               DevClass DASD,
               DPS yes,
               Reserve_Release yes
```

4. To define 4 unsupported tape devices and 23 unsupported DASD that support both dynamic path selection and reserve/release, use the following RDEVICE statements:

```
Rdevice 2040-2043 Type Unsupported,        /*   Unsupported tape     */
                  DevClass Tape            /*   device               */

Rdevice 1280-1296 Type Unsupported,   /* Unsupported DASD supporting */
                  DevClass DASD,      /* reserve/release CCW and     */
                  DPS yes,            /* dynamic path selection CCW  */
                  Reserve_Release yes
```

RDEVICE Statement (3800 Printers)



Notes:

[1] The defaults you receive appear above the line in the Options fragment.
[2] The following operands can be specified in any order.
[3] You cannot specify more than four output destinations.
[4] This operand can only be specified for model 1.

Purpose

Use this RDEVICE statement to define 3800 and 3900 printers to CP.

Operands

rdev
rdev-rdev
is the real device number (or numbers) of the 3800 printer that you want defined. The maximum number of devices allowed within a range is 256. Each *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

EQid *eqid*
assigns the device equivalency ID (EQID) *eqid* to the RDEV. The *eqid* is a string of 1–8 alphanumeric characters. Note that for CTCA, FCP, HiperSocket, and OSA devices, this EQID must be unique or be shared only by other devices of the same type. For all other devices, the EQID must be unique across all devices on the system.

NOEQid
removes a previously assigned EQID from this RDEV. After the device is varied online, it reverts back to a system-generated EQID.

TYpe 3800
tells CP that the real device (or devices) that you are defining is a 3800 printer.

MODel 1, 3, 6, or 8
tells CP the model number of the 3800 printer that you are defining. The default value is 1.

Note:

1. The 3800 Models 3 and 6 are supported in Model 1 compatibility mode, defined as a Model 1, or by using the Advanced Function Printing licensed program, defined as a Model 3 or as a Model 6. If you are using them as advanced function printers, specify AFP YES.
2. The 3800 Model 8 is supported using an Advanced Function Printing licensed program.

NO_SPOOLing
tells CP not to use the 3800 printer for spooling.

AFP Yes
AFP No
specifies whether the printer is to process files with advanced function printer (AFP) characteristics, XABs, or X'5A' CCWs. The default is YES.

CHARS *ffff*
specifies the name of a character-arrangement table for the separator page to be used after a cold start. The default values for 3800 printers are

For Model 1, CHARS GF10.
For Model 3 or Model 6, CHARS GF12.

CLasses *classes*
defines the output spooling class or classes that the 3800 Model 1, Model 3, or Model 6 printer can print. Each class is 1 alphanumeric character and you can specify up to 8 classes. If you specify more than one class, do not include any blanks between the classes (for example, CLASSES ABC). If omitted, the default is A. If specified as an asterisk (*), the printer can process any file, regardless of class.

To change the spooling classes without having to re-IPL, use the CP START UR command. For more information, see START UR in *z/VM: CP Commands and Utilities Reference*.

DEST OFF
DEST *dest*
specifies the output destination values the printer can print. Specify DEST in one of the following ways:

- By default — if you omit the DEST operand, DEST OFF is assumed.
- You can specify as many as four destination values by entering four different DEST operands, as follows:

```
Dest printer1 Dest printer2 Dest printer3 Dest printer4
```

- DEST * specifies that the printer should process files regardless of destination.

To change the spooling destinations without having to re-IPL, use the CP START UR command. For more information, see START UR in *z/VM: CP Commands and Utilities Reference*.

DPMSIZE *n*
specifies the maximum size of the delayed purge queue (see note 1) that the 3800 Model 1, Model 3, or Model 6 printer uses after a cold start. The default value for *n* is 1, the maximum is 9. If *0* is specified, no delayed purge queue is maintained (See note “2” on page 257).

Note:

1. After the 3800 prints a file, CP places the file on the delayed purge queue if a delayed purge queue is being maintained. When the queue is full, CP purges the oldest file. This delay helps ensure that the 3800 (a) transfers the last page of the file from the page buffer in the 3800 printer to paper and (b) stacks the printed page. If the 3800 fails before CP purges a file from the delayed purge queue, CP places the file in system hold.
2. Specifying *0* for DPMSIZE saves some spool DASD space because files printed on the 3800 are purged immediately. However, this decreases the possibility of recovering a printer file which has failed during printing.

FCB *lpi*
specifies the name of the forms control buffer for the separator page to be used after a cold start. Note that you can override this value by naming a forms control buffer on the START command. The default value for *lpi* is 6.

The number of lines to be printed on the separator page is determined by the FCB loaded on the printer. The default separator page contains 58 lines of data. If the page length defined by the FCB is less than the default separator page length, the separator page data must be customized in order to fit on a single page. This may be done by using CP Exit points that are provided in separator page processing. For more information, see CP EXIT 4400: Separator Page Data Customization in *z/VM: CP Commands and Utilities Reference*.

FLASH *flashlib*
specifies the flash overlay for use with this device.

FOrm *operform*
is the current spooling form number that the printer can process. This form is the default operator form for the real printer when the operator starts the device after a cold start without specifying a form. Specify this parameter in one of the following ways:

- FORM *operform*, where *operform* specifies a one- to eight-character operator form number for the files the printer can process.
- FORM *, where the asterisk specifies that the printer can process files regardless of form number.
- FORM STANDARD, where STANDARD indicates that the type of paper to be mounted on the printer is the type of paper that the installation has assigned the name STANDARD. Each installation establishes its own set of form names, assigns those form names to types of paper, and informs the operations staff and end users of the correlation between form names and types of paper.

  STANDARD is the default.

IMAGE_LIBrary *imagelib*
specifies the name of the image library to be used after a cold start. Note that you can override this value by specifying an image library name on the START command.

The default value is IMAGE IMAG3800.

LIMit None
(the default) tells CP that this 3800 printer can print files with an unlimited number of records.

LIMit *nnnnnnnnnn*
: tells CP that this 3800 printer limits the size of files it can print. The variable *nnnnnnnnnn* is a 1- to 10-digit decimal number that indicates the maximum number of records per file that this printer can print.

SEParator Yes
SEParator No
: specifies whether a separator is desired for output files. The default value is SEPARATOR YES.

MARK Yes
MARK No
: specifies whether a 3800 printer will mark separator trailer pages with separator bars.

  When 'Mark Yes' is in effect, 3 separator trailer pages are printed for each file. 'Mark No' causes only one trailer page to be printed, thus saving paper. However, using 'Mark No' could also make it more difficult to separate 3800 output, as there are no markings between files. The default is YES.

WCGM 2
WCGM 4
: specifies that the 3800 has either the 2-writable-character generation module feature or the 4-writable-character generation module feature.

  Note: You can only specify this option for a 3800 model 1.

  For a 3800 model 1, the default is WCGM 2. For 3800 models 3, 6, and 8, the default is WCGM 4.

Usage Notes

1. If the specified EQID has already been assigned to one or more other devices, CP will issue the following message:

   HCP048E
   Specified EQID already assigned to a different device.
2. If the issued command attempts to assign an EQID to multiple devices and the input device type requires each device to have a unique EQID, CP will issue the following message:

   HCP048E
   EQID must be unique for the specified device class.

Examples

1. To define a Model 3 3800 printer at device number 0500 to be dedicated to a virtual machine, use the following RDEVICE statement:

   ```
   Rdevice  0500  Type  3800  Model  3
   ```
2. To define three Model 8 3800 printers at device numbers 0501, 0502, and 0503 to be dedicated to virtual machines, use the following RDEVICE statement:

   ```
   Rdevice  0501-0503  Type  3800  Model  8
   ```
3. To define a 3800:

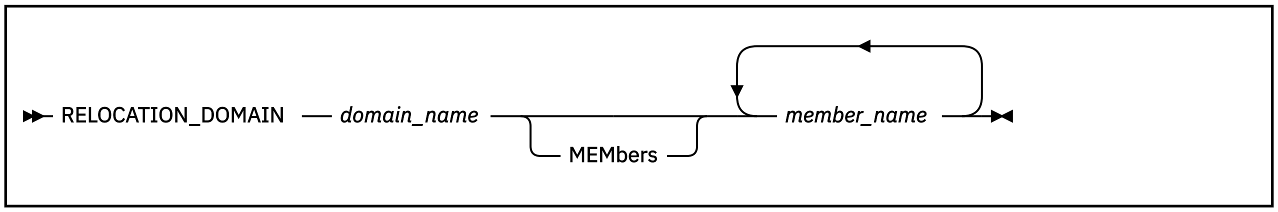
   - Model 1 printer at device number 100 with a class of 1, the 4-writable-character generation module, and an image library called IMAG3800
   - Model 3 printer at device number 150 driven by AFP
   - Model 6 printer at device number 180 driven by AFP

   use the following RDEVICE statements:

   ```
   /*-----------------------------------------------------------------*/
   /* What you always wanted to know about our printers, but were       */
   /* afraid to ask:                                                    */
   ```

```
/*                                                                    */
/*    3800 Model 1 is a CP Spooling Printer                           */
/*    3800 Model 3 is driven by the AFP Program Product               */
/*    3800 Model 6 is driven by the AFP Program Product               */


/*                                                                    */
/* Note: 3800 Models 3, 6 and 8 can be specified as either:           */
/*          Type 3800                                                 */
/*           - or -                                                   */
/*          Type AFP                                                  */
/*--------------------------------------------------------------------*/

RDevice  100  Type  3800  Model  1,
                          Class  1,
                          WCGM  4,
                          IMAGE_Library  IMAG3800

Rdevice  150  Type  3800  Model  3,
                          AFP  Yes

Rdevice  180  Type  AFP   Model  6
```

RELOCATION_DOMAIN Statement



Purpose

Use to define subsets of the SSI membership as domains. A domain specifies the set of systems among which a guest may relocate. The domain imposes a common set of architectural characteristics that ensure a guest is presented with consistent architecture whichever member of the domain the guest runs in. When an ordinary user first logs on, its associated domain is the entire SSI. A multi-configuration virtual machine is permanently assigned to the singleton domain of the member on which it is running.

How to Specify

Include as many statements as needed; they are optional. This statement must come after the SSI statement that defines the SSI cluster. If you specify more than one statement with the same domain name, the first definition will be accepted and subsequent statements will receive an error.

Operands

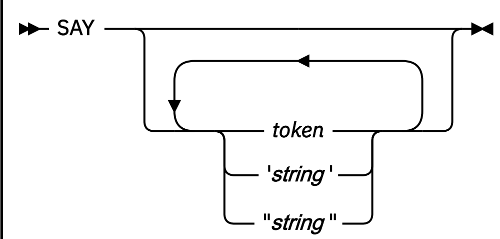
domain_name
The one- to eight-character name of the domain. The reserved word "SSI" is excluded from use (see SET VMRELOCATE command). Also, the names of SSI member systems are predefined as singleton domains and may not be modified by this statement. The domain name specified here may not be the same as the name of any member in the SSI cluster. The name may consist of only A-Z and 0-9.

MEMbers *member_name*
A list of one or more SSI member system names comprising the domain. Member system names are one- to eight-characters in length using only A-Z and 0–9.

Usage Notes

1. For more information about relocation domains, see "Using Relocation Domains" on page 754.

SAY Statement



Purpose

Use the SAY statement to write a line of text to the operator's console during initialization.

How to Specify

Include as many statements as needed; they are optional. You can place SAY statements anywhere in the system configuration file.

Operands

token
is the text to be displayed. Because the token is not delimited by quotation marks, it will be converted to upper case when it is displayed.

'string'
"string"
is the text to be displayed. The text inside the quotation marks is not converted to upper case. The string is displayed unchanged and without the delimiting quotation marks.

Usage Notes

1. Multiple blanks not within quotation marks are compressed to a single blank.
2. Multiple blanks within quotation marks are not compressed.
3. Variable substitution does not occur for any token specified on the SAY statement.
4. A SAY statement that contains no text will cause a blank line to be displayed.
5. Use two consecutive double quotation marks ("") to represent a " character within a string delimited by double quotation marks. Similarly, use two consecutive single quotation marks ('') to represent a ' character within a string delimited by single quotation marks.
6. Each SAY statement shows up as a message on the operator's console. The messages are not displayed as they are encountered. These, as well as all the other initialization messages, are queued in storage until the operator is autologged.
7. Messages that display the SAY statement text will also appear in a spooled console listing from the IPL. Use an appropriate text length on your SAY statements to ensure the entire text is captured in the spooled console file.

   The text limit for a SAY statement is approximately 1900 characters. This is the maximum amount of data that will be displayed at the operator's console. For data written to a console file, the existing limit is 299 characters. Because the logical record length of a console file is 132, also note that any data placed in a console file may be arbitrarily split into multiple lines.

Examples

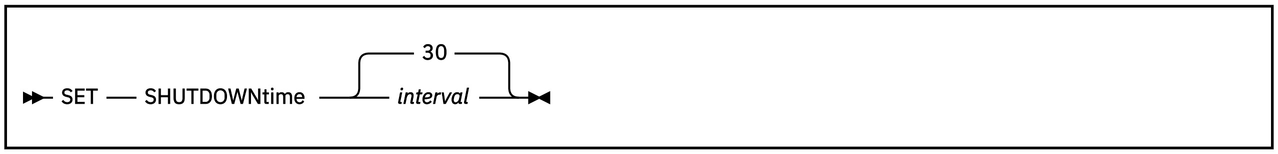
1. If your system configuration file contains the following:

```
      SAY 'at line 100, a test of SAY:'
      SAY "Reading file" -system- config
      SAY
      SAY 'Here''s my     test'    a b         c
      SAY
      SAY "Here''s another test"
```

Your operator would see:

```
      HCPZPQ2780I At line 100, a test of SAY:
      HCPZPQ2780I Reading file -SYSTEM- CONFIG
      HCPZPQ2780I
      HCPZPQ2780I Here's my     test A B C
      HCPZPQ2780I
      HCPZPQ2780I Here''s another test
```

SET SHUTDOWNTIME Statement



Purpose

The SET SHUTDOWNTIME statement specifies the CP shutdown interval. The CP shutdown interval is the time that is reserved for CP to shut down when a SHUTDOWN command is issued or the entire z/VM system shuts down automatically as the result of receiving a hardware deactivation signal.

The SET SHUTDOWNTIME configuration statement specifies how much of the entire z/VM system-shutdown interval is reserved for CP shutdown. The entire z/VM system-shutdown interval might also include a shutdown-signal timeout interval, which provides time for guest operating systems to shut down. A shutdown-signal timeout interval occurs only if the CP shutdown interval is less than the entire z/VM system-shutdown interval.

Operands

interval

specifies the amount of time that is reserved for a CP shutdown to be completed. The value specifies the number of seconds in the range 0 - 32767.

If the CP shutdown time is not specified in the SET SHUTDOWNTIME system configuration statement or by issuing the SET SHUTDOWNTIME command, the default is 30 seconds.

Usage Notes

1. The CP shutdown interval is an estimation of the time that CP requires to complete an orderly CP shutdown. If CP requires more time to shut down, the actual interval can be longer.
2. The CP shutdown interval can be dynamically reset for the current system IPL by the SET SHUTDOWNTIME command.
3. The interval that is specified on this statement can affect the interval that is designated for the entire z/VM system shutdown process.

   If the CP SHUTDOWN command is issued without a WITHIN, BY, or IMMEDIATE operand, the start of the CP shutdown process can be delayed. The start of the CP shutdown process is delayed until the system-default shutdown-signal timeout interval elapses or until all signaled guests indicate that they shut down, whichever occurs first. The time that is designated for the entire z/VM system to shut down is the sum of the system-default shutdown-signal timeout interval and the CP shutdown interval. The entire z/VM system shutdown can take less time than the sum of the system-default shutdown-signal timeout interval and the CP shutdown interval if one or both of the following conditions occur:

   - All signal-enabled guests shut down before the system-default signal-timeout interval elapses.
   - The CP shutdown process finishes in less time than is reserved by the CP SET SHUTDOWNTIME specification.

   Use caution when you specify the WITHIN, BY, or IMMEDIATE operand on the SHUTDOWN command. The WITHIN, BY, or IMMEDIATE operand specifies the time interval for the entire z/VM system shutdown. The actual shutdown-signal timeout interval is the difference between the entire z/VM system shutdown interval and the CP shutdown interval. The actual shutdown-signal timeout interval might not provide guests with enough time to complete an orderly shutdown.

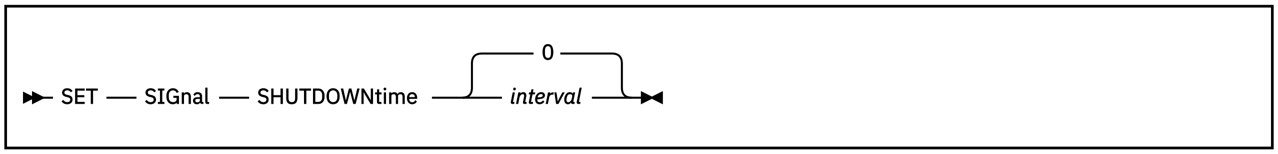
   For examples of the interaction between the SHUTDOWN command timing operands, the CP shutdown interval, and the system-default shutdown-signal timeout interval, see the examples in SHUTDOWN in *z/VM: CP Commands and Utilities Reference*.

Tip:

- The system-default shutdown-signal timeout interval is specified by the SET SIGNAL SHUTDOWNTIME system configuration statement or command. For more information, see "SET SIGNAL Statement" on page 265.
- The CP shutdown interval is specified by the SET SHUTDOWNTIME system configuration statement or command.

4. When a CP shutdown occurs due to a hardware deactivation, if the SHUTDOWNTIME is more than 300 seconds (5 minutes), there is not time for guests to shut down.

SET SIGNAL Statement



Purpose

Use the SET SIGNAL statement to specify the duration of the system-default shutdown-signal timeout interval.

Operands

SHUTDOWNtime *interval*
specifies the duration of the system-default shutdown-signal timeout interval. The *interval* value specifies a number of seconds in the range 0 - 32767. If the system-default shutdown-signal timeout interval is not specified by the SET SIGNAL SHUTDOWNTIME command or system configuration statement, the default is 0 seconds.

If a SIGNAL, FORCE, or SHUTDOWN command is issued without specifying an interval, the system-default shutdown-signal timeout interval specifies the interval for guest operating systems to complete their shutdown processing. If the interval value is greater than 0, a shutdown signal is sent to enabled users. If the interval value is 0, shutdown signals are suppressed.

Usage Notes

1. The system default shutdown signal timeout interval can be dynamically reset for the current system IPL by the SET SIGNAL command.
2. A user is enabled for shutdown signals only when a guest operating system is running in the virtual machine and is enabled to receive these signals.
3. The interval that is specified on this statement can affect the interval that is designated for the entire z/VM system shutdown process.

   If the CP SHUTDOWN command is issued without a WITHIN, BY, or IMMEDIATE operand, the start of the CP shutdown process can be delayed. The start of the CP shutdown process is delayed until the system-default shutdown-signal timeout interval elapses or until all signaled guests indicate that they shut down, whichever occurs first. The time that is designated for the entire z/VM system to shut down is the sum of the system-default shutdown-signal timeout interval and the CP shutdown interval. The entire z/VM system shutdown can take less time than the sum of the system-default shutdown-signal timeout interval and the CP shutdown interval if one or both of the following conditions occur:

   - All signal-enabled guests shut down before the system-default signal-timeout interval elapses.
   - The CP shutdown process finishes in less time than is reserved by the CP SET SHUTDOWNTIME specification.

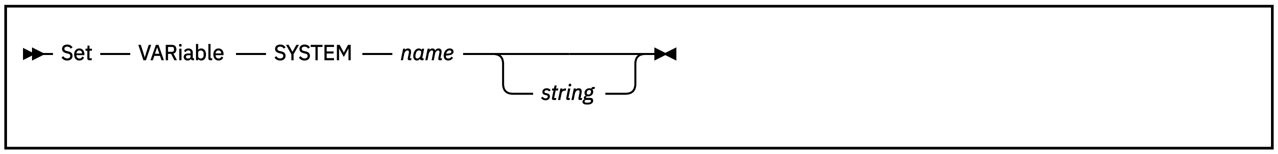
   Use caution when you specify the WITHIN, BY, or IMMEDIATE operand on the SHUTDOWN command. The WITHIN, BY, or IMMEDIATE operand specifies the time interval for the entire z/VM system shutdown. The actual shutdown-signal timeout interval is the difference between the entire z/VM system shutdown interval and the CP shutdown interval. The actual shutdown-signal timeout interval might not provide guests with enough time to complete an orderly shutdown.

   For examples of the interaction between the SHUTDOWN command timing operands, the CP shutdown interval, and the system-default shutdown-signal timeout interval, see the examples in SHUTDOWN in *z/VM: CP Commands and Utilities Reference*.

   Tip:

- The system-default shutdown-signal timeout interval is specified by the SET SIGNAL SHUTDOWNTIME system configuration statement or command.
- The CP shutdown interval is specified by the SET SHUTDOWNTIME system configuration statement or command. For more information, see “SET SHUTDOWNTIME Statement” on page 263.

SET VARIABLE Statement



Purpose

Use the SET VARIABLE statement to define and set an environment variable that is accessible to every class G user on the system. SET VARIABLE can also be used to change the value of an environment variable or to delete it by setting it to the null string. System environment variables allow automated programs to adapt to different operating environments or modes in a coordinated fashion across multiple guest virtual machines.

How to Specify

The SET VARIABLE statement is optional. You can place the SET VARIABLE statement anywhere in the system configuration file. If you specify more than one SET VARIABLE statement for the same variable name, the last statement overrides any previous specifications.

Operands

SYSTEM
indicates that this is a system environment variable.

name
specifies the name of the environment variable. This name must satisfy the following rules:

- It is 1 - 63 characters in length with no blanks.
- Valid characters are 0-9, A-Z, _ (underscore), and . (period).
- Lowercase characters a-z can be specified, but are folded to uppercase before being processed.
- It cannot begin or end with a period.
- It cannot begin with a "CP." prefix.
- It cannot begin with a numeric digit.

string
is the data string that will become the value for the specified variable. This string must satisfy the following rules:

- It is 1 - 255 characters in length.
- Valid characters are the hexadecimal values x'40' - x'FE'.
- The string can be enclosed in double or single quotation marks. If the string is enclosed in quotation marks, multiple consecutive blanks (x'40') can be embedded within the string, but are stripped from the beginning and end of the string. The beginning and ending quotation marks are removed from the string before it is processed.

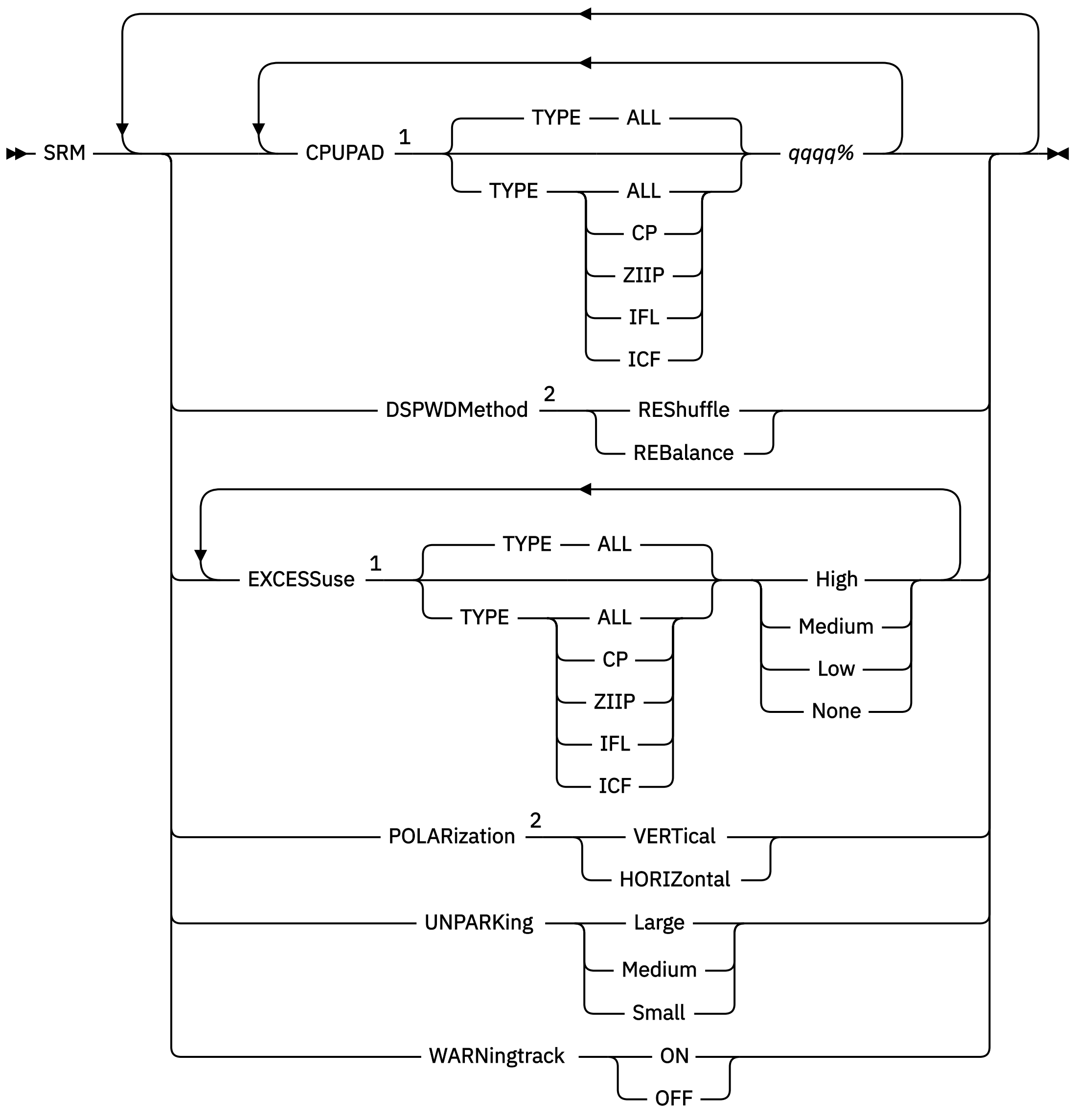
If the string is omitted, the specified variable is deleted.

Usage Notes

1. The SET VARIABLE statement differs from other system configuration statements in that the string provided for the environment variable value can include lowercase characters that are not converted to uppercase, without requiring that the string be enclosed in quotation marks. The other operands and keywords on this statement, including the name, are converted to uppercase before they are processed.

2. When the value is not enclosed in quotation marks, multiple consecutive embedded blanks are condensed into one blank, and comment delimiters and unmatched quotation marks cannot be included within the string.
3. If you need to include an odd number of single quotation marks in the value, enclose the entire string in double quotation marks. If you need to include an odd number of double quotation marks in the value, enclose the entire string in single quotation marks.
4. Instances of two consecutive quotation marks in the value are not reduced to a single quotation mark. This is true whether or not the entire string is enclosed in quotation marks.
5. Avoid breaking and continuing the variable value across multiple lines, because configuration file statement processing will insert one to three blanks between the last token of one line and the first token of the next line. Exactly how many blanks are inserted varies based on whether there are blanks before the continuation comma and whether the first token of the next line starts in column 1. Specify the variable value all on one line. Because system configuration files can have a variable record length of up to 4000 characters, the largest variable value can be accommodated easily.
6. For a complete list of environment variables defined, issue the QUERY VARIABLE ALL command.
7. When SET VARIABLE is specified for an environment variable that already has a value, the new value specified replaces the previous value.
8. Variable names that start with the prefix "CP." are reserved for IBM use and cannot be defined, changed, or deleted by the SET VARIABLE command.
9. A maximum number of 1000 system environment variables can exist at a time.
10. If SET VARIABLE is specified to delete a variable that does not exist, the statement is processed without error.

SRM Statement



Notes:

[1] The CPUPAD and EXCESSUSE operands can be specified more than once per statement, but only to specify settings for different CPU types using the TYPE operand. That is, the CPUPAD and EXCESSUSE setting for a given CPU type can be specified only once per statement.

[2] You can specify the DSPWDMETHOD and POLARIZATION operands only once per statement.

Purpose

Use the SRM statement to change system resource manager settings related to HiperDispatch. For more information about HiperDispatch, see Chapter 15, “z/VM HiperDispatch,” on page 423.

Operands

CPUPAD TYPE
  specifies the CPU type pool to which the CPUPAD setting applies.

CPUPAD *qqqq*%
specifies the amount of additional CPU capacity the system should attempt to keep unparked beyond that which the system projects it will need. *qqqq* is a value from 0 through 8000, which represents the amount of additional CPU capacity as a percentage of an entire CPU. A value of 100 represents an entire CPU. This setting applies only when the system is running in vertical polarization mode. When the system's projection plus the specified amount of additional CPU capacity exceeds the partition's entitlement, this value might not be used in determining how many CPUs are unparked.

When not otherwise specified, the default CPUPAD setting for a CPU type is 100%.

When multithreading is enabled, this value specifies the amount of additional core capacity the system should be prepared to consume beyond what the system projects it will need. A value of 100 represents an entire core.

CPUPAD is taken into account only when global performance data (GPD) is not available. GPD is a setting turned on in the partition's activation profile on the HMC (Hardware Management Console) or SE (Support Element). This setting allows PR/SM to tell a partition about usage on other partitions within the CPC.

DSPWDMethod REShuffle
sets the dispatcher work distribution algorithm to reshuffle. This algorithm periodically alters the home assignments of active virtual CPUs in an attempt to balance the number of virtual CPUs on all dispatch vectors. Unless a system configuration file SRM statement or a CP SET SRM command changes the work distribution algorithm, the system uses the reshuffle algorithm.

DSPWDMethod REBalance
sets the dispatcher work distribution algorithm to rebalance. This algorithm tracks virtual machine CPU utilization and periodically alters the home assignments of virtual CPUs in an attempt to balance the CPU demand on dispatch vectors. This option should be used with caution. Use this option only when specifically instructed to do so by IBM.

The rebalance algorithm is not compatible with multithreading, and multithreading will not be enabled if rebalance is requested.

EXCESSuse
specifies how aggressively the system should attempt to use unentitled CPU capacity. This setting applies only when the system is running in vertical polarization mode, and affects how aggressively vertical low CPUs are used to consume unentitled CPU capacity.

EXCESSuse TYPE
specifies the CPU type pool to which the EXCESSUSE setting applies.

EXCESSuse High
specifies that the system should be aggressive in using unentitled CPU capacity.

EXCESSuse Medium
specifies that the system should be moderately aggressive in using unentitled CPU capacity.

When not otherwise specified, the default EXCESSUSE setting for a CPU type is Medium.

EXCESSuse Low
specifies that the system should not be aggressive in using unentitled CPU capacity.

EXCESSuse None
specifies that vertical-low logical cores should always be parked when global performance data (GPD) is available (GPD-on).

POLARization VERTical
requests that z/VM and PR/SM operate the partition in vertical polarization mode. In vertical polarization mode, the partition's logical CPUs have differing entitlements to physical CPU resources. Some logical CPUs are entitled to an entire physical CPU, while others are entitled to only a fraction of a physical CPU or perhaps only to the unused CPU entitlements left unconsumed by other partitions. Further, in a vertical polarization mode partition, PR/SM dispatches high entitlement logical CPUs on the same physical CPUs over and over again, thereby preserving accumulated cache contents. When z/VM runs in a vertical polarization mode partition, it tends to run the system's workload on fewer logical CPUs, tends to keep MP guests' virtual CPUs together with respect to the partition's cache

topology, and tends not to move virtual CPUs among logical CPUs. In aggregate, these behaviors tend to improve system performance.

POLARization HORIZontal
requests that z/VM and PR/SM operate the partition in horizontal polarization mode. In a horizontal polarization mode partition, each online (configured) logical CPU of a particular CPU type gets an equal portion of the partition's weight for that CPU type. For instance, if a partition with 16 online logical CPUs of type CP had a PR/SM weight that entitled those logical CPUs to 8 real CPUs of type CP, then each of the 16 logical CPUs would be entitled to 50% of a real CPU; and when all other partitions consume their entitlement, these 16 will get no more than their 50% of a real CPU. Further, in a horizontal polarization mode partition, PR/SM puts less emphasis on holding logical CPUs fixed on physical CPUs, and z/VM tends to spread work evenly over the online logical CPUs and tends to let virtual CPUs move from one logical CPU to another fairly freely.

Horizontal polarization is not compatible with multithreading, and multithreading will not be enabled if horizontal polarization is requested.

UNPARKing
specifies the unparking heuristic that is to be used when global performance data (GPD) is available.

UNPARKing Large
unparks all of the logical cores that are powered, in other words, all of the entitled cores and some of the vertical-low cores.

UNPARKing Medium
unparks all of the vertical-high logical cores, all of the vertical-medium logical cores, and whichever is fewer of the following:

1. all of the vertical-low logical cores that z/VM predicts PR/SM will power
2. all of the vertical-low logical cores that z/VM predicts will be needed to help cover the workload plus the CPUPAD setting.

UNPARKing Small
unparks only the logical cores that are needed to cover the workload plus the CPUPAD setting.

WARNingtrack
enables or disables exploitation of the warning-track-interruption facility by CP. This facility notifies CP when PR/SM is about to stop running a logical processor on a physical processor. CP can then save the context of the work being dispatched and yield the logical processor voluntarily, so that the interrupted work is eligible to run on another logical processor. Enabling warning track can improve guest response time and overall performance of workloads that are run on vertical-low or vertical-medium logical processors. When the associated machine facility is available, warning track is turned on by default.

WARNingtrack ON
enables exploitation of the warning-track-interruption facility by CP.

WARNingtrack OFF
disables exploitation of the warning-track-interruption facility by CP.

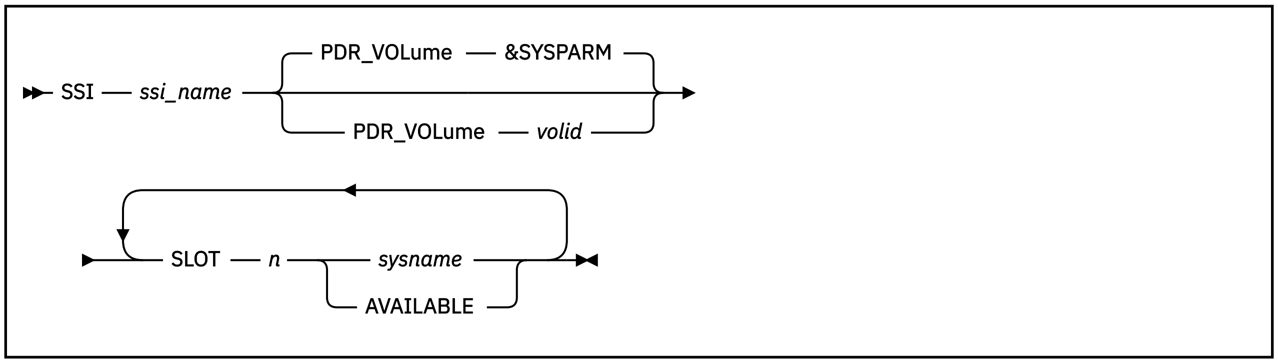
Usage Notes

1. When topology information is not available (for example, when running second level on z/VM), the default polarization is horizontal, and it cannot be changed.
2. When topology information is available, and the polarization type has not been specified with the SRM POLARIZATION configuration statement, the default polarization is vertical.
3. In using SRM EXCESSUSE, specifying something other than High, Medium, Low, or None produces message HCP002E.
4. In using SRM UNPARKING:
   - If an SRM UNPARKING statement is not specified, the system uses SRM UNPARKING LARGE.
   - Specifying something other than Large, Medium, or Small produces message HCP002E.

5. In using SRM WARNINGTRACK:

   - When the warning-track-interruption facility is not available in the configuration in which z/VM is running (a second-level system in a virtual machine, for example), the default warning-track setting is OFF, and it cannot be changed.
   - When the warning-track-interruption facility is available, and the warning-track setting has not been specified with an SRM WARNINGTRACK configuration statement, the default setting is ON.
   - The warning-track-interruption facility is most helpful in vertically-polarized configurations that include vertical-low and vertical-medium logical processors.

SSI Statement



Purpose

Use the SSI statement to define the name of the single system image (SSI) cluster and the systems that are members of the SSI cluster. The statement also identifies the location of the persistent data record (PDR).

How to Specify

SSI is an optional statement. If specified, only one statement is allowed. Subsequent SSI statements cause a message to be displayed and are ignored. Each slot can be specified only once. The SSI statement must be the same for all members in the SSI cluster.

The system initializing with a system configuration file containing the SSI statement must be one of the members of the SSI cluster. The SSI statement must be processed after the system name has been established by a previous SYSTEM_IDENTIFIER or SYSTEM_IDENTIFIER_DEFAULT statement.

Operands

ssi_name
is the name of the SSI cluster. The name can be up to eight alphanumeric characters.

PDR_VOLume *volid*
tells CP the volume label of the device that contains the persistent data record (PDR) for the SSI cluster. If omitted, the default is the volume containing the active PARM disk. The volume must be included in either the CP-owned or user volume list.

PDR_VOLume &SYSPARM
tells CP that the volume containing the active PARM disk is the device that contains the persistent data record (PDR) for the SSI cluster.

SLOT *n*
tells CP the number of the slot in the SSI member list. *n* must be a decimal number from 1 to 8. If any of the slots is not specified, it will automatically be defined as AVAILABLE.

sysname
AVAILABLE
is the 1- to 8-character system name of a system that is a member of the SSI cluster. The system name must be alphanumeric.

AVAILABLE tells CP that the slot is available for future use.

All members in an SSI cluster must be part of the ISFC collection that contains the SSI cluster. All members in the SSI cluster participate in all SSI-defined functions and services.

Usage Notes

1. Before the first system can join the SSI cluster, you must format the cylinders containing the persistent data record (PDR). You can do this by using the FORMSSI utility. For more information, see FORMSSI in *z/VM: CP Commands and Utilities Reference*.
2. The PDR volume must be an ECKD device and either a CP-owned DASD or in the user volume list.
3. When an SSI statement is specified, ACTIVATE ISLINK statements must also be specified to define direct ISFC connections to each of the other members of the SSI cluster. If other member(s) are joined to the cluster during IPL of an SSI member, ISFC connections to the joined members must be established before the system operator is logged on and IPL completes.
   - If there is not an ISFC connection to every joined member, message HCP1669I is displayed and the system waits until there are connections to all joined members.
   - If there are not enough ACTIVATE ISLINK statements to define direct connections to every joined member, message HCP1670E is displayed, followed by disabled wait state 1670.
4. If an error related to an SSI statement is found, a wait state 1682 occurs. Allowing the system to continue initialization as if it were not a member of an SSI cluster could result in loss of data. The wait state enables you to fix the problem without risking the loss of data.
5. If an SSI statement is found in the system configuration file, the system is defined as a member of an SSI cluster. Because of this membership, any system configuration file statements that define non-SSI cross-system linking are not allowed and result in a disabled wait state during IPL. The following is a list of configuration statements that conflict with an SSI statement:

   XLINK_DEVICE_DEFAULTS
   XLINK_SYSTEM_EXCLUDE
   XLINK_SYSTEM_INCLUDE
   XLINK_VOLUME_EXCLUDE
   XLINK_VOLUME_INCLUDE
6. If you want to add a new system to the SSI cluster, add the new system in an available slot in the SSI member list by using the CP SET SSI command and then promptly update the SSI statement. For more information, see SET SSI in *z/VM: CP Commands and Utilities Reference*.

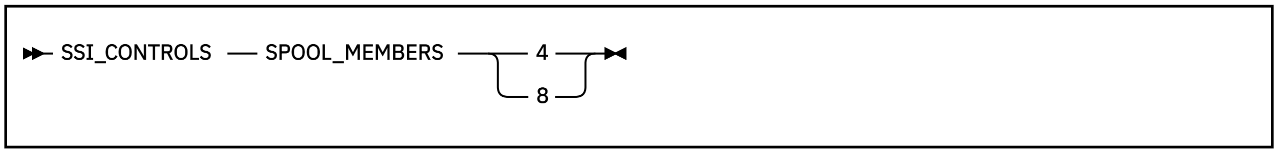
   A new system cannot be added to any of slots 5-8 until all existing members are running with the 8-member spool ID assignment algorithm. For more information, see "SSI_CONTROLS Statement" on page 275.
7. If you change the order of the members in the SSI member list, a cold start is required on all systems in the SSI cluster. A cold start deletes spool files. Files that are to be preserved during such a change should be dumped to tape using the CP SPXTAPE DUMP command. After the cold start, use SPXTAPE LOAD to restore the spool files.
8. The SET SSI PDRVOLUME command can be used to replace the current PDR volume with a new PDR volume. After each successful SET SSI PDRVOLUME command, always update the SSI configuration statement with the new PDR volume and make sure that the new PDR volume appears in either the CP-owned or user volume list in the system configuration file.

Examples

1. To define an SSI cluster that contains three systems and one available slot, use the following SSI statement:

```
SSI clustera PDR_VOL pdrvol ,
          SLOT 1  Member1 ,
          SLOT 2  Member2 ,
          SLOT 3  Member3
```

SSI_CONTROLS Statement



Purpose

Use the SSI_CONTROLS statement to specify settings that affect the behavior of members of the single system image (SSI) cluster.

How to Specify

SSI_CONTROLS is an optional statement. This statement can be included only if there is an SSI statement. If you specify more than one of these statements with the same operands, the last one overrides any previous specifications for that operand.

Operands

SPOOL_MEMBERS
specifies the maximum number of members of the cluster that will exist and therefore among how many members the per-user spool IDs must be partitioned. The options are 4 or 8 members. If no SSI_CONTROLS statement is specified, CP uses 4 as the value for SPOOL_MEMBERS.

4
indicates that there will be a maximum of 4 members in the SSI cluster and the spool IDs can be partitioned safely, so that each member gets one-fourth of the 0001-9999 spool IDs to allocate.

8
indicates there will be a maximum of 8 members in the SSI cluster and the spool IDs can be partitioned safely, so that each member gets one-eighth of the 0001-9999 spool IDs to allocate.

Usage Notes

1. The 8-member setting can be applied to a subset of the cluster members by using record qualifiers or the BEGIN and END statements. This is useful especially during the time when members are being upgraded one at a time to the 8-member support.
2. After any member in SSI slots 5-8 is defined, no member can be IPLed with SPOOL_MEMBERS 4 specified or with a level of CP that doesn't have the 8-member support.

Examples

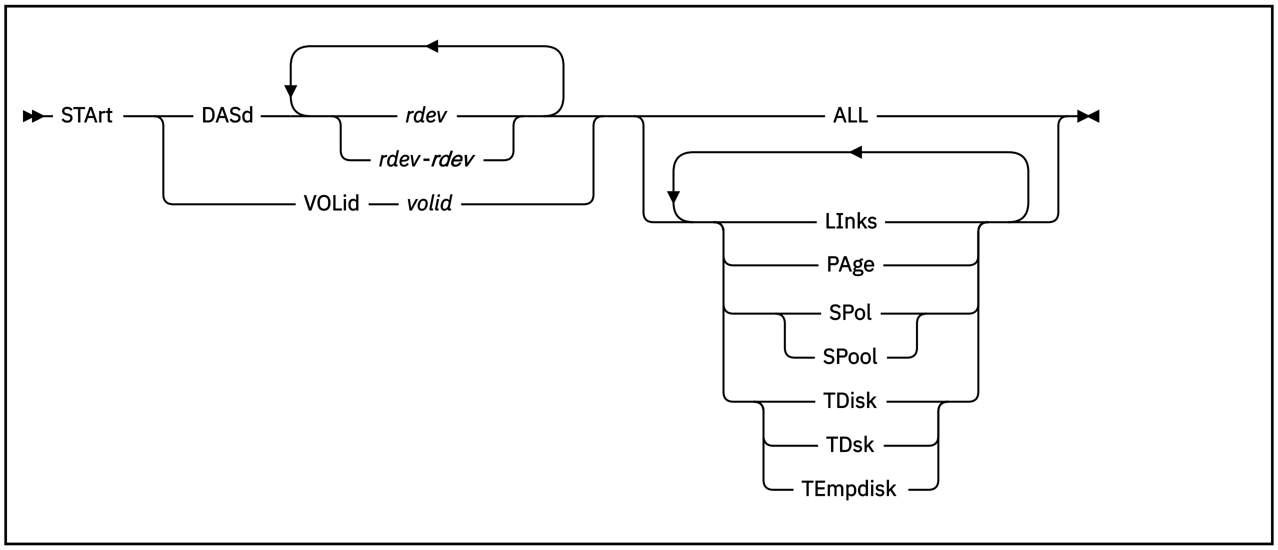
1. The following illustrates how to specify this statement when member VM1 is being upgraded to the 8-member support and is being enabled for a cluster with more than 4 members, while none of the other members have the support installed. This technique allows the support to be installed and the change to 8-member SPOOL capability to be done without a complete shutdown of the cluster. That is, each member can be upgraded with one IPL at separate times.

   ```
   VM1: SSI_CONTROLS SPOOL_MEMBERS 8
   VM2: VM3: VM4: SSI_CONTROLS SPOOL_MEMBERS 4
   ```

   Here's another way to do this, using BEGIN and END statements:

   ```
   VM1: BEGIN
   SSI_CONTROLS SPOOL_MEMBERS 8
   VM1: END
   VM2: VM3: VM4: BEGIN
   SSI_CONTROLS SPOOL_MEMBERS 4
   VM2: VM3: VM4: END
   ```

START (Disk) Statement



Purpose

Use the START statement to restart devices after they have been drained. You can also use them to change the processing options currently in effect for the devices.

How to Specify

Include as many statements as needed; they are optional. You can place START statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

DASd *rdev*
: is the real device number of the DASD you want started. The variable *rdev* is a hexadecimal number between X'0000' and X'FFFF'.

DASd *rdev-rdev*
: is a range of real device numbers specifying the DASD you want started. Each *rdev* is a hexadecimal number between X'0000' and X'FFFF'.

VOLid *volid*
: is the volume serial number of the volume you want started.

ALL
: tells CP to allow new operations on this device, including:

- Writing pages during page-out
- Allowing minidisk linking
- Allocating space for new spool records.
- Allocating temporary disk space.

LInks
: tells CP to start allowing users to link to minidisks on this device.

PAge
: tells CP to start writing pages to this device during page-out operations.

SPol
SPool
tells CP to allocate space on this device for new spool records.

TDisk
TDsk
TEmpdisk
tells CP to allocate temporary disk space on this device.

Usage Notes

1. CP processes the system configuration file during an IPL, which means CP has not attached any volumes to the system when processing your START statements. Therefore, when you specify a START VOLID statement, you can only use volumes that were previously specified in the system configuration file using CP_OWNED statements. For more information, see "CP_OWNED Statement" on page 73.

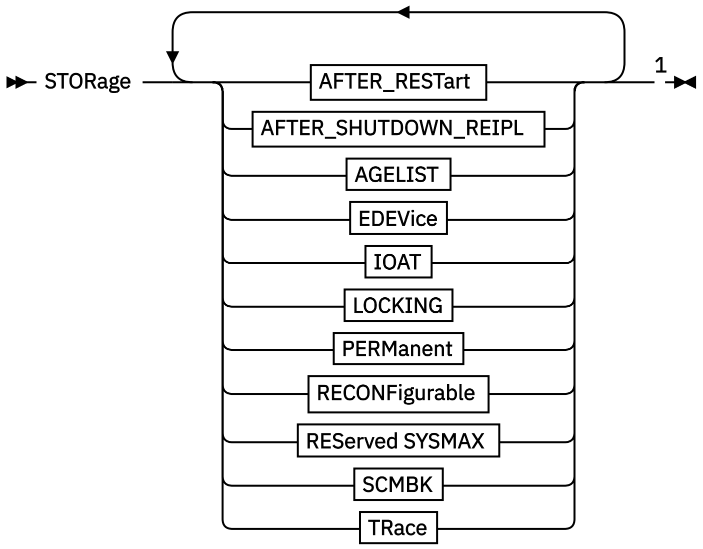
Examples

1. To have CP start allowing:
   - All new operations on all DASD at real devices numbers X'0700' through X'07FF'
   - Spooling on DASD at real device number X'0800'
   - Paging on volume SYSPG1 (which was, of course, previously defined on a CP_OWNED statement)

   use the following START statements:

```
Start  DASD   0700-07ff  All     /* Allow all activity on these DASD */

Start  DASD   0800       Spool   /* Allow spooling only on DASD 800  */

Start  VolID  syspg1     Page    /* Allow CP to allocate page space  */
                                 /*   on volume SYSPG1               */
```

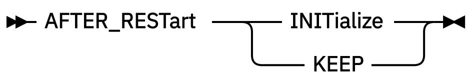
STORAGE Statement



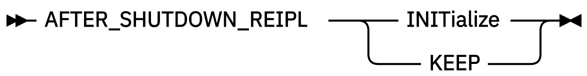
Notes:

[1] You can specify the operands in any order, but you can specify an operand only once per STORAGE statement.

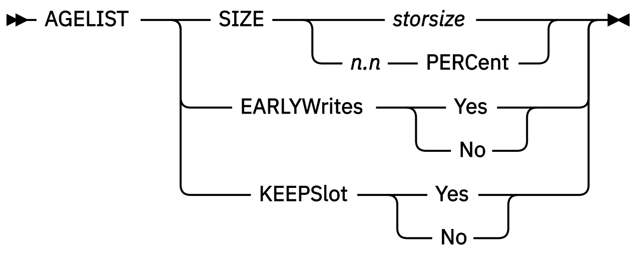
AFTER_RESTart



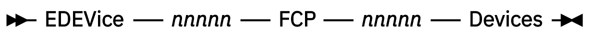
AFTER_SHUTDOWN_REIPL



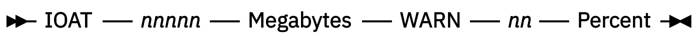
AGELIST

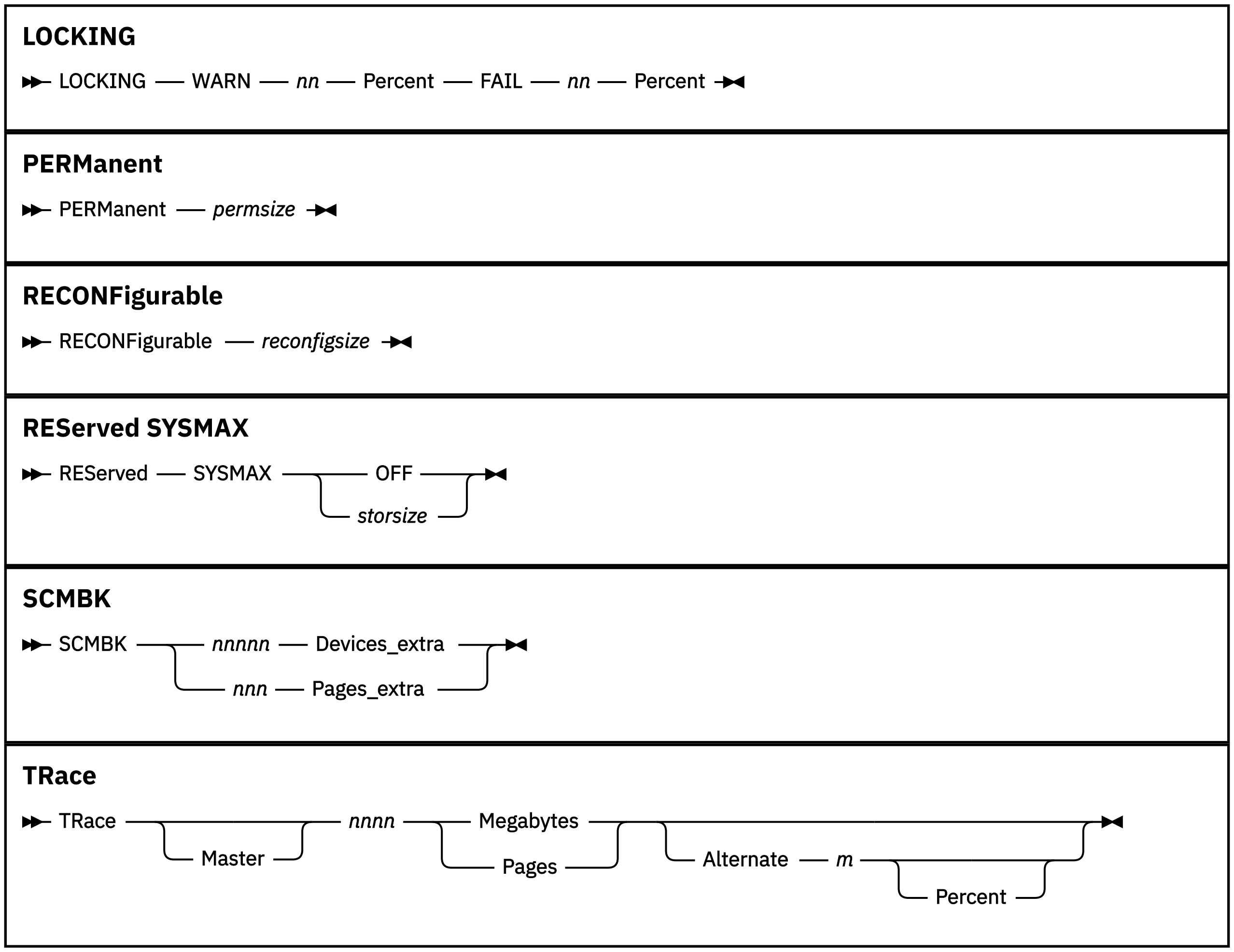


EDEVice



IOAT




LOCKING
LOCKING WARN nn Percent FAIL nn Percent
PERManent
PERManent permsize
RECONFigurable
RECONFigurable reconfigsize
REServed SYSMAX
REServed SYSMAX OFF storsize
SCMBK
SCMBK nnnnn Devices_extra nnn Pages_extra
TRace
TRace Master nnnn Megabytes Pages Alternate m Percent

Purpose

Use the STORAGE statement to allocate real storage for system use and to define storage usage warning and failure levels.

How to Specify

Include as many statements as needed; they are optional. You can place STORAGE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications for that operand.

Operands

AFTER_RESTart
specifies how to establish the real storage configuration when the system is automatically reIPLed after a CP abend or PSW RESTART. If AFTER_RESTART is not specified in the system configuration file, INITIALIZE is the default setting.

INITialize
indicates that the storage configuration should be initialized using the PERMANENT and RECONFIGURABLE values specified on the STORAGE statement in the system configuration file. If the IPL parameter STORE= is specified, the storage configuration is initialized using the value of the STORE= parameter as the amount of permanent storage.

KEEP
indicates the amount of permanent and reconfigurable storage online to z/VM at system termination should be reestablished during the reIPL, provided the CP nucleus being IPLed has

the same product, component, release, and version level as that of the terminating system. If these attributes do not match, the behavior described under the INITIALIZE option occurs. Note that the KEEP option is ignored when the IPL parameter STORE= is specified.

AFTER_SHUTDOWN_REIPL
specifies how to establish the real storage configuration when the system is reIPLed automatically after a SHUTDOWN REIPL. If AFTER_SHUTDOWN_REIPL is not specified in the system configuration file, INITIALIZE is the default setting.

INITialize
indicates the storage configuration should be initialized using the PERMANENT and RECONFIGURABLE values specified on the STORAGE statement in the system configuration file. If the IPL parameter STORE= is specified, the storage configuration is initialized using the value of the STORE= parameter as the amount of permanent storage.

KEEP
indicates the amount of permanent and reconfigurable storage online to z/VM at system termination should be reestablished during the reIPL, provided the CP nucleus being IPLed has the same product, component, release, and version level as that of the terminating system. If these attributes do not match, the behavior described under the INITIALIZE option occurs. Note that the KEEP option is ignored when the IPL parameter STORE= is specified.

AGELIST SIZE
is the target size of the global aging list used by the frame replenishment algorithm, specified in one of the following ways:

storsize
is an amount of storage. The format of the storage size is *nu*, where *n* is a 1- to 7-digit decimal number greater than 0 and *u* is the 1-character storage unit suffix (M,G,T,P,E).

The specified storage size must not exceed 5% of the current size of the Dynamic Paging Area (DPA).

***n.n* PERCent**
is a percentage of the current size of the DPA.

The decimal percentage can have at most one decimal place, and be in the range of .1 through the maximum allowed percentage of 5%. Regardless of the size of the configuration, CP never sets a target size smaller than 1 MB.

The size of the DPA varies over time and it includes available frames and those that contain pageable content. When a percent value is specified, the system recalculates the target storage size periodically to adjust to the changing size of the DPA.

See usage note “2” on page 283.

If AGELIST SIZE is not specified on a STORAGE statement, the system uses a default size of 2%.

AGELIST EARLYWrites
specifies how the frame replenishment algorithm backs up page content to auxiliary storage. When Yes is specified, pages are backed up in advance of frame reclaim to maintain a pool of readily reclaimable frames. When No is specified, pages are backed up only when the system is in need of frames.

If AGELIST EARLYWrites is not specified on a STORAGE statement, the default value for the system is Yes.

AGELIST KEEPSlot
indicates whether the auxiliary storage address (ASA) to which a page is written during frame replenishment should remain allocated when the page is later read and made resident. The default value is Yes.

Specifying Yes preserves a copy of the page on the paging device and eliminates the need to rewrite the contents if the page is unchanged prior to the next steal operation. Keeping the slot might reduce the amount of paging I/O, but could result in more fragmentation on the device.

When AGELIST KEEPSlot is set to No:

- The ASA is released upon read and made available for assignment to another page. Releasing the paging ASA on a read operation has the advantage of requiring a smaller auxiliary storage footprint.
- Some resident pages might still have copies on auxiliary storage unless EARLYWrites NO is also specified.

EDEVice *nnnnn* FCP *nnnnn* Devices

specifies the number of expected emulated devices that will be defined within the z/VM LPAR, and the number of unique FCP subchannels that will be used for this configuration. This information will be used to set the size of the reserved pool for memory allocations during device configuration.

The number of emulated devices can be 1–65,535, and each one will occupy roughly 17,000 bytes of pool storage. The number of FCP devices can also be 1–65,535, and each will occupy about 113,000 bytes of pool storage. There cannot be more than 65,535 combined EDEVs and FCPs. If this statement is not specified, a default pool for 100 EDEVs and 10 FCP subchannels will be allocated.

IOAT

specifies how much storage will be allocated to the IOAT subpool at IPL and specifies when to send a warning message to the operator indicating the subpool's storage is running low. The IOAT operand should be specified when PCIe functions are used. The number of megabytes needed is determined by trial and error. For more information about PCIe functions, see Chapter 17, “Using PCIe Functions for z/VM Guests,” on page 439.

The size of the subpool can range from 0 to 65535 megabytes. If 0 is specified or if the subpool fills up, non-subpool storage is obtained when it is needed after IPL, if it is available. The QUERY FRAMES command can be used to monitor IOAT subpool usage.

The valid range for WARN is 0 to 99 percent.

For example, if you want 2 megabytes of storage reserved at IPL and a warning message when 80% of that storage has been used, specify:

```
STORAGE IOAT 2 Megabytes WARN 80 Percent
```

If the amount of IOAT storage specified is not available when the system is IPLed, message HCP1166E is issued and the IOAT storage size is set to 0. This scenario is more likely when the amount of requested IOAT storage is over 2048 MB (2G) and the IPL parameter STORE= is not used.

LOCKING

specifies when to issue warning and failure messages for PCIe storage pin requests of available system storage. These can be used by the system programmer to control storage usage by PCIe functions. The percentage specified in the statement is compared to the percentage of system locked pages to available pages. For more information about PCIe functions, see Chapter 17, “Using PCIe Functions for z/VM Guests,” on page 439.

The valid percent range for WARN and FAIL is 0 - 99. The WARN value must be less than or equal to the FAIL value.

For example, if you want a warning message when 50% of available pages are locked and lock requests to fail at 80%, you specify:

```
STORAGE LOCKING WARN 50 Percent FAIL 80 Percent
```

If LOCKING is not specified, no warning or failure messages will be displayed.

PERManent *permsize*

specifies the amount of permanent memory you want z/VM to bring online during IPL. After permanent memory comes online to z/VM, it remains online until system termination.

The format of *permsize* is *nu*, where *n* is a 1- to 7-digit decimal number greater than 0, and *u* is the 1-character storage unit suffix (M, G, T, P, or E). The value is rounded up to a multiple of the storage increment size, which is calculated from the storage allocations in the partition's image profile.

The default for permanent storage is first calculated as the lesser of the size of storage element 0 or 4GB, then the greater of the following:

- The first calculation above
- The size of storage element 0
- The storage increment size.

RECONFigurable *reconfigsize*
specifies the amount of reconfigurable memory you want z/VM to bring online during IPL. Reconfigurable memory that comes online to z/VM can be removed later using the SET STORAGE command.

The format of *reconfigsize* is *nu*, where *n* is a 1- to 7-digit decimal number that is greater than 0 and *u* is the 1-character storage unit suffix (M, G, T, P, or E). The value is rounded up to a multiple of the storage increment size, which is calculated from the storage allocations in the partition's image profile. You can also specify a *reconfigsize* of 0, without a storage unit suffix, to indicate that the system should be IPLed with no reconfigurable storage. This is the default.

Reconfigurable storage is restricted to a percentage of all configured storage. This percentage could vary with different levels of z/VM. The amount of reconfigurable storage is limited to 50% of all online storage.

REServed SYSMAX
specifies the maximum amount of storage that can be reserved for users, NSSs, and DCSSs using the SET RESERVED command.

OFF
indicates there is no explicit system maximum storage that can be reserved. In this case the total amount of reserved storage is capped only by the size of the DPA at the time the SET RESERVED command is issued.

storsize
is the maximum amount of storage that can be reserved. The format of the storage size is *nu*, where *n* is a 1- to 7-digit decimal number greater than 0 and *u* is the 1-character storage unit suffix (M,G,T,P,E).

SCMBK *nnnnn* Devices_extra
SCMBK *nnn* Pages_extra
tells CP how much additional storage to allocate for SCMBKs. CP allocates the SCMBK space in contiguous pages of storage. During initialization, CP allocates one SCMBK for every subchannel on the system and rounds the amount of storage up to the nearest page. Thus, you have enough SCMBK space for every real device on your system plus some free SCMBK space. How much free SCMBK space you have will vary, depending upon how many real devices you have on your system.

When you specify the SCMBK operand, CP calculates the amount of SCMBK storage exactly as if you had not specified the SCMBK operand, and then CP adds on the extra storage you want.

When you specify the DEVICES_EXTRA operand, CP takes the number of extra devices you specified, calculates the number of pages of storage that it needs to reserve for those extra devices, and then adds the extra pages to the number of pages of SCMBK space that it originally calculated for the existing subchannels on your system. The variable *nnnnn* must be a 1- to 5-digit decimal number between 0 and 65535.

When you specify the PAGES_EXTRA operand, CP takes the number of pages that it originally calculated for the existing subchannels on your system and adds to that the number of extra pages you specified. The variable *nnn* must be a 1- to 3-digit decimal number between 0 and 511.

TRace
tells CP how many 4 KB pages or megabytes to allocate to the internal trace table for the master processor and alternate processors in the configuration. The specification must be in the range of 3 - 131072 pages or 1 - 512 megabytes.

For example, if you run z/VM on a uniprocessor and you specify:

```
    Storage  Trace  100  Pages
```

CP allocates 100 pages for the internal trace table.

If you run z/VM on a multiprocessor and you specify:

```
    Storage  Trace  Master  100  Pages  Alternate  90  Percent
```

CP allocates 100 pages for the internal trace table of the master processor, and 90 percent of the 100 pages (that is, 90 pages) as the internal trace table for each of the other processors in the complex.

If you omit the TRACE operand, CP calculates the internal trace table storage size by allocating 1 page of storage for every 64 pages (256 KB) of real storage or 100 pages, whichever is smaller.

Usage Notes

1. The STORAGE statement specifies the amount of permanent and reconfigurable real storage that CP brings online during IPL. If you specify the IPL parameter STORE=, the value of this parameter identifies the amount of permanent storage that comes online during IPL. When the IPL parameter STORE= is used, the STORAGE PERMANENT and STORAGE RECONFIGURABLE statements in the system configuration file are ignored. Regardless of the specification used to initialize storage at IPL, the SET STORAGE command can be used after IPL to add more permanent or reconfigurable storage, or to remove reconfigurable storage.
2. The target size of the global aging list, as specified on the AGELIST operand, is a guideline to CP. The actual amount of storage in the global aging list at any point in time might differ significantly below the target amount, and at times might rise above the value. The size of the aging list can be changed after IPL using the SET AGELIST command. For more information, see SET AGELIST in *z/VM: System Operation*.
3. The SCMBK operand will be ignored when format-1 measurement blocks are being used.
4. The EDEVICE operand is compared with any EDEVICE statements in the system configuration file. The larger of the two will be used to determine the pool sizes.
5. z/VM limits allocations within reconfigurable storage to data that can be moved easily if a SET STORAGE command is issued to remove the storage. To provide the most flexibility to z/VM in allocating memory, you should limit the amount of reconfigurable storage you define to the amount you expect to eventually remove. z/VM converts a reconfigurable storage increment to permanent storage when there is insufficient permanent storage available for normal system operation.
6. Any portion of your `Initial` or `Reserved` storage allocations, as specified in the image profile for your logical partition, can be defined to z/VM as permanent or reconfigurable storage. However, z/VM requires that there be 4 GB or more of permanent storage online before any reconfigurable storage can be defined. Reconfigurable storage cannot be defined on all supported levels of hardware. See *z/VM: Migration Guide* for the platforms that support reconfigurable storage.
7. The amount of permanent storage that is online to z/VM after IPL might exceed your STORAGE PERMANENT specification. This can occur when the amount of permanent storage that z/VM initializes before processing your STORAGE PERMANENT specification exceeds your permanent value. Before reading the system configuration file, z/VM uses the amount of online storage at system initialization to determine how much storage is initialized as permanent storage:

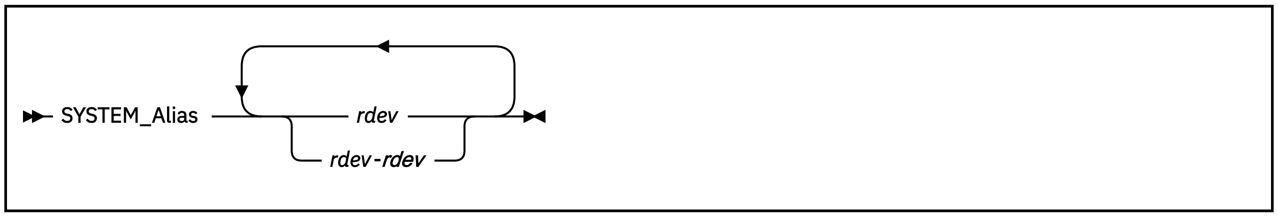
   - If the online storage at system initialization is less than or equal to 4 GB, all online storage is initialized as permanent storage.
   - If the online storage at system initialization is greater than 4 GB, one storage increment or 4 GB (whichever is greater) is initialized as permanent storage.

   The amount of PERMANENT storage that you specify on the STORAGE statement includes the permanent storage that is initialized before z/VM reads the system configuration file.
8. An incorrect STORAGE specification can negatively affect the responsiveness of your system. Prior to IPL, use the CPSYNTAX utility to make sure the system configuration file is syntactically correct. You

might also want to have your STORAGE statements specified within a TOLERATE_CONFIG_ERRORS NO section. When CP finds an error in a system configuration file statement that follows a TOLERATE_CONFIG_ERRORS NO statement, it remembers the error occurrence and, after displaying all of the errors encountered, the operator is prompted as to whether processing should continue.

9. When you specify a fixed AGELIST size, the value specified is adjusted downward to 5% of the dynamic paging area (DPA) size, if necessary. The adjustment takes place during IPL, after storage initialization completes, and also after completion of a SET STORAGE command to remove storage from the configuration. If an adjustment is made, message HCP2597I is issued to the system operator. There is no automatic increase in the fixed AGELIST size when a SET STORAGE command is used to add storage to the configuration. If you want the AGELIST size to vary based on the DPA size, as well as when the storage configuration size is increased and decreased, define the AGELIST size using a DPA percentage rather than a fixed value.

SYSTEM_ALIAS Statement



Purpose

Use the SYSTEM_ALIAS statement to specify HyperPAV alias devices to be automatically attached to the system during system initialization.

Operands

rdev
rdev-rdev
is the real device number of a HyperPAV alias device to be attached to the system during system initialization. The variable *rdev* must be a hexadecimal number between X'0000' and X'FFFF'. You can specify a single real device, a list, a range, or any combination thereof.

Usage Notes

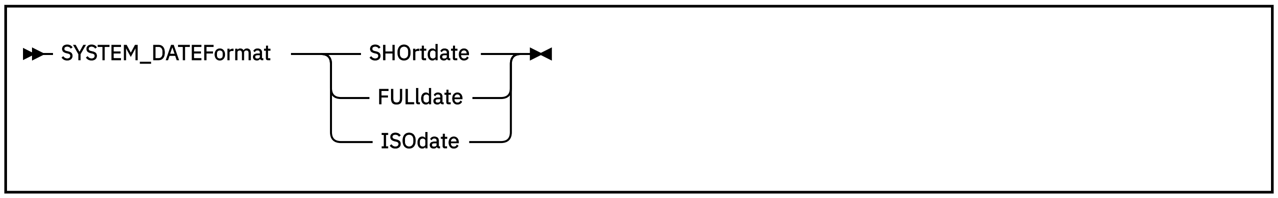
1. Devices specified that are not HyperPAV alias devices are ignored.

Examples

1. To have HyperPAV alias devices 5C44-5C47 attached to the system during system initialization, use the following SYSTEM_ALIAS statement:

```
SYSTEM_ALIAS 5C44-5C47
```

SYSTEM_DATEFORMAT Statement



Purpose

Use the SYSTEM_DATEFORMAT statement to set the system-wide default date format for commands that provide multiple date formats.

How to Specify

Include as many statements as needed; they are optional. You can place SYSTEM_DATEFORMAT statements anywhere in the system configuration file. If you specify more than one SYSTEM_DATEFORMAT statement, the last statement overrides any previous specifications.

Operands

SHOrtdate
specifies that dates in command responses be displayed in mm/dd/yy, mm/dd, or yy/mm/dd format, where mm is the month, dd is the day of the month, and yy is the 2-digit year.

FULldate
specifies that dates in command responses be displayed in mm/dd/yyyy or yyyy/mm/dd format, where mm is the month, dd is the day of the month, and yyyy is the 4-digit year.

ISOdate
specified that dates in command responses be displayed in yyyy-mm-dd format, where yyyy is the 4-digit year, mm is the month, and dd is the day of the month.

Usage Notes

1. If the SYSTEM_DATEFORMAT statement is not in the system configuration file, then the system-wide default date format is SHORTDATE.
2. The format of the dates, such as mm/dd/yy, mm/dd, and yy/mm/dd, are dependent upon the command or routine that displays or generates the date.

Examples

1. To define a default date format of mm/dd/yy, mm/dd, or yy/mm/dd, use the following SYSTEM_DATEFORMAT statement:

```
SYSTEM_DATEFORMAT SHORTDATE
```

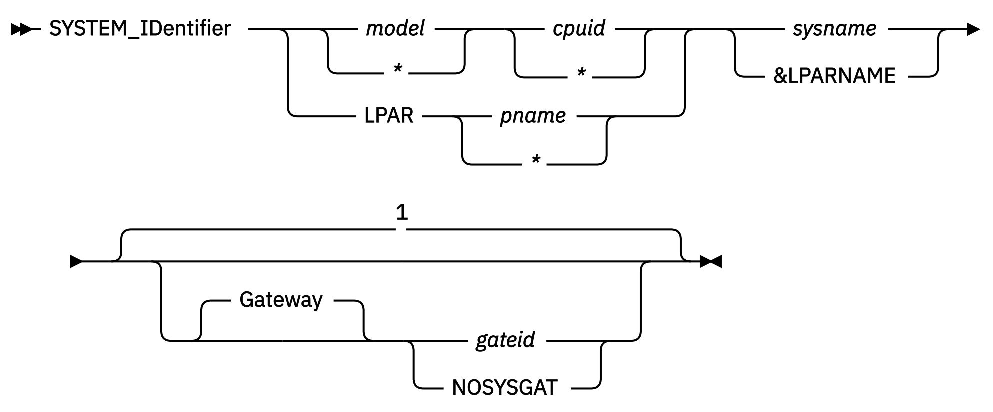
2. To define a default date format of mm/dd/yyyy or yyyy/mm/dd, use the following SYSTEM_DATEFORMAT statement:

```
SYSTEM_DATEFORMAT FULLDATE
```

3. To define a default date format of yyyy-mm-dd, use the following SYSTEM_DATEFORMAT statement:

```
SYSTEM_DATEFORMAT ISODATE
```

SYSTEM_IDENTIFIER Statement



Notes:

[1] The default gateway name is the system name.

Purpose

Use the SYSTEM_IDENTIFIER statement to define the system name (system ID) for the z/VM system to be run in a specified logical partition (LPAR) or on a specified processor. If your installation has several processors, you can specify a SYSTEM_IDENTIFIER statement for each one. At initialization, CP matches either the real LPAR name or the processor model and CPU identification numbers with those specified in the SYSTEM_IDENTIFIER statements to determine the system name to be used. The selected system name appears on printed output separator pages and in the status area of 3270 display screens.

The SYSTEM_IDENTIFIER statement also identifies the system gateway name for a system. The system gateway is identified automatically when CP initializes on the system. A system gateway provides a way to access private or global resources on a specific system within a CS or TSAF collection. It also provides access to private or global resources in a CS collection from an adjacent TSAF collection. Similarly, resources in a TSAF collection can be accessed from an adjacent CS collection.

By default, the system gateway name is the same as the system name for the system; this is the recommended naming convention. However, if this default conflicts with another gateway name, you can use the SYSTEM_IDENTIFIER statement to specify a unique system gateway name for the system.

How to Specify

Include as many statements as needed; they are optional. You can place SYSTEM_IDENTIFIER statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications. If one or more statements match the system being IPLed, the last matching statement overrides any previous matching statements.

Operands

model

is a 1- to 4-character string that specifies the processor unit model number. CP evaluates the string using pattern matching rules that are described in usage note "3" on page 288. An asterisk (*) tells CP to match any processor unit model number.

cpuid

is a 1- to 6-character string that specifies the processor identification number of a CPU in the processor complex. CP evaluates the string using pattern matching rules that are described in usage note "3" on page 288. An asterisk (*) tells CP to match any processor identification number.

LPAR *pname*
specifies the logical partition name. *pname* is a 1- to 8-character string that can consist of only letters A to Z and numbers 0 to 9. During statement processing, any lowercase letters are translated to uppercase. CP evaluates the string using pattern matching rules that are described in usage note "3" on page 288. An asterisk (*) tells CP to match any logical partition name.

sysname
is the 1- to 8-character system name (system ID).

If the system is a member of an SSI cluster, *sysname* must be alphanumeric (no special characters).

If the system is not a member of an SSI cluster, it is best (but not required) that you do not use the characters @, #, ¢, and " in the *sysname* because the system, by default, assigns these characters as logical line editing symbols. For more information, see Logical Line Editing Symbols in *z/VM: CP Commands and Utilities Reference*.

&LPARNAME
tells CP to use the LPAR name as the system name. If CP is executing in a virtual machine, the user ID is used as the system name. LPAR names longer than 8 characters are truncated.

Gateway *gateid*
is the 1- to 8-character name of the system gateway that will be identified when this system initializes. The *gateid* name is optional; if not specified, the system gateway name defaults to the system name. If this system is a member of an SSI cluster, the system gateway name must match the system name. ISFC will then use this value as the system's node name within a CS collection.

If you specify NOSYSGAT as the *gateid* value, a system gateway name is not identified for the system. Also, the specified system cannot join a CS collection or an SSI cluster.

Usage Notes

1. You can use the *sysname* defined on this statement as a record qualifier on other statements to be processed for the same z/VM system.
2. You can include the *sysname* in a nickname group defined on an EQUATE statement. For more information, see "EQUATE Statement" on page 155.
3. Pattern matching generally follows the rules used by the CMS LISTFILE command. Use an asterisk (*) to match any number of characters; use a percent sign (%) to match any single character.
4. If the system is a member of an SSI cluster, the system name is the member name. Because each member of an SSI cluster must have a unique system name and IBM recommends using a common system configuration file, each member should have its own SYSTEM_IDENTIFIER statement in the system configuration file.
5. To use LPAR *pname* or LPAR * in the SYSTEM_IDENTIFIER statement, the LPAR name must be unique within your enterprise. If the LPAR name is not unique, you must specify the processor model and CPU ID instead.

Examples

1. Set BOSTON3 as the system name for any processor in the complex. In addition, this method tells CP to match any processor model and any processor in that complex.

```
  System_Identifier * %20721 Boston3     /* Set Boston3 as our   */
                                         /* system name          */
                                         /* for any processor    */
                                         /* in the complex       */
```

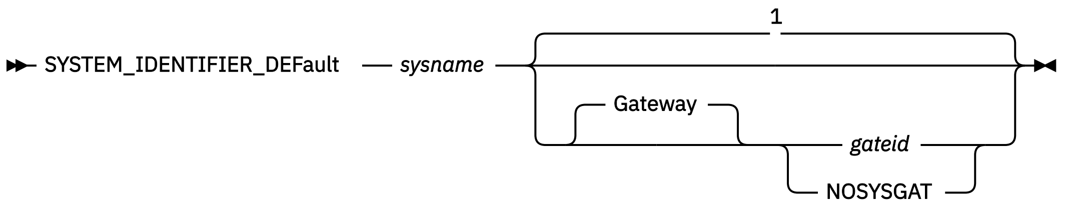
2. To make the system name be the same as the LPAR name, use the following SYSTEM_IDENTIFIER statement:

```
System_Identifier * * &LPARNAME         /* Use LPAR name as the sysname */
```

3. When the system runs in the LPAR named LP1, use the following statement to make the system name "YANKEE1":

   ```
   System_Identifier LPAR LP1 YANKEE1     /* If in LPAR LP1, use YANKEE1 */
   ```

SYSTEM_IDENTIFIER_DEFAULT Statement



Notes:

[1] The default gateway name is the system name.

Purpose

Use the SYSTEM_IDENTIFIER_DEFAULT statement to provide CP with a default system name. CP uses the SYSTEM_IDENTIFIER_DEFAULT statement when z/VM is loaded on a system whose model and CPU identification numbers do not correspond with any of those specified on SYSTEM_IDENTIFIER statements.

The default system name then appears on printed output separator pages and in the status area of 3270 display screens.

The SYSTEM_IDENTIFIER_DEFAULT statement also identifies the system gateway name for a system. The system gateway is identified automatically when CP initializes on the system. A system gateway provides a way to access private or global resources on a specific system within a CS or TSAF collection. It also provides access to private or global resources in a CS collection from an adjacent TSAF collection. Similarly, resources in a TSAF collection can be accessed from an adjacent CS collection.

By default, the system gateway name is the same as the system name for the system; this is the recommended naming convention. However, if this default conflicts with another gateway name, you can use the SYSTEM_IDENTIFIER_DEFAULT statement to specify a unique system gateway name for the system.

Because each system in an SSI cluster must have a unique system name and IBM recommends using a common system configuration file, IBM recommends you not include a SYSTEM_IDENTIFIER_DEFAULT statement in the system configuration file for an SSI member.

How to Specify

Include as many statements as needed; they are optional. You can place SYSTEM_IDENTIFIER_DEFAULT statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

sysname
: is the 1- to 8-character system name.

Gateway *gateid*
: is the 1- to 8-character name of the system gateway that will be identified when this system initializes. The *gateid* name is optional; if not specified, the system gateway name defaults to the *sysname* for the system. If this system is a member of the SSI cluster, the system gateway name must match the system name. ISFC will then use this value as its node name within a CS collection.

  If you specify NOSYSGAT as the *gateid* value, a system gateway name is not identified for the system. Also, the specified system cannot join a CS collection or an SSI cluster.

Usage Notes

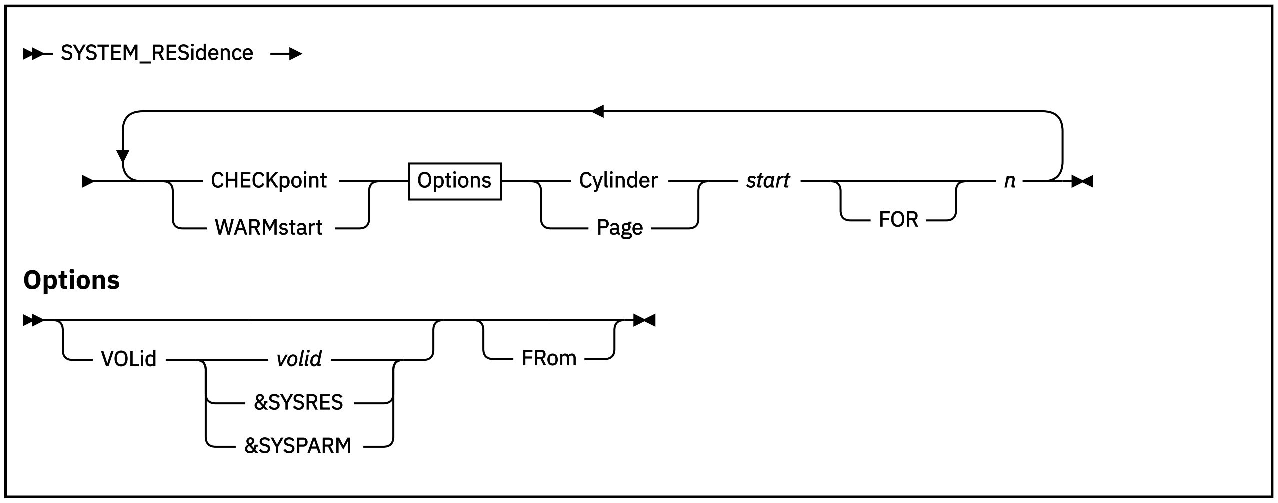
1. It is recommended that the characters @, #, ¢, and " not be used in the *sysname* because the system, by default, assigns these characters as logical line editing symbols. For more information, see Logical Line Editing Symbols in *z/VM: CP Commands and Utilities Reference*.

Examples

1. To define a default system name of BOSTON3 for your system, use the following SYSTEM_IDENTIFIER_DEFAULT statement:

```
System_Identifier_Default  Boston3         /* No matter what machine */
                                           /* CP is IPLed on, let us */
                                           /* be known as BOSTON3    */
```

SYSTEM_RESIDENCE Statement



Purpose

Use the SYSTEM_RESIDENCE statement to describe the layout of the CP system residence disk.

How to Specify

Include as many statements as needed; they are optional. You can place SYSTEM_RESIDENCE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

CHECKpoint
tells CP that you are defining the real starting location and the maximum number of pages or cylinders that the checkpoint process will use to preserve system restart data.

WARMstart
tells CP that you are defining the real starting location and the maximum number of pages or cylinders that will be used for saving warm start data.

VOLid *volid*
tells CP the volume label of the device to use as the checkpoint or warm start device. If omitted, it defaults to the system residence volume.

VOLid &SYSRES
tells CP that the system residence device is to be used as the checkpoint or warm start device.

VOLid &SYSPARM
tells CP that the volume containing the active PARM disk is to be used as the checkpoint or warm start device.

FRom Cylinder *start*
FRom Page *start*
tells CP the real starting location (in cylinders or pages) where the checkpoint or warm start information is saved. For CKD device types, *start* is a nonzero decimal number from 1 to 65511. For FBA devices, *start* is a 1-to-6 digit 4 KB page number that is greater than 3 (pages 0 through 3 are reserved).

FOR *n*
defines the maximum number of cylinders or pages that CP should use for the checkpoint or warm start area. For CKD device types, *n* is a 1-digit decimal number from 1 to 9 that represents cylinders. For FBA device types, *n* is a decimal number from 1 to 2000 that represents 4 KB pages.

Usage Notes

1. The checkpoint and warm start areas must not overlap.
2. The system residence volume is the volume on which the CP module resides. This is usually the IPL volume, but may be another volume selected through the full screen loader.
3. The checkpoint and warm start areas must be on a CP_OWNED volume and must not reside on an NVMe EDEVICE.
4. Once you have IPLed the system with Checkpoint and warm start areas defined, you cannot move them without doing a CLEAN START, therefore you should allocate enough space to allow for expected growth.
5. The checkpoint and warm start areas can be extended (in the same place) if there is room without doing a CLEAN START. For example, if your current specification is:

   ```
   SYSTEM_RESIDENCE CHECKpoint volid ESARES from Cyl 100 for 5
   ```

   you could change it to:

   ```
   SYSTEM_RESIDENCE CHECKpoint volid ESARES from Cyl 100 for 9
   ```

   if cylinders 105 – 108 are not in use for other purposes.

Examples

1. To define the system residence volume with the:
   - Checkpoint area starting at cylinder 101 and occupying 2 cylinders on DASD SYS002 and
   - Warm start area starting at cylinder 204 and occupying 1 cylinder on DASD SYS001,

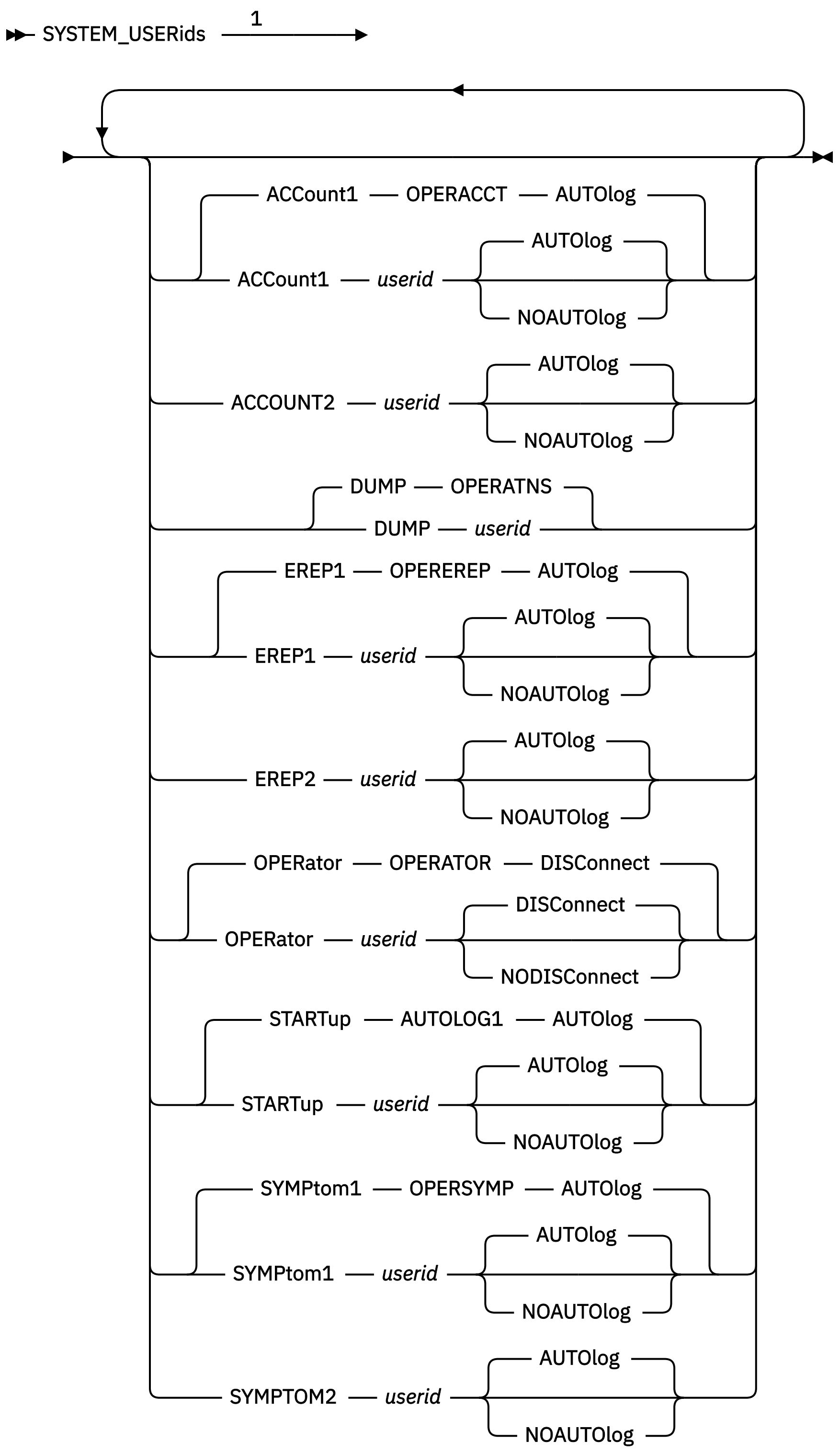
   specify the following SYSTEM_RESIDENCE statement:

   ```
     System_Residence,
        CheckPoint  VolID sys002  from Cylinder 101  for 2,
        WarmStart   VolID sys001  from Cylinder 204  for 1
   ```
2. To define the system residence volume with the:
   - Checkpoint area starting at cylinder 568 and occupying 9 cylinders on DASD ESARES and
   - Warm start area starting at cylinder 784 and occupying 9 cylinders on DASD ESARES,

   specify the following SYSTEM_RESIDENCE statement:

   ```
     System_Residence,
        WarmStart   VolID esares  from Cylinder 568  for 9,
        CheckPoint  VolID esares  from Cylinder 784  for 9
   ```

SYSTEM_USERIDS Statement


SYSTEM_USERids
1
ACCount1 OPERACCT AUTOlog
ACCount1 userid AUTOlog NOAUTOlog
ACCOUNT2 userid AUTOlog NOAUTOlog
DUMP OPERATNS
DUMP userid
EREP1 OPEREREP AUTOlog
EREP1 userid AUTOlog NOAUTOlog
EREP2 userid AUTOlog NOAUTOlog
OPERator OPERATOR DISConnect
OPERator userid DISConnect NODISConnect
STARTup AUTOLOG1 AUTOlog
STARTup userid AUTOlog NOAUTOlog
SYMPtom1 OPERSYMP AUTOlog
SYMPtom1 userid AUTOlog NOAUTOlog
SYMPTOM2 userid AUTOlog NOAUTOlog

Notes:

[1] You must specify at least one of the following operands.

Purpose

Use the SYSTEM_USERIDS statement to specify user IDs that will perform special functions during and after IPL. These functions include accumulating accounting records, system dump files, EREP records, and symptom records, and specifying the primary system operator's user ID and disconnect status.

How to Specify

Include as many statements as needed; they are optional. You can place SYSTEM_USERIDS statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

ACCount1 *userid*
ACCOUNT2 *userid*
: specifies the user IDs of the virtual machines for which the *ACCOUNT system service is to accumulate and checkpoint accounting records. You can specify one or two user IDs. One or both of these virtual machines are automatically logged on during CP initialization. The user IDs are alphanumeric character strings of up to eight characters. OPERACCT is the default for ACCOUNT1.

AUTOlog
NOAUTOlog
: specify whether CP should log on the indicated user ID at IPL.

DUMP *userid*
: is the user ID of the virtual machine that receives spooled system dumps. The variable *userid* must be an alphanumeric character string of as many as eight characters. OPERATNS is the default.

EREP1 *userid*
EREP2 *userid*
: are the user IDs of the virtual machines for which the *LOGREC system service is to accumulate and checkpoint error records. You can specify one or two user IDs. EREP1 can be shortened to EREP. The variable *userid* must be an alphanumeric string 1- to 8-characters long. These virtual machines are automatically logged on during CP initialization. OPEREREP is the default for EREP1.

OPERator *userid*
: is the primary system operator's user ID. The variable *userid* is an alphanumeric character string of as many as eight characters. The default is OPERATOR.

DISConnect
NODISConnect
: indicates whether CP should disconnect the primary system operator when a software-initiated IPL occurs and the primary system operator is not logged onto the primary system console.

If you specify DISCONNECT, and the primary system operator is not logged onto the primary system console when a software-initiated IPL occurs, CP disconnects the primary system operator. If you specify NODISCONNECT, CP does not force the primary system operator to be disconnected. If you do not code this operand, DISCONNECT is assumed.

Specifying NODISCONNECT may affect system security. For more information, see usage note "5" on page 296.

STARTup *userid*
: specifies the user ID that is to be logged on automatically at IPL to perform functions you select. AUTOLOG1 is the default.

SYMPtom1 *userid*
SYMPTOM2 *userid*
: specifies the user IDs of virtual machines for which the CP *SYMPTOM system service is to accumulate and checkpoint symptom records. One or two user IDs can be specified. The user IDs are alphanumeric strings of as many as eight characters each. OPERSYMP is the default for SYMPTOM and SYMPTOM1.

Usage Notes

1. If the user ID specified on the SYSTEM_USERIDS ACCOUNT statement has a virtual machine definition in the user directory, that user ID is logged on at system initialization.
2. CP creates system dump files as a result of system abends and system restarts. You can also use the VMDUMP command to send virtual machine dumps to the virtual machine you specify in the SYSTEM_USERIDS statement.
3. If the user ID specified on the SYSTEM_USERIDS EREP statement has a virtual machine definition in the user directory at system initialization, the associated virtual machine is logged on to the system.
4. CP logs on a system operator with the user ID you specify on the OPERATOR operand, regardless of whether that user ID has a virtual machine definition in the user directory.
5. If you specify NODISCONNECT, CP allows the primary system operator to remain logged onto the primary system console after all software-initiated IPLs. This makes the primary system operator's user ID available to any user with access to the primary system console, which could lead to a system-wide compromise of security. If you specify NODISCONNECT, make sure that the primary system console and all alternate system consoles are in a secure environment and are continuously monitored by authorized personnel.
6. When the primary system operator logs off, the next user that logs on with at least one of the privilege classes required for the system operator will become the primary system operator. However, this can be overridden by either:

   - Specifying alternate operators using the ALTERNATE_OPERATORS statement in the system configuration file
   - Selecting a new operator using the SET SYSOPER command.

   For more information, see "ALTERNATE_OPERATORS Statement" on page 56. See also SET OPER in *z/VM: CP Commands and Utilities Reference*.
7. If the user ID that is specified by the SYMPTOM1 or SYMPTOM2 operand has a virtual machine definition in the user directory at system initialization, the virtual machine that is designated by that user ID is logged on to the system.
8. Further consideration is required when you change some accounting records. For more information, see "Disassociating a User ID from the Retrieval of Accounting Records" on page 355, "Disassociating a User ID from the Retrieval of EREP Records" on page 357, and "Disassociating a User ID from the Retrieval of Symptom Records" on page 362.

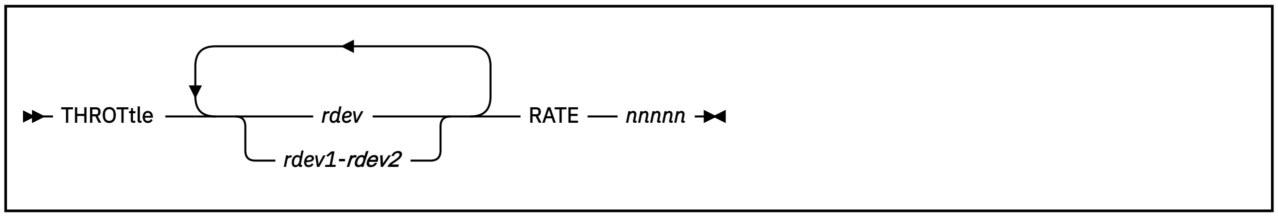
Examples

1. The following SYSTEM_USERIDS statement specifies:

   - The system operator as user ID OPERATOR and to specify that CP should disconnect the system operator when a software re-IPL occurs and the system operator is not logged on to the system console
   - Dumps should go to the OPERATNS user ID
   - CP should automatically log the AUTOLOG1 user ID at IPL
   - One EREP user ID called EREP and to have CP automatically log on that user ID at IPL
   - One accounting user ID called DISKACNT
   - One symptom records user ID called OPERSYMP

```
System_UserIDs,
   Operator  operator,       /* use OPERATOR ID as operator          */
                Disconnect,  /*  If bounce, disconnect oper          */
   Dump      operatns,       /* use OPERATNS ID for dump collection  */
   StartUp   autolog1,       /* CP will XAUTOLOG this ID during init */
   Erep      erep Autolog,   /* Collect I/O error & Send to EREP ID  */
   Account   diskacnt,       /* Send Accting Records to Diskacnt ID  */
   Symptom   opersymp        /* Send Symptom Records to Opersymp ID  */
```

THROTTLE Statement



Purpose

Use the THROTTLE statement to limit (throttle) the number of I/O operations that a guest operating system can initiate to a specific real device. This prevents a guest from interfering with or dominating I/O resources.

The THROTTLE statement adds real devices to the I/O throttling list. After initialization completes, CP begins limiting (throttling) the guest's I/O rate to the specified device or devices.

How to Specify

Include as many statements as needed; they are optional. You can place THROTTLE statements anywhere in the system configuration file. If you specify more than one statement with the same device number, CP uses the I/O rate on the last statement, and ignores all previous statements with that device number.

Operands

rdev
rdev1-rdev2

identifies the real device or devices that you want to throttle. The variable *rdev* is the real device number of the device and must be a hexadecimal number between X'0000' and X'FFFF'. You can specify a single device address, a range of device addresses, or any combination thereof.

RATE *nnnnn*

specifies the I/O throttling rate (number of I/O operations per second that this device can process from a guest operating system). The variable *nnnnn* must be a decimal number between 1 and 10,000.

Usage Notes

1. To add devices to or delete devices from the I/O throttling list after initialization, use the SET THROTTLE command. For more information, see SET THROTTLE in *z/VM: CP Commands and Utilities Reference*.
2. To display the list of devices being throttled and their respective I/O rates, use the QUERY THROTTLE command. For more information, see QUERY THROTTLE in *z/VM: CP Commands and Utilities Reference*.

Examples

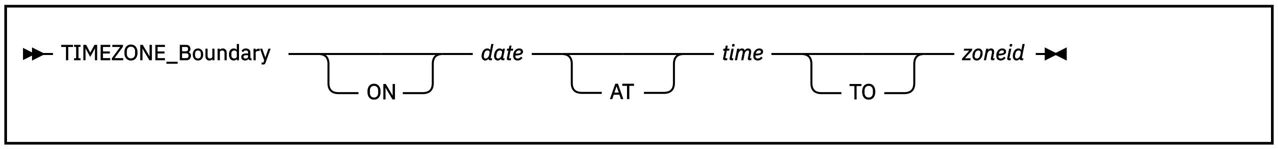
1. To limit all guest operating systems using 1c10 to 1 I/O operations per second, use the following THROTTLE statement:

```
Throttle 1c10 rate 1 */
```

During initialization, CP responds:

```
Device 1C10 added to throttle set. Rate = 1.
```

TIMEZONE_BOUNDARY Statement



Purpose

Use the TIMEZONE_BOUNDARY statement to tell CP what time zone to choose at IPL so that it can determine the local time.

How to Specify

Include as many statements as needed; they are optional. You can place TIMEZONE_BOUNDARY statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

ON *date*
: is the date, in the format yyyy-mm-dd, on which CP should begin using a given time zone.

AT *time*
: is the time, in the format *hh:mm:ss*, when CP should begin using a given time zone.

TO *zoneid*
: is a time zone definition that was established by an earlier TIMEZONE_DEFINITION statement.

Usage Notes

1. Before you can use this statement, you must already have defined the timezone to which you are changing. You can do so with TIMEZONE_DEFINITION statements in the system configuration file. For more information, see "TIMEZONE_DEFINITION Statement" on page 300.
2. As its name suggests, this statement establishes the boundaries for the time zones that CP can use. You must specify at least one boundary that has occurred before the current date; otherwise, CP will use Coordinated Universal Time (UTC) to determine the new time. For example, if you include the following statements in your system configuration file:

   ```
   TimeZone_Boundary  on 2007-03-11  at 02:00:00  to EDT
   TimeZone_Boundary  on 2007-11-04  at 02:00:00  to EST
   TimeZone_Boundary  on 2008-03-09  at 02:00:00  to EDT
   ```

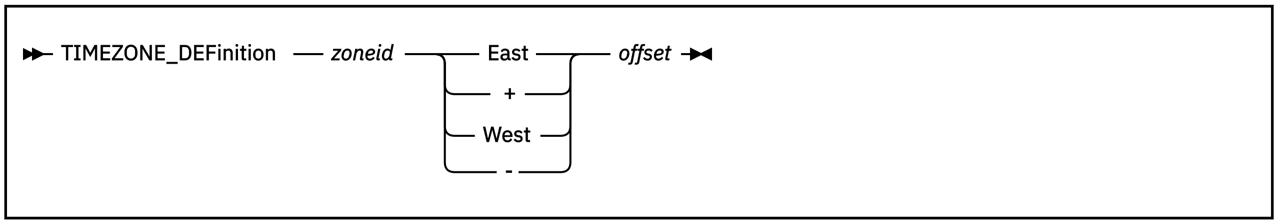
   CP determines the local time according to where the time of the system's clock fits in to the boundaries. If you IPL the system before 02:00:00 UTC on March 11, 2007, CP assumes that the local time is UTC, because nothing has told it otherwise. If you IPL between 02:00:00 UTC on March 11, 2007, and 02:00:00 EDT on November 04, 2007, CP assumes that EDT is the local time. If you IPL after 02:00:00 EDT on November 04, 2007, but before 02:00:00 EST March 09, 2008, CP takes EST as the local time; if you IPL any time after 02:00:00 EST on March 09, 2008, CP assumes that the local time is EDT. CP determines which time zone 02:00:00 is in from the preceding TIMEZONE_BOUNDARY statement. If there is no preceding statement, CP assumes that 02:00:00 is in UTC.
3. After the system is up and running, you can issue the CP SET TIMEZONE command to change the time zone of the running system. If you do so, you may only choose a time zone that has already been defined with either a TIMEZONE_DEFINITION statement or the CP DEFINE TIMEZONE command. For more information, see "TIMEZONE_DEFINITION Statement" on page 300. See also DEFINE TIMEZONE in *z/VM: CP Commands and Utilities Reference*.

Examples

1. To define eastern daylight time (EDT) and eastern standard time (EST) through the year 2009, use the following TIMEZONE_BOUNDARY statements:

```
/*---------- Timezone Boundary Condition Definitions ----------*/

TimeZone_Boundary  on 2006-04-02  at 02:00:00  to EDT
TimeZone_Boundary  on 2006-10-29  at 02:00:00  to EST

TimeZone_Boundary  on 2007-03-11  at 02:00:00  to EDT
TimeZone_Boundary  on 2007-11-04  at 02:00:00  to EST

TimeZone_Boundary  on 2008-03-09  at 02:00:00  to EDT
TimeZone_Boundary  on 2008-11-02  at 02:00:00  to EST

TimeZone_Boundary  on 2009-03-08  at 02:00:00  to EDT
TimeZone_Boundary  on 2009-11-01  at 02:00:00  to EST
```

TIMEZONE_DEFINITION Statement



Purpose

Use the TIMEZONE_DEFINITION statement to define system time zones according to their difference from Coordinated Universal Time (UTC).

How to Specify

Include as many statements as needed; they are optional. You can place TIMEZONE_DEFINITION statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

zoneid
is the time zone ID that you are defining. The variable *zoneid* must contain 1-4 alphanumeric characters.

Important Note
The zone IDs UTC and GMT are predefined. You cannot specify UTC or GMT as a *zoneid*.

East
+
tells CP to add the value specified by *offset* to Coordinated Universal Time (UTC) to define this time zone ID. EAST is the same as +.

West
-
tells CP to to subtract the value specified by *offset* from the Coordinated Universal Time (UTC) to define the time zone ID. WEST is the same as -.

offset
is the difference between UTC and *zoneid*. You can enter *offset* as *hh:mm:ss* or *hh.mm.ss*, where *hh* is required and *mm* and *ss* are optional. You can specify *ss* only if you have specified *mm*.

Usage Notes

1. This statement defines the time zones that will always be available after an IPL. You can use the CP DEFINE TIMEZONE command to define additional time zones after the system is up and running. For more information, see DEFINE TIMEZONE in *z/VM: CP Commands and Utilities Reference*.

Examples

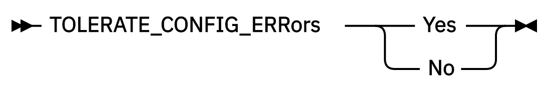
1. To define time zone:
   - EDT as 4 hours west of UTC
   - EST as 5 hours west of UTC

- DRB as 5 ½ hours west of UTC
- RFC as 4 ½ hours west of UTC
- IKA as 3 ½ hours west of UTC
- DAC as 4 ½ hours east of UTC
- PDT as 7 hours west of UTC
- PST as 8 hours west of UTC

use the following TIMEZONE_DEFINITION statements:

```
TimeZone_Definition  edt  West  4.00.00    /* EDT is West 4 from UTC */
TimeZone_Definition  est  West  5.00.00    /* EST is West 5 from UTC */
TimeZone_Definition  drb  West  5.30.00    /* DRB is West 5 hrs, 30  */
                                           /*    minutes from UTC    */
TimeZone_Definition  rfc  West  4.30.00    /* RFC is West 4 hrs, 30  */
                                           /*    minutes from UTC    */
TimeZone_Definition  ika  West  3.30.00    /* IKA is West 3 hrs, 30  */
                                           /*    minutes from UTC    */
TimeZone_Definition  dac  East  4.30.00    /* DAC is East 4 hrs, 30  */
                                           /*    minutes from UTC    */
TimeZone_Definition  pdt  West  7.00.00    /* PDT is 7 West of UTC   */
TimeZone_Definition  pst  West  8.00.00    /* PST is 8 West of UTC   */
```

TOLERATE_CONFIG_ERRORS Statement



Purpose

Use the TOLERATE_CONFIG_ERRORS statement to mark sections of the system configuration file that must be without errors when CP processes them.

How to Specify

Include as many statements as needed; they are optional. You can place TOLERATE_CONFIG_ERRORS statements anywhere in the system configuration file.

If you do not specify any TOLERATE_CONFIG_ERRORs statements, CP will tolerate errors in your system configuration file.

The first statement you specify must be TOLERATE_CONFIG_ERRORS NO. All subsequent statements must alternate between YES and NO. You cannot have two TOLERATE_CONFIG_ERRORS statements in a row with the same operand. That is, you cannot tell CP to tolerate errors, process a few more statements, and then tolerate errors again.

Operands

Yes
: tells CP to begin tolerating errors in the system configuration file.

No
: tells CP not to tolerating any errors in the system configuration file until encountering a TOLERATE_CONFIG_ERRORS YES statement.

Usage Notes

1. When CP finds an error in a system configuration file statement that follows a TOLERATE_CONFIG_ERRORS NO statement, it remembers the error occurrence and, after displaying all the errors encountered, prompts the operator with a question. The operator can respond to the question in one of the following ways:
   - Stopping the IPL process
   - Continuing with the normal IPL process
   - Continuing with the normal IPL process but not automatically logging on any virtual machines.
2. If CP does not find any errors while processing the system configuration file or if all errors found are in sections of the system configuration file where TOLERATE_CONFIG_ERRORS YES is in effect, the normal IPL process continues.
3. If you are using the IMBED and the TOLERATE_CONFIG_ERRORS statements, remember that CP processes the statements in an imbed file as if those statements were included in the master system configuration file. If you are not tolerating errors in the master file and then repeat the statement in the imbedded file, CP will flag the duplicate statement in the imbedded file as an error.
4. If you change a system configuration file, you should run the CPSYNTAX exec to make sure there are no syntax errors in the system configuration file.

Examples

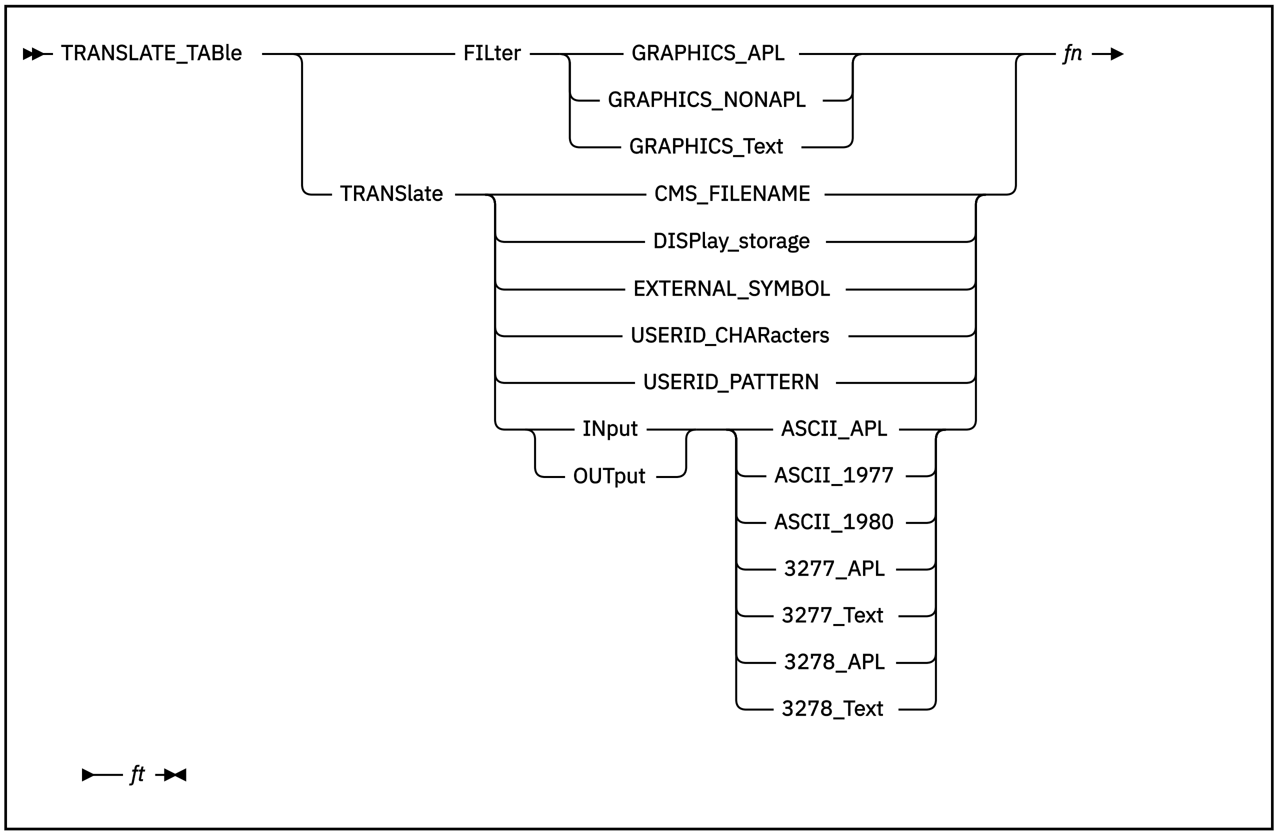
1. To tell CP not to ignore any syntax problems found in the processing of the CP_OWNED statements, use the following TOLERATE_CONFIG_ERRORS statements:

```
Tolerate_Config_Errors  no             /* Don't Ignore errors here */
 CP_Owned  Slot 1 ESARES
 CP_Owned  Slot 2 ESAP01
 CP_Owned  Slot 3 ESAP02
 CP_Owned  Slot 4 ESAP03
 CP_Owned  Slot 5 ESAP04
 CP_Owned  Slot 6 ESAP05
 Tolerate_Config_Errors  yes     /* Back to normal mode      */
```

2. To make sure CP does not ignore any errors when defining your checkpoint and warm start areas, use the following TOLERATE_CONFIG_ERRORS statements:

```
Tolerate_Config_Errors  no             /* Don't Ignore Any Errors in */
                                       /* the System Residence Stmt  */
System_Residence,
   CheckPoint  VolID  esares  From  Cylinder  784  For  9,
   WarmStart   VolID  esares  From  Cylinder  568  For  9
Tolerate_Config_Errors  yes           /* From here on, errors are OK */
```

TRANSLATE_TABLE Statement



Purpose

Use TRANSLATE_TABLE statements to specify replacements for the standard translation tables that CP uses to accomplish certain tasks.

How to Specify

Include as many statements as needed; they are optional. You can place TRANSLATE_TABLE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

FILter
tells CP to use the specified file to filter out invalid characters.

GRAPHICS_APL
tells CP to use this translation table to filter out invalid characters and 3270 orders out of a 3270 data stream destined for an APL display.

GRAPHICS_NONAPL
tells CP to use this translation table to filter invalid characters out of a 3270 data stream destined for an EBCDIC (non-APL) display.

GRAPHICS_Text
tells CP to use this translation table to filter out invalid characters and 3270 orders out of a 3270 data stream destined for an 3270 text display.

TRANSlate
tells CP to use the specified file to translate input or output characters.

CMS_FILENAME
defines which characters are acceptable to use when defining CMS file names and file types.

DISPlay_storage
tells CP to use the hexadecimal-to-EBCDIC translation table when issuing the CP DISPLAY command with the T option or the CPTYPE command. For more information, see DISPLAY and CPTYPE in *z/VM: CP Commands and Utilities Reference*.

EXTERNAL_SYMBOL
defines which characters are acceptable to use when defining entry point names.

USERID_CHARacters
defines the translation test table used to validate user IDs in the system configuration file. This statement overrides the table in HCPTBL. Indicate valid characters by setting their positions in the translation test table to X'00'. A non-zero value means that the character is not valid in a user ID.

If you change the USERID_CHARACTERS table, make corresponding changes to the USERID_PATTERN table. For more information, see "USER Directory Statement" on page 616 and "IDENTITY Directory Statement" on page 510.

If the translation test table you define invalidates any characters from the default acceptable set:

- Uppercase letters (A through Z)
- Numbers (0 through 9)
- Other (# $ @ _ -)

then results are unpredictable.

USERID_PATTERN
defines the translation test table used to validate user ID patterns in the system configuration file. This statement overrides the table in HCPTBL. Indicate valid characters by setting their positions in the translation test table to X'00'. Any non-zero value means that the character is not valid in a user ID pattern.

If you change the USERID_PATTERN table, make corresponding changes to the USERID_CHARACTERS table. Make sure the USERID_PATTERN table includes the pattern-matching characters asterisk (*) and percent (%) as valid characters.

INput
OUTput
tells CP to use the specified file to translate input or output characters.

ASCII_1977
defines the translation table used to translate input characters coming from a TTY UASCII-1977 level display to EBCDIC, or output EBCDIC characters to those going to a TTY UASCII-1977 terminal.

ASCII_1980
defines the translation table used to translate input characters coming from a TTY UASCII-1980 level display to EBCDIC, or output EBCDIC characters to those going to a TTY UASCII-1980 terminal.

ASCII_APL
defines the translation table used to translate ASCII APL input characters coming from an ASCII APL-capable display to EBCDIC APL characters, or to translate EBCDIC APL characters to those going to an ASCII APL-capable display.

3277_APL
defines the translation table used to translate input characters coming from a 3277 APL display, or output characters going to such a display.

3278_APL
defines the translation table used to translate input characters coming from a 3278 APL display, or output characters going to such a display.

3277_Text
defines the translation table used to translate input characters coming from a text-only 3277 display, or output characters coming from such a display.

3278_Text
defines the translation table used to translate input characters coming from a text-only 3278 display, or output characters going to such a display.

fn
is the file name of the translation table file.

ft
is the file type of the translation table file.

Usage Notes

1. The specified file must reside on the same minidisk as the system configuration file so that CP can find the file during IPL processing.
2. As with the system configuration file, comments in the file containing the translation table must be delimited with a beginning /* and an ending */. A single comment may span multiple lines in the file.
3. Data found outside the aforementioned comment delimiters must be EBCDIC hexadecimal. Because each byte in the translation table is represented by 2 hexadecimal digits, no token in the file can consist of an odd number of characters. Translated characters may be grouped in clusters to improve the readability of the table.
4. Exactly 256 translated characters (512 data bytes) must exist in the file. If the number of characters specified does not equal 256, CP will generate an error and otherwise ignore the statement.

Examples

1. To display the underscore character (X'6D') as an underscore character rather than the period (X'4B') defined in the default translation table used by the DISPLAY command, use the following:

```
  /* Use the following translate table located in file,     */
  /*    DISPLAY STORAGE, as the hexadecimal to EBCDIC       */
  /*    translate table.                                    */

     Translate_Table  Translate  Display_Storage  display storage
```

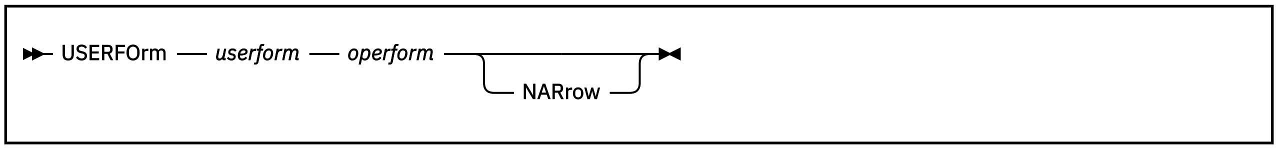
   On the same minidisk as the system configuration file, you would put a file called DISPLAY STORAGE containing the following information:

```
/*-----------------------------------------*
 * Translate table for display of storage *
 *-----------------------------------------*/

  4B4B4B4B 4B4B4B4B  4B4B4B4B 4B4B4B4B  /* .... ....  .... ....
*/
  4B4B4B4B 4B4B4B4B  4B4B4B4B 4B4B4B4B  /* .... ....  .... ....
*/
  4B4B4B4B 4B4B4B4B  4B4B4B4B 4B4B4B4B  /* .... ....  .... ....
*/
  4B4B4B4B 4B4B4B4B  4B4B4B4B 4B4B4B4B  /* .... ....  .... ....
*/
  404B4B4B 4B4B4B4B  4B4B4B4B 4C4D4E4F  /* .... ....  .... <(+|
*/
  504B4B4B 4B4B4B4B  4B4B4B5B 5C5D5E5F  /* &;.. ....  ...$
*);* */
  60614B4B 4B4B4B4B  4B4B4B6B 6C6D6E6F  /* -/.. ....  ..., %_>?
*/
  4B4B4B4B 4B4B4B4B  4B4B7A7B 7C7D7E7F  /* .... ....  ..
# @'=" */
  4B818283 84858687  88894B4B 4B4B4B4B  /* .abc defg  hi.. ....
*/
  4B919293 94959697  98994B4B 4B4B4B4B  /* .jkl mnop  qr.. ....
*/
  4B4BA2A3 A4A5A6A7  A8A94B4B 4B4B4B4B  /* ..st uvwx  yz.. ....
*/
```

```
   4B4B4B4B 4B4B4B4B  4B4B4B4B 4B4B4B4B  /* .... ....  .... ....
 */
   4BC1C2C3 C4C5C6C7  C8C94B4B 4B4B4B4B  /* .ABC DEFG  HI.. ....
 */
   4BD1D2D3 D4D5D6D7  D8D94B4B 4B4B4B4B  /* .JKL MNOP  QR.. ....
 */
   4B4BE2E3 E4E5E6E7  E8E94B4B 4B4B4B4B  /* ..ST UVWX  YZ.. ....
 */
   F0F1F2F3 F4F5F6F7  F8F94B4B 4B4B4B4B  /* 0123 4567  89.. ....
 */
```

CP reads the DISPLAY STORAGE file at IPL time and would replace the default table used with the CP DISPLAY or CPTYPE commands with the table specified in the file.

USERFORM Statement



Purpose

Use the USERFORM statement to create a list of user form names and their corresponding operator form numbers, and to specify forms as NARROW so that a narrow separator page is printed.

How to Specify

Include as many statements as needed; they are optional. You can place USERFORM statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

userform
is a 1- to 8-character alphanumeric name of a user form.

operform
is a 1- to 8-character alphanumeric operator form number.

NARrow
tells CP that the form is no more than 86 columns wide. If you specify NARROW, separator information does not print beyond column 86.

Usage Notes

1. The *userform* and *operform* operands establish the correspondence between the user form and operator form. If you do not specify any user form and operator form pairs, CP does not generate a form list and does not distinguish between user and operator forms.

Examples

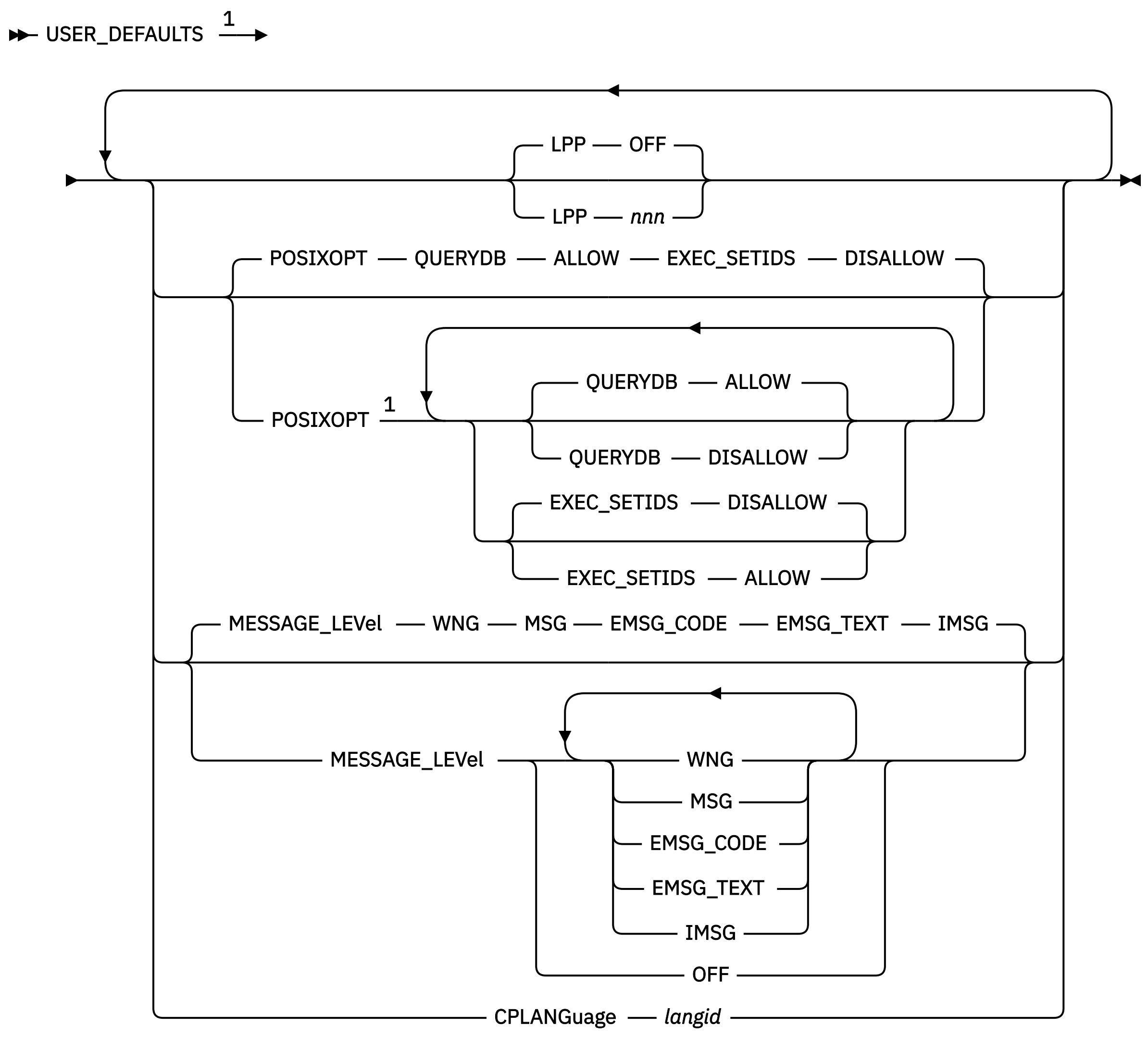
1. To define three user form and operator form pairs with the following characteristics:

   - User form DOCUMENT paired with operator form G1PN which are at most 86 characters wide
   - User form LISTING paired with operator form G1PW
   - User form PLAIN paired with operator form G1PN which are at most 86 characters wide

   use the following USERFORM statements:

   ```
   Userform  document  g1pn  Narrow
   Userform  listing   g1pw
   Userform  plain     g1pn  Narrow
   ```

USER_DEFAULTS Statement



Notes:

[1] You must specify at least one operand.

Purpose

Use the USER_DEFAULTS statement to define default attributes and permissions for all users on the system.

How to Specify

Include as many statements as needed; they are optional. You can place the USER_DEFAULTS statement anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

LPP
sets the default global lines per page value for all virtual printers and virtual consoles defined on the system.

The lines per page value is used by VM functions which generate virtual printer and console output. For CP, these include console spooling, SPXTAPE log files, CP dump and trace output, DDR, and load maps generated by HCPLDR. CMS functions which use the lines per page value for virtual printer output include the PRINT, ASSEMBLE, UPDATE, GENMSG, TAPE, FILEPOOL FORMAT AUDIT, and SET DOSLNCNT commands.

LPP OFF
indicates that internal defaults will be used as the global value for lines per page. The defaults are the same as those which have been used on prior releases of VM for virtual printers and consoles.

LPP *nnn*
sets the default global lines per page value to *nnn*. *nnn* must be a decimal number with a range of 30 to 255.

POSIXOPT
defines the default POSIX authorizations to permit or prohibit all users on the system from querying other users' POSIX information or having their POSIX security values changed.

These authorizations can be overridden for a specific user by adding the POSIXOPT directory statement to that users' virtual machine definition. For more information, see "POSIXOPT Directory Statement" on page 588.

QUERYDB ALLOW
(the default) specifies by default, users are permitted to query other users' POSIX database information.

QUERYDB DISALLOW
specifies that by default, users are not permitted to query other users' POSIX database information.

EXEC_SETIDS DISALLOW
(the default) specifies by default, users are not permitted to have their POSIX security values changed on behalf of a POSIX *exec()* function call designating a file with the *setuid* or *setgid* attributes.

EXEC_SETIDS ALLOW
specifies that by default, users are permitted to have their POSIX security values changed on behalf of a CP *exec()* processing. The server requesting the change must be authorized to do so by the SETIDS option of the POSIXOPT directory statement.

MESSAGE_LEVEL
sets the default message level for a user who accesses the system, overriding the default of WNG ON, MSG ON, and IMSG ON. At least one of the following message levels must be specified:

WNG
enables the user to receive messages from other users issued by the WARNING or WNG command.

MSG
enables the user to receive messages from other users issued by the MESSAGE, MSG, or MSGNOH command.

EMSG_CODE
enables the user to receive the message code portion of error messages.

EMSG_TEXT
enables the user to receive the text portion of error messages.

IMSG
enables the user to receive informational messages

OFF
prevents the user from receiving any messages, warnings, error messages, or informational messages.

CPLANGUAGE *langid*
langid is the system default language. *langid* must be 1 to 5 characters and must consist of CMS file system characters only. If the CPLANGUAGE operand is not specified on any USER_DEFAULTS statement, CP uses the language that was built into the CP nucleus as the system default language.

Usage Notes

1. If the USER_DEFAULTS statement is not specified, or the POSIXOPT operand is not specified, the defaults are to permit all users to query other users' POSIX database information, and prohibit all users from having their POSIX security values changed on behalf of CP *exec()* processing. ESM (External Security Manager) may override each user's settings.
2. If the USER_DEFAULTS statement is not specified, or the LPP operand is not specified, the default global value for lines per page is 'OFF'.
3. The LPP option of the SPOOL command may be used to change the LPP value for individual virtual printers and consoles.

Examples

1. The following example shows how to specify the USER_DEFAULTS statement. Note that you can specify more than one operand on a single USER_DEFAULTS statement.

   To specify that:

   - All users in the system can not query other users' POSIX database information.
   - All users in the system are prohibited from having their POSIX security values changed by other (authorized) users.

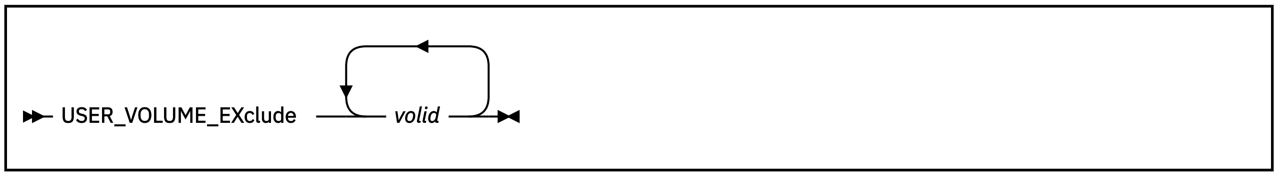
   use the following USER_DEFAULTS statement:

   ```
   User_Default posixopt Querydb disallow Exec_Setids disallow
   ```
2. To specify that the default lines per page value for all virtual printers and virtual consoles is 56, use the following USER_DEFAULTS statement:

   ```
   User_Defaults lpp 56
   ```
3. To specify that internal defaults should be used (as in prior releases of VM) as the lines per page value for all virtual printers and virtual consoles, use the following USER_DEFAULTS statement:

   ```
   User_Defaults lpp OFF
   ```

USER_VOLUME_EXCLUDE Statement



Purpose

Use the USER_VOLUME_EXCLUDE statement to define volumes that are to be excluded from the user volume list. (User volumes are those volumes that you use to contain minidisks.) During system IPL, CP attaches to the system any volumes whose volume identifiers match a generic volume identifier specified in a USER_VOLUME_INCLUDE statement unless the volume is also specified in a USER_VOLUME_EXCLUDE statement.

How to Specify

The USER_VOLUME_EXCLUDE statement is optional; if specified, you can include as many statements as you need as long as the total number of volumes specified on a single statement does not exceed 500. If you specify a generic volume identifier, CP counts each volume that matches the pattern as one.

You can place USER_VOLUME_EXCLUDE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

volid

is a volume serial number or a generic volume serial number for a specific subset of volumes. The variable *volid* is a 1- to 6-character alphanumeric string with no imbedded blanks. A generic volume serial number is a 1- to 6-character string with asterisks (*) in place of one or more arbitrary characters and percent signs (%) in place of exactly one arbitrary character. For example:

```
User_Volume_Exclude  es%vm*
```

creates a list that includes all volumes whose volume serial number start with ES and have VM as their fourth and fifth characters.

A single asterisk (*) prevents CP from attaching all user volumes from the system that are also specified on any USER_VOLUME_EXCLUDE statements.

Usage Notes

1. If a volume identifier matches a *volid* pattern specified both in USER_VOLUME_EXCLUDE and USER_VOLUME_INCLUDE statements, CP does not attach the volume to the system during initialization. A duplicate *volid* causes the entire statement to be rejected. A volume is automatically attached to the system if:

   It is explicitly listed in the USER_VOLUME_LIST statement
   It is explicitly listed in the USER_VOLUME_INCLUDE statement
   It satisfies a USER_VOLUME_INCLUDE pattern and does not satisfy a USER_VOLUME_EXCLUDE statement.

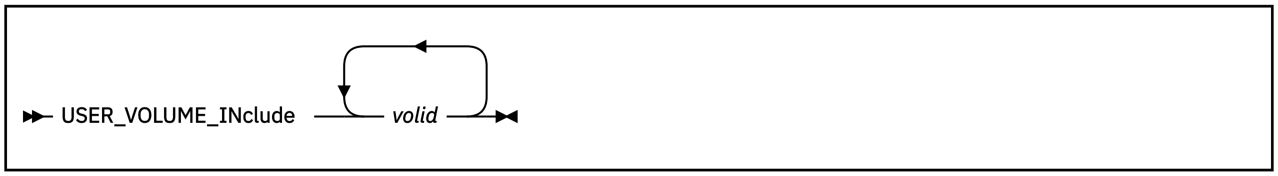
2. Dedicated devices are not user volumes. They must be attached to the user to which they are dedicated and not to the system. Therefore, if you specify USER_VOLUME_INCLUDE statements, you may want to specify dedicated devices in USER_VOLUME_EXCLUDE statements.

Examples

1. To exclude volume XAUSRF from the user volume list, use the following USER_VOLUME_EXCLUDE statement:

```
                             /* Exclude the following User Volume    */
User_Volume_Exclude  xausrf  /* from the System User Volume List.    */
                             /* NOTE: a previous USER_VOLUME_INCLUDE */
                             /* XAUSR* would have caused this volume */
                             /* to be mounted.                       */
```

USER_VOLUME_INCLUDE Statement



Purpose

Use the USER_VOLUME_INCLUDE statement to define user volumes by a generic volume identifier and to include those volumes in the user volume list. (User volumes are those volumes that you use to contain minidisks.) During system IPL, CP attaches to the system any volumes whose volume identifiers match a generic volume identifier specified in a USER_VOLUME_INCLUDE statement. A volume must be attached to the system before users can link to minidisks it contains.

How to Specify

The USER_VOLUME_INCLUDE statement is optional; if specified, you can include as many statements as you need as long as the total number of volumes specified on a single statement does not exceed 500. If you specify a generic volume identifier, CP counts each volume that matches the pattern as one.

You can place USER_VOLUME_INCLUDE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

volid

is a volume serial number or a generic volume serial number for a specific subset of volumes. The variable *volid* is a 1- to 6-character alphanumeric string with no imbedded blanks. A generic volume serial number is a 1- to 6-character string with asterisks (*) in place of one or more arbitrary characters and percent signs (%) in place of exactly one arbitrary character. For example:

```
User_Volume_Include  es%vm*
```

creates a list that includes all volumes whose volume serial number start with ES and have VM as their fourth and fifth characters.

A single asterisk (*) attaches all volumes to the system.

Usage Notes

1. You can specify volumes that you do not want defined as user volumes by specifying them on USER_VOLUME_EXCLUDE statements. For more information, see “USER_VOLUME_EXCLUDE Statement” on page 312.
2. If a volume identifier matches a generic volume identifier specified in both USER_VOLUME_INCLUDE and USER_VOLUME_EXCLUDE statements, CP does not attach the volume to the system during initialization. A duplicate *volid* causes the entire statement to be rejected. A volume is automatically attached to the system if:

   It is explicitly listed in the USER_VOLUME_LIST statement

   It is explicitly listed in the USER_VOLUME_INCLUDE statement

   It satisfies a USER_VOLUME_INCLUDE pattern and does not satisfy a USER_VOLUME_EXCLUDE statement.
3. If a volume is normally attached to the system using a USER_VOLUME_INCLUDE statement, CP does not notify the operator if the volume is not mounted. If a user volume is necessary for normal system

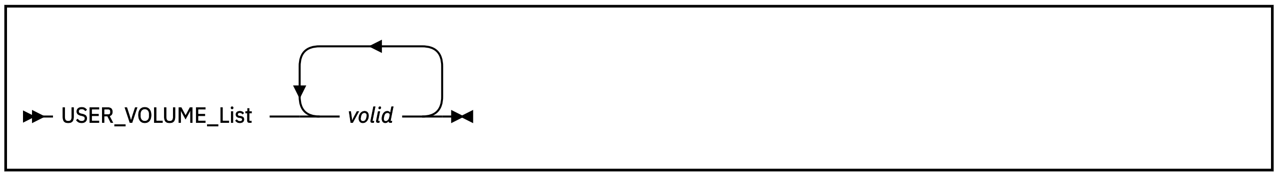
operation, specify it with a USER_VOLUME_LIST statement so that the operator is notified during system initialization that the volume is not mounted. For more information, see "USER_VOLUME_LIST Statement" on page 316.

Examples

1. To include all volumes whose first 5 characters are XAUSR in the user volume list, use the following USER_VOLUME_INCLUDE statement:

```
                               /* Include the following User Volumes */
User_Volume_Include  xausr*    /* on the System User Volume List..   */
                               /* So, any volume starting with       */
                               /* XAUSR* will be mounted.            */
```

USER_VOLUME_LIST Statement



Purpose

Use the USER_VOLUME_LIST statement to generate a list of user DASD volumes. (User volumes are those volumes you use to contain minidisks.) At system IPL, CP attaches the specified user volumes to the system. A volume must be attached to the system before users can link to minidisks it contains.

How to Specify

The USER_VOLUME_LIST statement is optional; if specified, you can include as many statements as you need as long as the total number of volumes specified on a single statement does not exceed 500.

You can place USER_VOLUME_LIST statements anywhere in the system configuration file.

Operands

volid
is the volume serial number of the DASD that you want included in the user volume list. The variable *volid* is a 1- to 6-character string where each character can be a letter, number, or hyphen.

Usage Notes

1. If you do not specify user volumes in any USER_VOLUME_LIST statements, you can still attach a volume to a virtual machine or system using the CP ATTACH command. For more information, see ATTACH in *z/VM: CP Commands and Utilities Reference*.
2. If you specify a volume on a USER_VOLUME_LIST statement that is not mounted when you load CP, CP considers that volume unavailable. CP notifies the operator that the volume is not mounted and continues processing, if possible. The system operator can mount and attach the volume to the system when it is needed using the CP ATTACH command. This provision leaves space for future growth.
3. You can also define user volumes by specifying a generic volume identifier. To do this, use the USER_VOLUME_INCLUDE statement. For more information, see "USER_VOLUME_INCLUDE Statement" on page 314.
4. At system initialization (IPL), if more than one DASD volume has the same serial number (*volid*), and that *volid* is in the CP-owned or user volume list, and no *rdev* has been specified for that *volid*, the volume with the lowest device number is attached to the system. This rule does not apply to duplicates of the system residence volume (the volume containing the minidisk from which the CP nucleus module was read).
5. Specifying a *volid* that has been specified previously on a USER_VOLUME_RDEV or USER_VOLUME_LIST statement causes this statement to be rejected.
6. A volume is automatically attached to the system if:

   It is explicitly listed in a USER_VOLUME_RDEV statement
   It is explicitly listed in a USER_VOLUME_LIST statement
   It is explicitly listed in a USER_VOLUME_INCLUDE statement
   It satisfies a USER_VOLUME_INCLUDE pattern and does not satisfy a USER_VOLUME_EXCLUDE statement.

Examples

1. To add volumes USPK1, USPK2, USPK3, USPK4, and USPK5 to the user DASD volume list, use the following USER_VOLUME_LIST statements:

```
User_Volume_List  uspk1        /* Attach the following disks to the  */
User_Volume_List  uspk2        /*   system at IPL time.  These disks */
User_Volume_List  uspk3        /*   contain user minidisks.  They    */
User_Volume_List  uspk4        /*   will be attached to the system   */
User_Volume_List  uspk5        /*   IPL time.                        */
```

USER_VOLUME_RDEV Statement

```
►► USER_VOLUME_Rdev — volid — RDEV — rdev ►◄
```

Purpose

Use the USER_VOLUME_RDEV statement to specify a user DASD volume at a specific real device number. (User DASD volumes are those volumes you use to contain minidisks.) At system IPL, if the device at the specified RDEV address is labeled with the specified volume ID, CP attaches the volume to the system. A volume must be attached to the system before users can link to minidisks it contains.

How to Specify

The USER_VOLUME_RDEV statement is optional; if specified, you can include as many statements as you need. Each statement may specify only one volume.

You can place USER_VOLUME_RDEV statements anywhere in the system configuration file.

Operands

volid
is the volume serial number of the DASD that you want included in the user volume list. The variable *volid* is a 1- to 6-character string where each character can be a letter, number, or hyphen.

RDEV *rdev*
specifies the real device address of the DASD volume you want included in the user volume list. The variable *rdev* is a 1- to 4-character hexadecimal device address.

Usage Notes

1. If you do not specify user volumes in any USER_VOLUME_RDEV statements, you can still attach a volume to a virtual machine or system using the CP ATTACH command. For more information, see ATTACH in *z/VM: CP Commands and Utilities Reference*.
2. If you specify a volume on a USER_VOLUME_RDEV statement that is not mounted when you load CP, CP considers that volume unavailable. CP notifies the operator that the volume is not mounted and continues processing, if possible. The system operator can mount and attach the volume to the system when it is needed using the CP ATTACH command. This provision leaves space for future growth.
3. You can also define user volumes by specifying a generic volume identifier. For more information, see "TRANSLATE_TABLE Statement" on page 304.
4. If the DASD volume at the specified RDEV address has a different volume ID than specified on the USER_VOLUME_RDEV statement, or if no DASD volume exists at the specified RDEV address, no volume with the specified volume ID is attached.
5. At system initialization (IPL), if more than one DASD volume has the same serial number (*volid*), and that *volid* is in the CP-owned or user volume list, and no *rdev* has been specified for that *volid*, the volume with the lowest device number is attached to the system. This rule does not apply to duplicates of the system residence volume (the volume containing the minidisk from which the CP nucleus module was read).
6. Specifying a *volid* that has been specified previously on a USER_VOLUME_RDEV or USER_VOLUME_LIST statement causes this statement to be rejected.
7. Using the USER_VOLUME_RDEV statement instead of the USER_VOLUME_LIST statement provides a convenient method for managing a case where there might be multiple DASD defined to the system that have the same volume ID, and the device with the higher device number is the one that should be attached. This eliminates the need to set the devices OFFLINE_AT_IPL and then attach the devices after the IPL process has completed.

8. A volume is automatically attached to the system if:

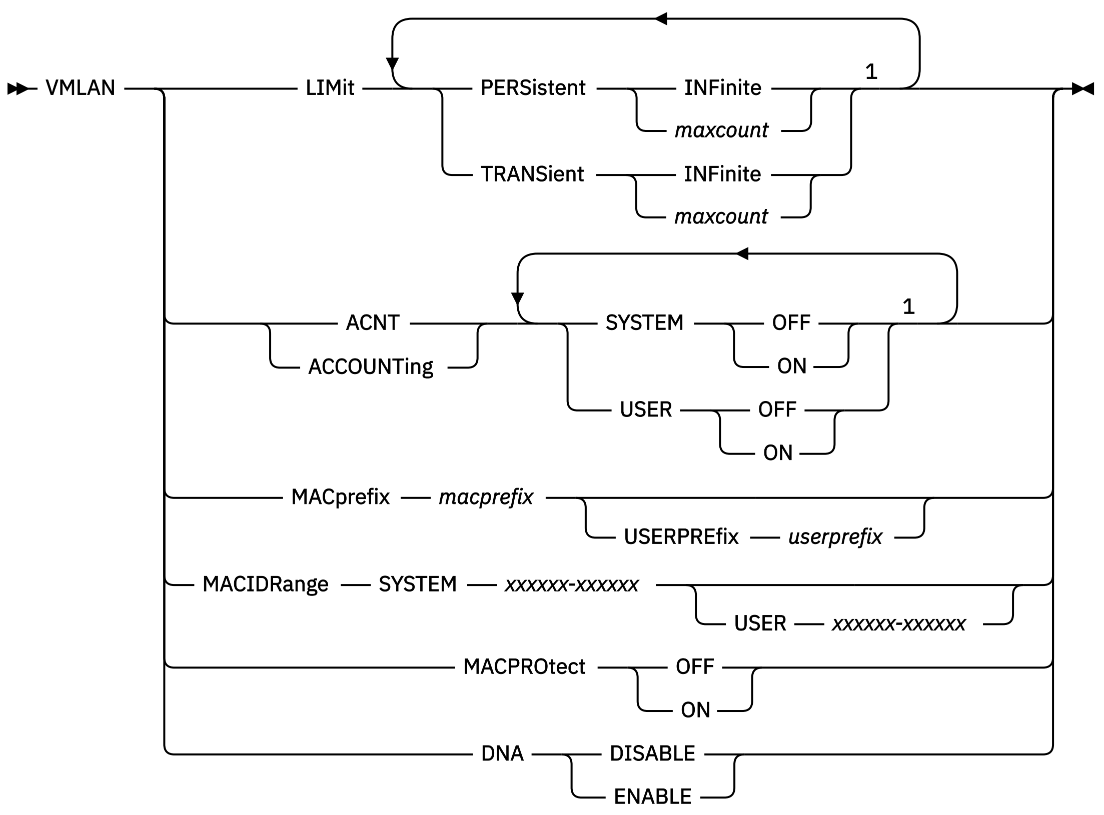
   It is explicitly listed in a USER_VOLUME_RDEV statement
   It is explicitly listed in a USER_VOLUME_LIST statement
   It is explicitly listed in a USER_VOLUME_INCLUDE statement
   It satisfies a USER_VOLUME_INCLUDE pattern and does not satisfy a USER_VOLUME_EXCLUDE statement.

Examples

1. To add volumes USPK7 and USPK8, at device addresses 0554 and 0F23 respectively, to the user DASD volume list, use the following USER_VOLUME_RDEV statements:

```
User_Volume_Rdev  uspk7 RDEV 0554 /* Attach the following disks to the */
User_Volume_Rdev  uspk8 RDEV 0F23 /*   system at IPL time.             */
```

VMLAN Statement



Notes:

[1] You can specify the operands in any order, but you can specify an operand only once.

Purpose

Use the VMLAN statement to control global attributes for virtual and real networking support.

How to Specify

Include as many statements as needed; they are optional. You can place VMLAN statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

PERSistent INFinite
: indicates the number of PERSISTENT guest LANs and virtual switches allowed on the CP Host system as infinite. This is the default.

PERSistent *maxcount*
: is the maximum number of PERSISTENT guest LANs and virtual switches allowed on the CP Host system at one time. The value *maxcount* must be a decimal number in the range 0 - 1024. If PERSistent *maxcount* is not specified, the system default is INFinite.

  A persistent LAN is created when the DEFINE LAN statement is used to create a LAN during system initialization, or when a Class B user issues the DEFINE LAN command to define a SYSTEM owned LAN. A persistent virtual switch is created when the DEFINE VSWITCH statement is used to create the switch during system initialization, or when a Class B user issues the DEFINE VSWITCH command to define a virtual switch.

TRANSient INFinite
indicates the number of TRANSIENT guest LANs allowed on the CP Host system as infinite. This is the default.

TRANSient *maxcount*
is the maximum number of TRANSIENT guest LANs allowed on the CP Host system at one time. The value *maxcount* must be a decimal number in the range 0 - 1024. If TRANSient *maxcount* is not specified, the system default is INFinite.

A transient LAN is created when the DEFINE LAN command is used to define a general user LAN.

ACNT|ACCOUNTing SYSTEM ON|OFF
sets the default accounting state for guest LANs owned by the SYSTEM user ID. The default state of this attribute is **OFF**. Class B users are authorized to override this setting on the **DEFINE LAN** and **SET LAN** commands.

ACNT|ACCOUNTing USER ON|OFF
sets the default accounting state for guest LANs owned by individual users. The default state of this attribute is **OFF**. Class B users are authorized to override this setting on the **DEFINE LAN** and **SET LAN** commands.

MACprefix *macprefix*
specifies the 3-byte prefix (manufacturer ID) used when CP automatically generates locally administered MAC addresses on the system. It must be 6 hexadecimal digits in the range 020000 - 02FFFF. In combination with the MAC ID automatically assigned by CP, the MACPREFIX allows unique identification of virtual adapters within a network. If MACPREFIX is not specified, the default is 020000 (02-00-00).

In an SSI cluster, the MACPREFIX value must be different for each member system and cannot be the same as the USERPREFIX value for the cluster. The default MACPREFIX for a cluster member is 02*xxxx* (02-*xx*-*xx*), where *xxxx* is the slot number assigned to the system in the SSI member list on the SSI configuration statement.

USERPREfix *userprefix*
specifies the 3-byte prefix used when user-defined locally administered MAC addresses are generated. It must be 6 hexadecimal digits in the range 020000 - 02FFFF.

A user-defined MAC address is an explicit MAC assignment made from the user directory for a network device. When a persistent MAC address is required, the last 6 hexadecimal digits of the address are specified on the MACID operand of the NICDEF directory statement. The USERPREFIX and MACID values are concatenated to make up the 12 hexadecimal digits of the MAC address assigned to the network device. If USERPREFIX is not specified, the default is set to the MACPREFIX value.

In an SSI cluster, the USERPREFIX value must be identical for all member systems and cannot be the same as the MACPREFIX value for any member. The default is 020000 (02-00-00).

MACIDRange SYSTEM *xxxxxx-xxxxxx* USER *xxxxxx-xxxxxx*
is the range of identifiers (up to 6 hexadecimal digits each) to be used by CP when generating the unique identifier part (last 6 hexadecimal digits) of a virtual adapter MAC address. If a MACIDRANGE SYSTEM range is not specified, CP creates unique identifiers in the entire range (000001 - FFFFFF).

USER *xxxxxx-xxxxxx* is the subset of the SYSTEM range of identifiers that is reserved for explicit user definition of MACIDs. The MACIDs within this range are reserved for the SET NIC MACID command, or for virtual adapters defined with the MACID operand on the NICDEF directory statement.

MACPROtect
turns MAC address protection on or off for virtual network devices on the system. Turning MAC address protection on prevents a guest from using a MAC address that is not assigned to the user's network device. MACPROTECT is valid only for ETHERNET virtual switches and guest LANs.

Three levels of inheritance are used when determining the MAC address protection level for a MAC address assigned to a network device. The highest is the system level, followed by the virtual switch or guest LAN level. The lowest level is the protection set for a specific network data device.

The MAC address protection level assigned is determined first at the device, then virtual switch or guest LAN and lastly the system level. If MAC address protection is set at the device level using the SET NIC CP command, then its current setting is used. When the device is set to UNSPECIFIED, it will inherit the MACPROTECT setting of the virtual switch. When the virtual switch MACPROTECT setting is set to UNSPECIFIED, the MAC address specified by this configuration statement will be used.

OFF

set the MAC address protection level to off for all virtual switches, guest LANs and network devices whose MAC address protection setting are not specified (set to UNSPECIFIED). Off is the default setting.

If MAC address protection is set to UNSPECIFIED by the SET NIC CP command for a specific device or the SET VSWITCH command for virtual switch coupled devices or the SET LAN command for LAN coupled devices, then the device's MAC address protection will be set off.

ON

set the MAC address protection level on for coupled devices defined with a MAC address protection set to UNSPECIFIED. Set the MAC address protection level to on for all virtual switches, guest LANs and network devices whose MAC address protection setting are not specified (set to UNSPECIFIED).

If MAC address protection is set to UNSPECIFIED by the SET NIC CP command for a specific device or the SET VSWITCH command for virtual switch coupled devices or the SET LAN command for LAN coupled devices, then the device's MAC address protection will be set on.

DNA DISABLE|ENABLE

Disable or Enable Directory Network Authorization

DISABLE

Turns off Directory Network Authorization management on the system. A NICDEF directory statement specified by the system administrator will not authorize the NIC defined to connect to the specified virtual network. Either a SET VSWITCH PORTNUM | GRANT CP command or a MODIFY VSWITCH System CONFIG statement must be issued on the target VSwitch to authorize the connection and to specify the NIC characteristics. Therefore, NICDEF PORTNUM, PORTTYPE, VLAN, PQUPLINKTX, PROMISCUOUS, and NOPROMISCUOUS operands will be ignored when VMLAN DNA DISABLE is specified.

ENABLE

Turns on Directory Network Authorization management on the system (default setting). This will allow a system administrator to create and authorize a virtual NIC created by a NICDEF directory statement to connect to a virtual network. All NIC attributes and VSwitch authorization can be performed via the system directory instead of issuing either SET VSWITCH PORTNUM | GRANT CP commands or a MODIFY VSWITCH System CONFIG statement. Authorization is only granted automatically when the target VSwitch is not managed by an ESM.

Usage Notes

1. Use the QUERY VMLAN command to find out the limits that have been established for a running CP system.
2. The limits established by the VMLAN statement only apply AFTER system initialization. The DEFINE LAN and DEFINE VSWITCH statements in SYSTEM CONFIG are not affected by these limits. Therefore, it is possible to find (using QUERY VMLAN) that the current number of guest LANs is higher than the limit.
3. To prohibit users from defining any guest LANs on a given system, set the limit to zero (0). For example, "VMLAN LIMIT TRANSIENT 0" would prevent general users from creating guest LANs.
4. Use the MACPREFIX and MACIDRANGE parameters, or the USERPREFIX parameter, if you require a predictable MAC address assignment for virtual NICs on your system. When you combine systems

using a virtual switch, each host should define a unique MACPREFIX in the SYSTEM CONFIG file to ensure that each host will generate unique MAC addresses.

5. When MACPREFIX and USERPREFIX are set to the identical value (explicitly or by default), and a virtual adapter is defined with the DEFINE NIC command or with the NICDEF or SPECIAL directory statement, a MAC address is assigned by concatenating the MACPREFIX value with the MACID value found first in the following priority list:
   a. The MACID from a SET NIC MACID command issued for the device
   b. The MACID specified on a NICDEF directory statement defining the device, which must be in the USER subset of the MACIDRANGE SYSTEM range
   c. An available MACID from the MACIDRANGE SYSTEM range
6. When MACPREFIX and USERPREFIX are set to different values, (as must be the case in an SSI cluster), the MACIDRANGE option is ignored and the entire MACID range (000001 - FFFFFF) is used for both automatic and user-defined locally administered MAC addresses on the system. When a virtual adapter is defined with the DEFINE NIC command or with the NICDEF or SPECIAL directory statement, a MAC address is assigned by concatenating the USERPREFIX value with the MACID value found first in the following priority list:
   a. The MACID from a SET NIC MACID command issued for the device
   b. The MACID specified on a NICDEF directory statement defining the device

   If a user-defined MACID cannot be found, the USERPREFIX is ignored. CP selects the next available MACID value and appends it to the MACPREFIX to create the MAC address.

Examples

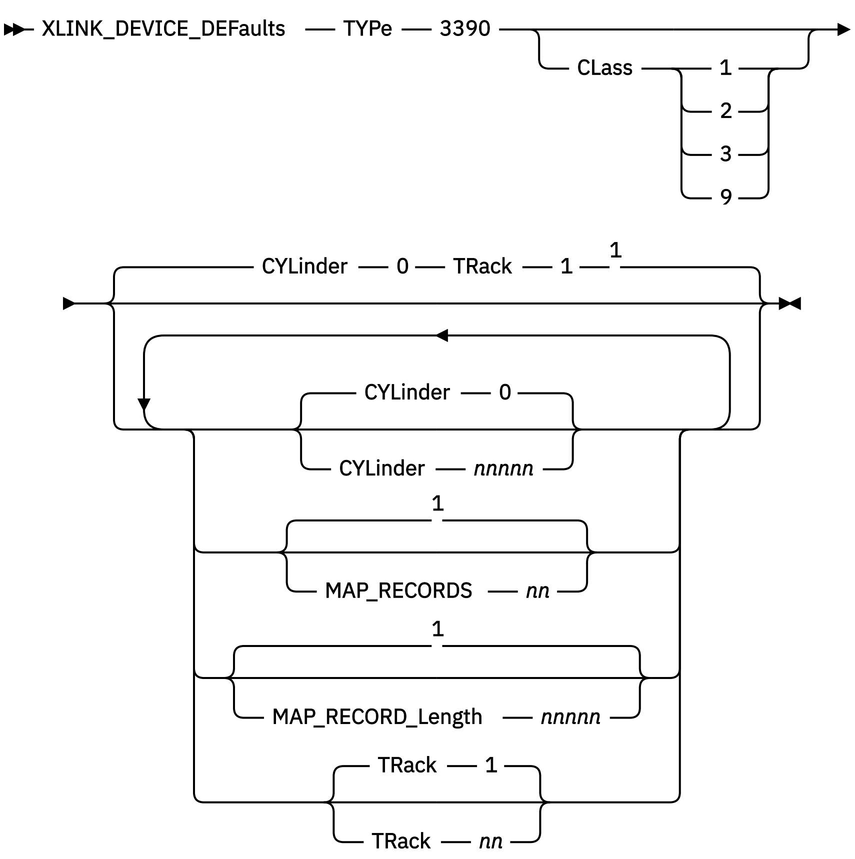
1. To allow any number of PERSISTENT guest LANs, but set a limit of 20 guest LANs for use on a transient basis, specify the following:

   ```
     vmlan limit persistent infinite transient 20
   ```

2. To enable accounting for every user-owned guest LAN, specify the following:

   ```
   vmlan accounting user on
   ```

XLINK_DEVICE_DEFAULTS Statement



Notes:

[1] The map record defaults depend on the DASD type.

Purpose

Use the XLINK_DEVICE_DEFAULTS statement to change the defaults for the cross-system link (XLINK) CSE area location and format for specific types of DASD.

Note: Although this statement allows you to change the definitions of cylinder, track, map records, and record length for the CSE area, it is not recommended. This statement allows you to provide link support for new devices with new geometries, should that become necessary in the future.

Attention: Current upper limits may exceed capacities of present devices and cause errors when attempting to access beyond a particular device limit. Therefore, any change of defaults should be done with caution.

You cannot specify the XLINK_DEVICE_DEFAULTS statement with the SSI statement.

How to Specify

Include as many statements as needed; they are optional. You can place XLINK_DEVICE_DEFAULTS statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

TYPe 3390

tells CP that you want to change the defaults for a 3390 DASD. Optionally, you can specify:

CLass 1

tells CP that this 3390 DASD belongs in the 1113 cylinder size group.

CLass 2

tells CP that this 3390 DASD belongs in the 2226 cylinder size group.

CLass 3

tells CP that this 3390 DASD belongs in the 3339 cylinder size group.

CLass 9

tells CP that this 3390 DASD belongs in the 10017 or greater, cylinder size group. This class supports the maximum number of cylinders on the 3390.

If you are not sure which class to specify, we recommend that you choose a cylinder size class larger than the actual number of cylinders on the device.

CYLinder *nnnnn*

tells CP the number of the cylinder where the CSE area is located for the specified DASD type. The variable *nnnnn* is a decimal number from 0 to one less than the maximum number of cylinders on the DASD. If omitted, the default is 0.

Note: Although this statement allows you to change the cylinder definition for the CSE area, it is not recommended. The statement allows you to provide link support for new devices with new geometries, should that become necessary in the future.

Attention: Current upper limits may exceed capacities of present devices and cause errors when attempting to access beyond a particular device limit. Therefore, any change of defaults should be done with caution.

MAP_RECORD *nn*

defines the number of map records on the CSE area. The variable *nn* is a decimal number from 2 to 56. Specify as many records as will fit on one track. Table 13 on page 325 shows the map record defaults for each device type.

Table 13. Map Record Defaults for Each DASD Type

Device Type	Number of Map Records	Map Record Length
3390-1 (native)	12	1113
3390-1 (emulated)	12	1113
3390-2 (native)	8	2226
3390-2 (emulated)	8	2226
3390-3 (native)	8	3339
3390-3 (emulated)	8	3339
3390-9 (native)	56	65520
OTHER	8	1024

Note: Although this statement allows you to change the definition of map records for the CSE area, it is not recommended. The statement allows you to provide link support for new devices with new geometries, should that become necessary in the future.

Attention: Current upper limits may exceed capacities of present devices and cause errors when attempting to access beyond a particular device limit. Therefore, any change of defaults should be done with caution.

MAP_RECORD_Length *nnnnn*

defines the number of cylinders CP should make available for XLINK to use. This value should be at least equal to the number of cylinders on the volume. If *nnnnn* is smaller than the number of cylinders on the volume, CP denies links to minidisks on cylinders with a number higher than *nnnnn*. The variable *nnnnn* is a decimal number from 203 to the maximum number of cylinders on the DASD. The maximum number that can be specified is 65520. Table 13 on page 325 shows the map record defaults for each device type.

For CKD devices *nnnnn* also defines the record length of the map records. On these devices 1 byte in the map record corresponds to 1 cylinder. All cylinders are mapped in 1 physical record per system. On ECKD capable CKD devices there may be several physical records per system. One byte in the map record corresponds to 2 cylinders. Each physical record is 256 bytes and maps 512 cylinders. The number of physical records needed depends on the number of cylinders made available to XLINK.

Note: Although this statement allows you to change the definitions of record length for the CSE area, it is not recommended. The statement allows you to provide link support for new devices with new geometries, should that become necessary in the future.

Attention: Current upper limits may exceed capacities of present devices and cause errors when attempting to access beyond a particular device limit. Therefore, any change of defaults should be done with caution.

TRack *nn*

tells CP on which track of the specified CKD DASD to place the CSE area. The variable *nn* is a decimal number from 0 to 64, but you cannot specify TRACK 0 if you also specified CYLINDER 0. If omitted, the default is TRACK 1.

CP uses multiple tracks for ECKD capable CKD devices. The number of tracks used for ECKD capable CKD devices depends on the number of cylinders on the volume. Thus, CP does not use the value specified in TRACK *nn* for an ECKD capable CKD device.

Note: Although this statement allows you to change the definition of track for the CSE area, it is not recommended. The statement allows you to provide link support for new devices with new geometries, should that become necessary in the future.

Attention: Current upper limits may exceed capacities of present devices and cause errors when attempting to access beyond a particular device limit. Therefore, any change of defaults should be done with caution.

Usage Notes

1. Support for the cross system extensions (CSE) environment has been removed. However, the cross-system link (XLINK) function that was included in CSE is still supported for non-SSI systems, and “CSE” is still used in some function and object names, command responses, and messages related to XLINK.
2. The CYLINDER, MAP_RECORD, MAP_RECORD_LENGTH, and TRACK operands let you specify CSE area values that are different from the default values associated with the device type. This is not recommended unless there is a valid reason for doing so.

Examples

1. The default placement of the XLINK map is on cylinder 0 track 1. If this default is inappropriate for your installation, you may specify a different DASD location on every XLINK_VOLUME_INCLUDE statement, or use the XLINK_DEVICE_DEFAULTS statement.

   For example, suppose that all of your 3390 devices, regardless of capacity, must reserve all tracks on cylinder 0 for other uses. Because of this, you decide to place the XLINK map on cylinder 5 track 0. Therefore, use the following XLINK_DEVICE_DEFAULTS statement:

   ```
   Xlink_Device_Defaults  Type 3390  Cylinder 5  Track 0
   ```

Any individual volumes on 3390 devices could have their XLINK map placed elsewhere on the device by use of the XLINK_VOLUME_INCLUDE statement.

XLINK_SYSTEM_EXCLUDE Statement

```
▶▶─ XLINK_SYSTEM_EXclude ── sysname ─▶◀
```

Purpose

Use the XLINK_SYSTEM_EXCLUDE statement to specify a system that CP is to exclude from the cross-system link (XLINK) function.

You cannot specify the XLINK_SYSTEM_EXCLUDE statement with the SSI statement.

How to Specify

Include as many statements as needed; they are optional. You can place XLINK_SYSTEM_EXCLUDE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

sysname
is the 1- to 8-character name of a system to be excluded from cross-system link.

Usage Notes

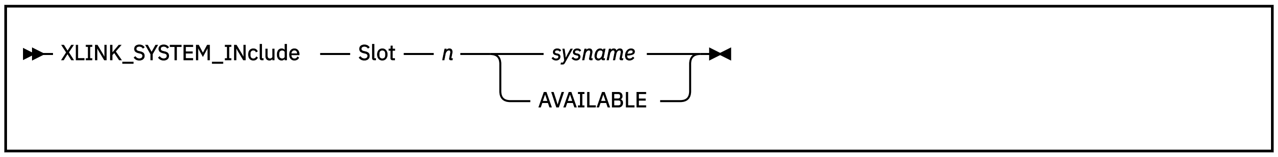
1. Support for the cross system extensions (CSE) environment has been removed. However, the cross-system link (XLINK) function that was included in CSE is still supported for non-SSI systems, and “CSE” is still used in some function and object names, command responses, and messages related to XLINK.
2. If a system name is specified on both an XLINK_SYSTEM_EXCLUDE statement and an XLINK_SYSTEM_INCLUDE statement, CP will ignore the XLINK_SYSTEM_EXCLUDE statement.
3. The XLINK function is not supported for FBA devices.

Examples

1. To exclude systems VM1, VM2, VM3, and VM4 from XLINK, use the following XLINK_SYSTEM_EXCLUDE statements:

```
Xlink_System_Exclude  vm1     /* Exclude the following systems from  */
Xlink_System_Exclude  vm2     /*  cross-system links so that no one  */
Xlink_System_Exclude  vm3     /*  on the other systems will have     */
Xlink_System_Exclude  vm4     /*  access to them.                    */
```

XLINK_SYSTEM_INCLUDE Statement



Purpose

Use the XLINK_SYSTEM_INCLUDE statement to specify a system that CP is to include in the cross-system link (XLINK) function.

You cannot specify the XLINK_SYSTEM_INCLUDE statement with the SSI statement.

How to Specify

Include as many statements as needed; they are optional. You can place XLINK_SYSTEM_INCLUDE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

Slot *n*
: tells CP what position this system will hold in the list of systems participating in cross-system link. Slot 1 is the first slot. The maximum value of *n* depends on the type of DASD under cross-system link protection.

sysname
AVAILABLE
: is the 1- to 8-character name of a system to be included in cross-system link.

AVAILABLE tells CP the slot is available for future use.

Usage Notes

1. Support for the cross system extensions (CSE) environment has been removed. However, the cross-system link (XLINK) function that was included in CSE is still supported for non-SSI systems, and “CSE” is still used in some function and object names, command responses, and messages related to XLINK.
2. If a system name is specified on both an XLINK_SYSTEM_EXCLUDE statement and an XLINK_SYSTEM_INCLUDE statement, CP ignores the XLINK_SYSTEM_EXCLUDE statement.
3. If you specify any systems on the XLINK_SYSTEM_INCLUDE or XLINK_SYSTEM_EXCLUDE statements and the system that you IPL is in neither list, CP prompts the operator with a warning that this omission may cause the loss of data integrity. The operator can choose to stop the system initialization process, continue as normal, or allow system initialization to continue but have CP skip the normal process of autologging users.
4. The order of system names in XLINK_SYSTEM_INCLUDE statements for all systems participating in cross-system link must be identical. That is, the system name for all XLINK_SYSTEM_INCLUDE SLOT 1 statements must be the same for all systems.
5. Marking a slot as AVAILABLE reserves it for future use. When an AVAILABLE slot is put to use, the slot should be updated with the system name on all participating systems in a timely manner.

Examples

1. To include systems BOSTON3, NEWYORK, and BOSTON4 in cross-system link in slots 1, 2, and 3 (respectively), use the following XLINK_SYSTEM_INCLUDE statements:

   ```
   /*-----------------------------------------------------*/
   /* Include the BOSTON3, NEWYORK, and BOSTON4 systems */
   /*    in cross-system linking.                       */
   /*-----------------------------------------------------*/
      Xlink_System_Include  Slot 1  boston3
      Xlink_System_Include  Slot 2  newyork
      Xlink_System_Include  Slot 3  boston4
   ```

XLINK_VOLUME_EXCLUDE Statement

```
►► XLINK_VOLUME_EXclude  — volid ►◄
```

Purpose

Use the XLINK_VOLUME_EXCLUDE statement to specify a DASD volume that CP is to exclude from the cross-system link (XLINK) function. The specified volume is not to be shared through cross-system link. If mounted and shared among systems, this volume has no extended link protection.

You cannot specify the XLINK_VOLUME_EXCLUDE statement with the SSI statement.

How to Specify

Include as many statements as needed; they are optional. You can place XLINK_VOLUME_EXCLUDE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

volid

is a volume serial number or a generic volume serial number for a specific subset of volumes to be excluded from cross-system link. The variable *volid* is a 1- to 6-character alphanumeric string with no imbedded blanks. A generic volume serial number is a 1- to 6-character string with asterisks (*) in place of one or more arbitrary characters and percent signs (%) in place of exactly one arbitrary character. For example:

```
Xlink_Volume_Exclude  es%vm*
```

excludes all volumes whose volume serial number start with ES and have VM as their fourth and fifth characters.

Usage Notes

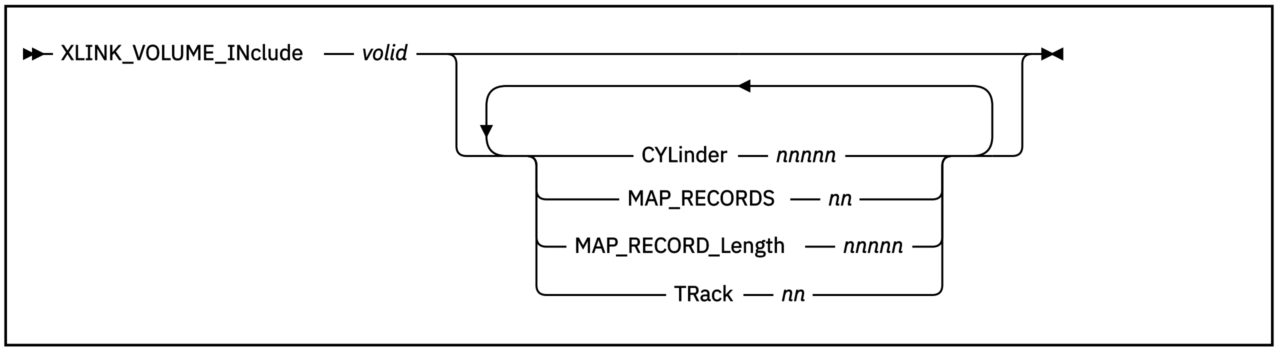
1. Support for the cross system extensions (CSE) environment has been removed. However, the cross-system link (XLINK) function that was included in CSE is still supported for non-SSI systems, and "CSE" is still used in some function and object names, command responses, and messages related to XLINK.
2. If you do not specify XLINK_VOLUME_EXCLUDE or XLINK_VOLUME_INCLUDE statements, and the system that you are IPLing is specified on an XLINK_SYSTEM_INCLUDE statement, all DASD volumes are included in cross-system link protection.
3. A volume is included in cross-system link protection if it is:
   - Explicitly listed on an XLINK_VOLUME_INCLUDE statement, or
   - Matched by a generic volume identifier on an XLINK_VOLUME_INCLUDE statement and not found (either explicitly or generically) on an XLINK_VOLUME_EXCLUDE statement.

Examples

1. To exclude volumes VMUSR4, VMUSR7, and VMUSR9 from cross-system linking, use the following XLINK_VOLUME_EXCLUDE statements:

```
Xlink_Volume_Exclude  vmusr4      /* Exclude VMUSR4, VMUSR7, and     */
Xlink_Volume_Exclude  vmusr7      /*    VMUSR9 from cross-system     */
Xlink_Volume_Exclude  vmusr9      /*    link operations ...          */
```

XLINK_VOLUME_INCLUDE Statement



Purpose

Use the XLINK_VOLUME_INCLUDE statement to specify a DASD volume that CP is to include in the cross-system link (XLINK) function.

Note: Although this statement allows you to change the definitions of cylinder, track, map records, and record length for the CSE area, it is not recommended. These operands allow you to provide link support for new devices with different geometries, should that become necessary in the future.

You cannot specify the XLINK_VOLUME_INCLUDE statement with the SSI statement.

How to Specify

Include as many statements as needed; they are optional. You can place XLINK_VOLUME_INCLUDE statements anywhere in the system configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

volid
is a volume serial number or a generic volume serial number for a specific subset of volumes to be included in cross-system link. The variable *volid* is a 1- to 6-character alphanumeric string with no imbedded blanks. A generic volume serial number is a 1- to 6-character string with asterisks (*) in place of one or more arbitrary characters and percent signs (%) in place of exactly one arbitrary character. For example:

```
Xlink_Volume_Include  es%vm*
```

includes all volumes whose volume serial number start with ES and have VM as their fourth and fifth characters.

CYLinder *nnnnn*
defines the cylinder where the CSE area is located for every *volid*. The variable *nnnnn* is a decimal number from 0 to one less than the maximum number of cylinders on the DASD. If omitted, the default is 0.

MAP_RECORDS *nn*
defines the number of map records on the CSE area. The variable *nn* should not be smaller than the number of systems that share the volume. If it is, then all links to minidisks on this volume are denied. We recommend that you use as many records as can fit on one track. See Table 14 on page 334 for the default values.

MAP_RECORD_Length *nnnnn*
defines the number of cylinders made available for XLINK use. This value should not be smaller than the number of cylinders on the volume. If it is, links to minidisks starting on a cylinder with a number higher than *nnnnn* are denied. The variable *nnnnn* is a decimal number from 203 to the maximum

number of cylinders on the DASD. The maximum number that can be specified is 65520. See Table 14 on page 334 for the default values.

For CKD devices, *nnnnn* also defines the record length of the map records. On these devices, one byte in the map record corresponds to one cylinder. All cylinders are mapped in one physical record per system. On ECKD-capable CKD devices, there may be several physical records per system, as one byte in the map record corresponds to two cylinders. Each physical record is 256 bytes and maps 512 cylinders. The number of physical records needed depends on the number of cylinders made available to XLINK.

TRack *nn*

defines the track number on which the CSE area is located for every CKD DASD volume matching *volidn*. The variable *nn* is a decimal number from 0 to 64, but it cannot be 0 if the value of CYLINDER is 0. If omitted, the default is 1.

For ECKD-capable CKD devices, multiple tracks are used. The number of tracks used for ECKD-capable CKD devices depends on the number of cylinders on the volume. The value specified in TRACK *nn* is not used for an ECKD-capable CKD device matching *volidn*.

Usage Notes

1. Support for the cross system extensions (CSE) environment has been removed. However, the cross-system link (XLINK) function that was included in CSE is still supported for non-SSI systems, and "CSE" is still used in some function and object names, command responses, and messages related to XLINK.
2. If you do not specify XLINK_VOLUME_INCLUDE or XLINK_VOLUME_EXCLUDE statements, and the system that you are IPLing is specified on an XLINK_SYSTEM_INCLUDE statement, all DASD volumes are included in cross-system link protection.
3. A volume is included in cross-system link protection if it is:
   - Explicitly listed on an XLINK_VOLUME_INCLUDE statement, or
   - Matched by a generic volume identifier on an XLINK_VOLUME_INCLUDE statement and not found (either explicitly or generically) on an XLINK_VOLUME_EXCLUDE statement.
4. The values you choose for CYLINDER and TRACK must be valid for the device type or types on which the specified volumes reside. This statement does not make sure that your values are valid, because when it is processed, the device types are not yet known.
5. Although the XLINK_VOLUME_INCLUDE enables you to change the definitions of cylinder, track, record length, and map records for the CSE area, we do not recommend that you do so. This capability is included to provide link support for new devices with new geometries, should that support become necessary in the future. Table 14 on page 334 shows the map record defaults for each device type.

Table 14. Map Record Defaults for Each Device Type

Device Type	Number of Map Records	Map Record Length
3390-1 (native)	12	1113
3390-1 (emulated)	12	1113
3390-2 (native)	8	2226
3390-2 (emulated)	8	2226
3390-3 (native)	8	3339
3390-3 (emulated)	8	3339
3390-9 (native)	56	65520
OTHER	8	1024

6. These defaults support the current CKD and ECKD-capable CKD DASD. The length of each map record is the maximum number of cylinders on any currently existing model of that device type. The number of records defined for other devices is 8, but the actual number of records used is limited to the number that fits on a track of such "*other*" device.

Examples

1. To include DASD volumes USRPK1, USRPK2, and USRPK3 in cross-system linking, use the following XLINK_VOLUME_INCLUDE statements:

```
   Xlink_Volume_Include  usrpk1    /* Include USERPK1, USERPK2 and      */
   Xlink_Volume_Include  usrpk2    /*    USERPK3 in cross-system link   */
   Xlink_Volume_Include  usrpk3    /*    operations ...                 */
```

Chapter 7. The Logo Configuration File

This chapter:

- Provides a summary of logo configuration file statements
- Provides general rules for coding the logo configuration file
- Describes each logo configuration file statement in detail

Using a Logo Configuration File

Creating a logo configuration file allows you to change features of logos that appear on the screen or on separator pages of printed output. The logo configuration file overrides the default logo information, which is defined in the HCPBOX ASSEMBLE file included in the CP module. For example, you can:

- Have certain logo picture files appear on certain terminals
- Include your own information in status and online fields
- Place often-used command text in the command input field on the logo screen

The name of the logo configuration file defaults to LOGO CONFIG unless you specify a different name on the LOGO_CONFIG statement in the system configuration file. You must place the logo configuration file and the files to which it refers on the parm disk, the CMS minidisk that CP will access at IPL time.

Summary of Logo Configuration File Statements

Table 15 on page 337 lists all the logo configuration file statements, gives you a brief description of each statement, and points you to where you can get more information about each statement.

Table 15. Logo Configuration File Statements (LOGO CONFIG)

Statement	Description	Location
CHOOSE_LOGO	Defines the logo picture files that CP uses for specified terminals and for separator pages of printed output.	"CHOOSE_LOGO Statement" on page 342
INPUT_AREA	Defines which file CP uses when displaying the input area of the logo.	"INPUT_AREA Statement" on page 348
ONLINE_MESSAGE	Specifies the file whose text CP should use for the online message area of the logo.	"ONLINE_MESSAGE Statement" on page 349
STATUS	Defines the text of the status fields displayed in the bottom right corner of the screen.	"STATUS Statement" on page 350

General Rules for Coding a Logo Configuration File

When creating or updating a logo configuration file, you must follow some general syntax rules.

Format

The logo configuration file can be a fixed or variable length record file. No individual record in the file should be longer than 4000 characters. No individual statement in the file should be longer than 4000 characters after all continuations of the statement have been resolved.

Comments

You can add comments to the file by delimiting them with /* and */. A single comment may span multiple records in the file. Thus,

```
/*-------------------------------------------------*
 * The following section defines the status areas  *
 *-------------------------------------------------*/
```

is valid. As many comments as necessary may be entered on a single record in the file. For example,

```
   Choose_logo default /* fn ft */ LOGO DEFAULT /* Default cmd */ 'DIAL PVM'
```

is allowed.

Continuations

To put a statement in more than one record of the configuration file, place a comma at the ends of all but the last line of the statement. For example, you can code the CHOOSE_LOGO statement as follows:

```
 Choose_logo Local         ,
    Address 100-200        ,
    Screen_size 27x132     ,
    MODEL5 LOGO            ,
    'DIAL VTAM'
```

A comma at the end of a line indicates that the statement is not yet complete. You should not code a comma if the statement information is complete but a comment continues to the next record.

```
 Choose_logo Local       ,
    Address 100-200      ,
    Screen_size 27x132   ,
    MODEL5 LOGO          ,
    'DIAL VTAM'              /* Go to VTAM if user enters
                                nothing for user or password  */
```

is valid, but a comma after 'DIAL VTAM' would make the statement invalid, as you have specified no subsequent options. The comma can be placed directly after the last entry on the line. Thus, the example above would also be valid if specified as

```
 Choose_logo Local,
    Address 100-200,
    Screen_size 27x132,
    MODEL5 LOGO,
    'DIAL VTAM'              /* Go to VTAM if user enters
                                nothing for user or password  */
```

Finally, a blank line does not cause a continuation to be terminated. You could specify

```
 Choose_logo Local,
    Address 100-200,

    Screen_size 27x132,
    MODEL5 LOGO,
    'DIAL VTAM'              /* Go to VTAM if user enters
                                nothing for user or password  */
```

because CP ignores any blank lines in the configuration file.

Case

In general, it does not matter whether information in the logo configuration file is entered in upper case or mixed case. Most entries in the configuration file are converted to upper case before they are processed. Thus,

```
   Status Running Running
```

and

```
Status Running RUNNING
```

would have the same results. An exception to this rule occurs when CP encounters a quoted string in the file. A quoted string is any string enclosed within single quotation marks; the string may contain blanks and special character sequences such as /* and */ that are not usually permitted in a single token. Where quoted strings are allowed and a quoted string is specified in the configuration file, the text inside the quotation marks is not converted to upper case. In the following example, the specified status area would remain in mixed case:

```
Status Running 'Running'
```

Categories

Each of the four statements in a logo configuration file modifies a different part of the screen.

- CHOOSE_LOGO defines the logo picture that occupies the center of a screen.
- INPUT_AREA defines the input area that contains the user ID, password, and command areas.
- ONLINE_MESSAGE defines the online message that appears at the top of the screen.
- STATUS defines the status area in the lower right hand portion of the screen.

Precedence

You can include more than one INPUT_AREA or ONLINE_MESSAGE statement in the configuration file, but CP will use only the last occurrence of a given statement during logo processing. For example,

```
   Input_Area  INPUT FILE
```

followed by

```
   Input_Area  INPUT AREA
```

would cause CP to use the file INPUT AREA instead of INPUT FILE.

You can code a STATUS statement for each of the different operands. You can code multiple STATUS statements with the same operand, but CP will use only the last one with a given operand. For example,

```
   Status Running 'Going   '
```

followed by

```
   Status Running 'Gone    '
```

would cause CP to use Gone instead of `Going`.

You should code CHOOSE_LOGO statements in three separate groups, one for each of the following categories:

- Locally attached terminals
- Terminals logging on through logical devices
- Terminals managed by a VSM.

Within these groups, you should code statements in the order of least specific to most specific. Statements that choose logo picture files for certain terminals within a group should occur after statements that choose files for all the terminals in that group.

CP puts the information specified on each group of CHOOSE_LOGO statements into separate search chains, which it searches from the bottom up. When CP encounters a terminal, it goes through the search chain established for similar terminals until it finds a statement that applies to the specific terminal. Thus, for a given terminal, CP uses the last statement in the logo configuration file that applies.

Selection

The type of terminal determines the process that CP uses to select a logo picture file. CP supports three types of terminals:

- Terminals directly managed by CP
- Terminals logging on through logical devices
- Terminals managed by a VTAM service machine (VSM)

For each type, CP looks at a series of CHOOSE_LOGO statements to find out which logo picture file to use for a terminal.

- For terminals that CP manages directly, CP searches through CHOOSE_LOGO statements with the LOCAL operand.
- For terminals represented by logical devices, it searches through CHOOSE_LOGO statements with the LDEV operand.
- For terminals managed by a VSM, it searches through CHOOSE_LOGO statements with the VSM operand.

In all three cases, if CP cannot find a statement that applies to a given terminal, it will use the file specified on the CHOOSE_LOGO DEFAULT statement, if there is one.

If CP finds a statement that applies to a terminal, but the logo picture file it specifies is too large for the screen, CP will skip the statement and try the next one in the search order.

When CP gets to the end of the search order, the last file it will try to use is either the one specified on a CHOOSE_LOGO DEFAULT statement, or, if there is no such statement, the default logo in HCPBOX ASSEMBLE. If the default logo picture file (from either source) is too large, CP will use the file specified on the CHOOSE_LOGO MINIMUM statement, if there is such a statement.

If all else fails, CP will use the logos defined in HCPBOX.

Creating Logo Screens

You can use the DRAWLOGO sample utility program to create logo screens for your system. DRAWLOGO invokes the X$DRWL$X sample XEDIT macro. With this utility, you can edit the text of the logo file and modify the 3270 screen attributes of the logo components in a logo file:

- The online message
- The picture
- The input area.

DRAWLOGO creates these logo files on the first CMS minidisk accessed in R/W mode. By default, the file type is LOGO. For example, assuming file mode A is accessed in read/write mode, to create LOCOTERM LOGO A, issue:

```
drawlogo locoterm
```

In addition to using XEDIT commands, you can use special functions that are defined for the PF keys:

- Insert 3270 attributes (for colors and highlighting, for example). When you insert an attribute into your logo, a special symbol appears on your screen to mark the position of the attribute. This symbol does not appear when the logo is put into production and displayed by CP.
- Change attributes.
- Insert input field markers.
- Display your logo as you are creating it.

DRAWLOGO creates a new logo configuration file (LOGO CONFIG), but it does not put it into production. To make your new logo screen active, target the LOGO CONFIG file after updating it to include your new logo screen.

For a complete description of DRAWLOGO, see "DRAWLOGO" on page 838.

Special Considerations When Creating Logos

You can use certain special characters to add color anywhere in the ONLINE_MESSAGE and CHOOSE_LOGO files. If you do so, be careful when you use the INPUT_AREA file, as the screen cursor will be tabbed to the strings defining the beginning of the color fields as well as to the various INPUT_AREA fields.

Other special characters are defined for use in building an INPUT_AREA file. The input area consists of the user ID, password, and command areas.

The use of four colors is supported. PF (program function) keys are defined to allow the insertion of the correct hexadecimal values. The colors defined are as follows:

Table 16. Colors for the CHOOSE_LOGO and ONLINE_MESSAGE Files

Color	**Hexadecimal Representation**
Blue	1DF0
Green	1D40
Red	1DC8
White, highlighting	1DF8
Protected field	1D70

The input area comprises three input fields, user ID, password, and command areas. Customarily they appear in this order; the user ID area must precede the password area. The fields may all appear on the same line or on different lines.

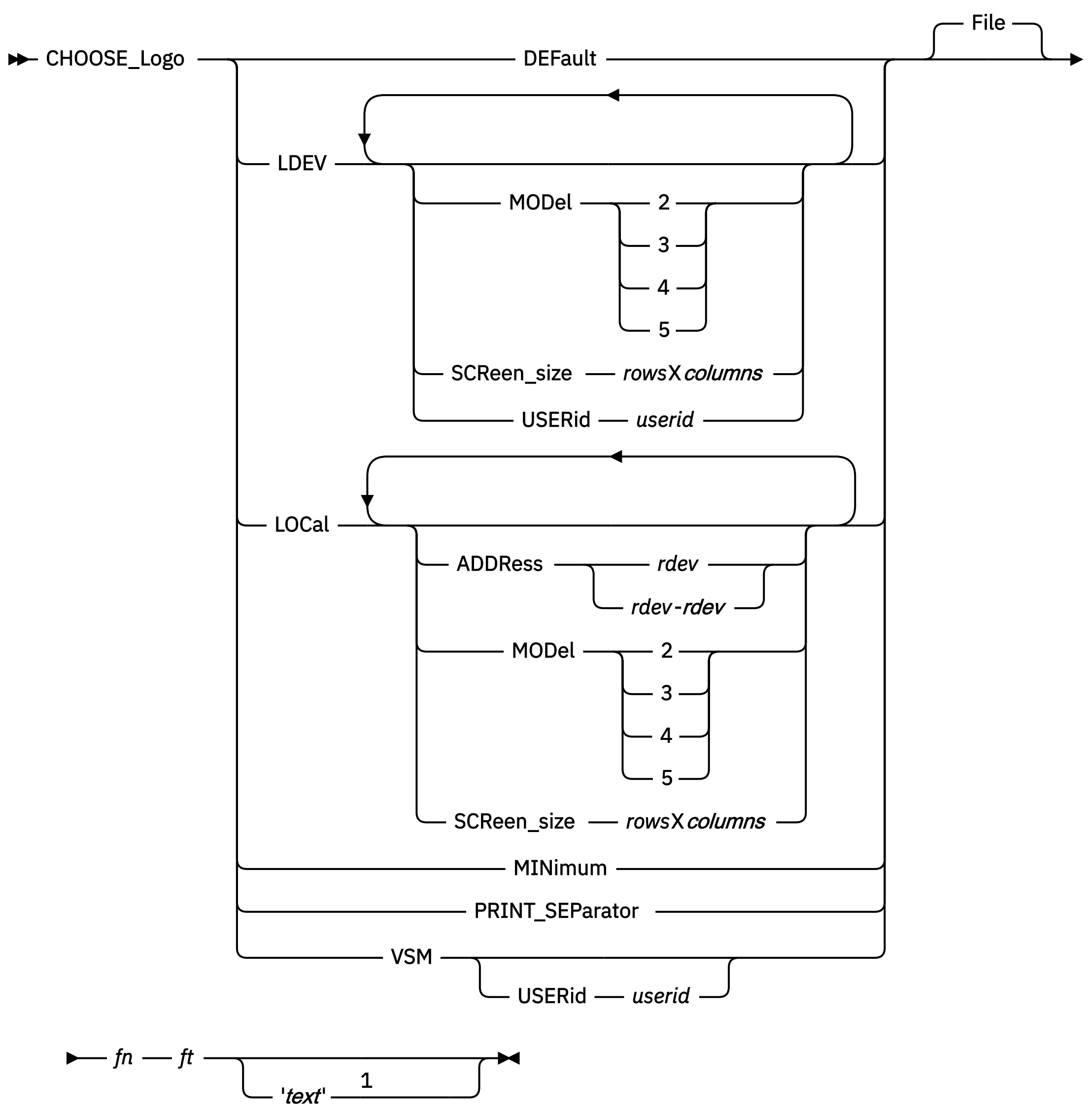
PF (program function) keys are also defined to assist in their use. A table of the different fields and their values follows.

Table 17. Input Area Fields

Field ID	**Hexadecimal Representation**
User ID	1D4013
Password	1D4C
Command	1D40

The above values are the same as in HCPBOX. Logo processing scans the input area for these values in order to determine where to read from the screen. For the logon process to operate correctly, these values must be used.

CHOOSE_LOGO Statement



Notes:

[1] If you specified MINIMUM, PRINT_SEPARATOR, or VSM, you cannot specify *text*.

Purpose

Use the CHOOSE_LOGO statement to tell CP which logo picture files to use for terminals and printer separator pages. You can have separate logo picture files for:

- All terminals logging on to a system
- Terminals directly attached to a particular system
- Locally-attached terminals at certain device addresses
- Terminals logging on through a logical device
- Terminals too small to accommodate files selected for them by another CHOOSE_LOGO statement
- Separator pages of printed output from spooling printers
- Terminals logging on through a particular VTAM service machine (VSM) user ID.

How to Specify

Include as many statements as needed; they are optional. You can place CHOOSE_LOGO statements anywhere in the LOGO CONFIG file. Be aware that when looking for the proper logo picture file to display, CP starts at the bottom of your LOGO CONFIG file and works its way up until it finds the first CHOOSE_LOGO statement that applies. Therefore, when specifying multiple CHOOSE_LOGO statements, put the generic ones at the top of the file and put the specific ones at the bottom.

Operands

DEFault
tells CP which logo picture file to use when no other CHOOSE_LOGO statements apply to a specific terminal.

LDEV
tells CP which logo picture file to use for terminals logging on to the system through a logical device. You can choose to have one logo picture file for all terminals or you can choose a specific model number, screen size, or user ID:

MODel
tells CP the model number of the terminal. The table below shows the screen size associated with each model:

Model	Equivalent Screen Size
2	24 by 80
3	32 by 80
4	43 by 80
5	27 by 132

Note: MODEL operands only apply to terminals with at least as many rows and columns as the screen size that the model number implies. For example, if you had the following statements in your LOGO CONFIG file:

```
Choose_Logo  Default         File  DEFAULT  LOGO
Choose_Logo  Ldev  Model  3  File  PVM      LOGO
```

and a user with a 24 by 80 terminal used the PVM logical device to get to your system, that user would be looking at the DEFAULT LOGO, not the PVM LOGO. The user needs at least a 32 by 80 terminal to see the PVM LOGO.

SCReen_size *rows*X*columns*
tells CP the screen size of the terminal. SCREEN_SIZE operands only apply to terminals with at least as many rows and columns as the user's actual screen size.

USERid *userid*
tells CP to use a specific logo picture file for a specific user ID that creates the logical device (for example, PVM or TCP/IP). You can use this operand to tell CP to use a certain logo picture file for terminals logging on through certain types of logical devices.

LOCal
tells CP which logo picture file to use for devices attached directly to the system. You can choose to have one logo picture file for all locally-attached devices, or you can choose a specific address, a specific address range, the model number, or the screen size:

ADDRess *rdev*
ADDRess *rdev-rdev*
tells CP which logo picture file to use for a specific device address (or range of device addresses) for terminals that CP manages directly. *rdev* must be a hexadecimal number between X'0000' and X'FFFF'.

MODel
tells CP the model number of the terminal. The table below shows the screen size associated with each model:

Model	Equivalent Screen Size
2	24 by 80
3	32 by 80
4	43 by 80
5	27 by 132

Note: MODEL operands only apply to terminals with at least as many rows and columns as the screen size that the model number implies. For example, if you had the following statements in your LOGO CONFIG file:

```
Choose_Logo  Default           File  DEFAULT  LOGO
Choose_Logo  Local  Model  4   File  LOCAL    LOGO
```

and someone used a locally-attached terminal with 32 rows and 80 columns, that user would be looking at the DEFAULT LOGO, not the LOCAL LOGO. The user needs at least a 43 by 80 terminal to see the LOCAL LOGO.

SCReen_size *rows*X*columns*
tells CP the screen size of the terminal. SCREEN_SIZE operands only apply to terminals with at least as many rows and columns as the user's actual screen size.

MINimum
tells CP which logo picture file to use when the file specified on another CHOOSE_LOGO statement is too large to fit on that terminal's screen.

PRINT_SEParator
tells CP which logo picture file to use on separator pages of output printed on spooling printers.

VSM
tells CP which logo picture file to use for people logging on through VTAM service machines (VSMs). You can choose to have one logo picture file for all VSMs, or you can choose a specific user ID:

USERid *userid*
tells CP the user ID of the VSM through which the user is logging on. You can use this operand to tell CP to use a specific logo picture file for terminals logging on through a specific VSM.

fn ft
File *fn ft*
tells CP the file name and file type of the logo picture file. The FILE keyword is optional, unless you gave the logo picture file a name that CP might mistake for a different operand. For example, use FILE if the logo picture file's name is MOD LOGO, because CP interprets the file name MOD as an abbreviation of the MODEL keyword.

'*text*'
tells CP what text to display in the command area of the logo's area. Enclose the text in single or double quotation marks, so that CP does not mistake an imbedded blank as the end of the text. If the text is longer than the input area, CP truncates the text.

If you need to include a single quotation mark in the title, enclose the whole string with double quotation marks. If you need to include a double quotation mark in the title, enclose the whole string with single quotation marks. Also, you may use two consecutive double quotation marks ("") to represent a " character within a string delimited by double quotation marks. Similarly, you may use two consecutive single quotation marks ('') to represent a ' character within a string delimited by single quotation marks.

Usage Notes

Notes for Logo Picture Selection on Terminals:

For terminal picture selection, CP determines the amount of space that remains on the screen after accommodating the online message (at the top of the screen), the input area (at the bottom of the screen) and leaving one reserved line. For example, if the screen on which a logo is to be displayed measures 32 rows by 80 columns and the input area contains 6 records, the picture must fit in the remaining space:

```
   remaining rows =    32  (rows in screen)
                    -  6  (rows in input area)
                    -  1  (rows in online message)
                    -  1  (reserved row)
                 =    24
```

Thus, the picture file selected for this screen can measure no more that 24 rows by 80 columns.

A Note About Picture Widths

For VSM logos, the screen is assumed to be 24 rows by 78 columns and the input area need not be accommodated within the 24 rows. Thus, a VSM logo picture file can never contain more than 22 rows or be wider than 78 characters.

No matter how large a screen is on which CP is to display a logo, the number of rows in the logo picture file multiplied by the logical record length of the picture file cannot exceed 3000 characters.

The type of terminal on which the logo is to be displayed determines the process that CP uses to select a logo picture file. For each type of terminal, CP looks at a series of CHOOSE_LOGO statements to find out which logo picture file to use for the terminal:

- For terminals that CP manages directly, CP searches through CHOOSE_LOGO statements with the LOCAL operand.
- For terminals represented by logical devices, it searches through CHOOSE_LOGO statements with the LDEV operand.
- For terminals managed by a VSM, it searches through CHOOSE_LOGO statements with the VSM operand.

In all three cases, CP attempts to find a CHOOSE_LOGO statement that applies to the screen and for which the picture file exists on a CP-accessed disk and fits in the space remaining on the screen. The CHOOSE_LOGO statements are scanned in reverse order of their specification in the logo configuration file.

If no CHOOSE_LOGO statement meets the aforementioned criteria, the next step taken by CP will depend on whether or not a CHOOSE_LOGO DEFAULT statement was specified in the logo configuration file.

- If CHOOSE_LOGO DEFAULT was specified, CP will attempt to read the file specified on the statement and will check if the picture in the file fits in the space remaining on the screen. The maximum width (logical record length) of a default logo file is 78 characters. If the file does not exist on a CP-accessed minidisk or the file is too large to fit on the screen, CP will move on to minimum logo processing.
- If CHOOSE_LOGO DEFAULT was not specified, CP will check to see if the logo picture specified at HCPBOXNS in HCPBOX fits on the space remaining on the screen. If the picture at HCPBOXNS is too large to fit on the screen, CP will move on to minimum logo processing.

As the minimum logo screen does not include an input area, CP determines the amount of rows remaining for a logo picture as:

```
(number of rows in screen) - 1 (for online message) - 2 (reserved lines)
```

Thus, our example above would now read:

```
   remaining rows =    32  (rows in screen)
                    -  1  (rows in online message)
```

```
                     -  2  (reserved rows)
                   =    29
```

Thus, the minimum logo picture used for this screen can measure no more than 29 rows by 78 columns. Minimum logo processing proceeds as follows:

- If CHOOSE_LOGO MINIMUM was specified, CP will attempt to read the file specified on the statement and will check if the picture in the file fits in the space remaining on the screen. The maximum width (logical record length) of a minimum logo file is 78 characters. If the file does not exist on a CP-accessed minidisk or the file is too large to fit on the screen, CP will move on to basic logo processing.
- If CHOOSE_LOGO MINIMUM was not specified, CP will check to see if the logo picture specified at HCPBOXMS in HCPBOX fits on the space remaining on the screen. If the picture at HCPBOXMS is too large to fit on the screen, CP will move on to basic logo processing.

Finally, if no suitable logo picture has been found for the screen, CP will use the basic logo picture defined at HCPBOXBS in HCPBOX. This logo contains no picture; only an online message will be displayed at the top of the screen and a status area at the bottom of the screen. This logo is guaranteed to fit on all screens.

Notes for Separator Pages of Printed Output:

1. Files containing logo pictures for separator pages should have no more than 16 records, each of which should not be longer than 49 characters.
2. If you do not code a CHOOSE_LOGO PRINT_SEPARATOR statement, or if the file specified is too large to fit on a separator page, then CP will use the spooling logo defined at HCPBOXBX or, failing that, HCPBLKDB. If you change the logo picture file for the separator pages, enter the CPACCESS command to have CP re-access the minidisk where the separator logo is stored. Use the REFRESH command to activate the changes.

Examples

Defaults:

```
   Choose_Logo Default  CAMBVM3 DEFAULT
```

specifies that CP should use the file CAMBVM3 DEFAULT to create default logos.

```
   Choose_Logo Minimum  CAMBVM3 MINIMUM
```

specifies that CP should use the file CAMBVM3 MINIMUM as the logo picture file when no other picture file can be found to fit on the terminal screen.

Logical Devices:

```
   Choose_Logo LDEV LDEVS LOGO
```

specifies that CP should use the logo picture file LDEVS LOGO for all terminals logging on through logical devices.

```
   Choose_Logo LDEV Model 3 CAMBVM3 MOD3LOGO
```

specifies that CP should use the logo picture file CAMBVM3 MOD3LOGO for all the Model 3 terminals that log on through logical devices. Model 3 terminals have 32 rows and 80 columns.

```
   Choose_Logo LDEV Model 3 Userid PVM  PVM LOGO
```

specifies that CP should use the logo picture file PVM LOGO for all terminals that have at least 32 rows and 80 columns and that log in through PVM.

Locally Attached Terminals:

```
   Choose_Logo Local CAMBVM3 LOCAL
```

specifies that CP should display the file CAMBVM3 LOCAL on all locally attached terminals.

```
Choose_Logo Local Address 0010-001F CAMBVM3 LOCAL2 'DIAL VTAM'
```

specifies that for terminals attached locally between the addresses X'0010' and X'001F', CP should use the logo picture file CAMBVM3 LOCAL2, and place the command DIAL VTAM on the command line.

```
Choose_Logo Local Address 0200-02FF Screen_size 27X132 CAMBVM3 LOCMOD5
```

specifies that CP should display the file CAMBVM3 LOCMOD5 on all the Model 5 terminals (terminals that have 27 rows and 132 columns) between addresses X'0200' and X'02FF'.

Separator Pages:

```
Choose_Logo Print_separator PRINTSEP LOGO
```

specifies that CP should use the logo picture file PRINTSEP LOGO on separator pages of printed output.

Terminals Logging on through a VSM:

```
Choose_Logo VSM   Userid VTAM   CAMBVM3 VTAM
```

specifies that CP should use the logo picture file CAMBVM3 VTAM on all screens managed by the VSM named VTAM.

INPUT_AREA Statement

```
►► INPUT_AREA — fn — ft ►◄
```

Purpose

Use the INPUT_AREA statement to define a file whose contents will be used in place of the input area provided in HCPBOX ASSEMBLE. This input area contains the input fields for a user ID, password, and command text.

How to Specify

You can place INPUT_AREA statements anywhere in the logo configuration file. If you specify more than one INPUT_AREA statement, the last overrides any previous specifications.

Operands

fn
tells CP the file name of the file containing the input area text.

ft
tells CP the file type of the file containing the input area text.

Usage Notes

1. If the input area file contains more than 6 records or is wider than 80 characters, CP will use the input area in HCPBOX instead of the one specified in the file.
2. If more than one version of the file exists on the disks accessed by CP, CP will use the file on the lowest accessed minidisk.
3. Although you can use different logo pictures for different terminals, you can define only one input area, which will be displayed on all terminals.
4. CP is shipped with a sample utility program, DRAWLOGO, and a sample XEDIT macro, X$DRWL$X, to help you construct the input area file. Refer to Appendix A, "Sample Utility Programs," on page 837 for more information on DRAWLOGO and X$DRWL$X.
5. Within the file, you can choose to place more than one input field on one line.

Examples

1. To use the VM1 INPUT file to define the input area, use the following INPUT_AREA statement:

```
Input_Area VM1 INPUT                  /* Use the Input Area defined in   */
                                      /*       file, "VM1 INPUT"         */
```

ONLINE_MESSAGE Statement

```
▸▸─ ONLINE_MESSage ── fn ── ft ─▸◂
```

Purpose

Use the ONLINE_MESSAGE statement to change what appears in the online message section, which always appears at the top of the logo screen. If you omit this statement, CP uses the default online message defined in HCPBOX ASSEMBLE.

How to Specify

You can place ONLINE_MESSAGE statements anywhere in the logo configuration file. If you specify more than one ONLINE_MESSAGE statement, the last overrides any previous specifications.

Operands

fn
tells CP the file name of the file containing the online message text.

ft
tells CP the file type of the file containing the online message text.

Usage Notes

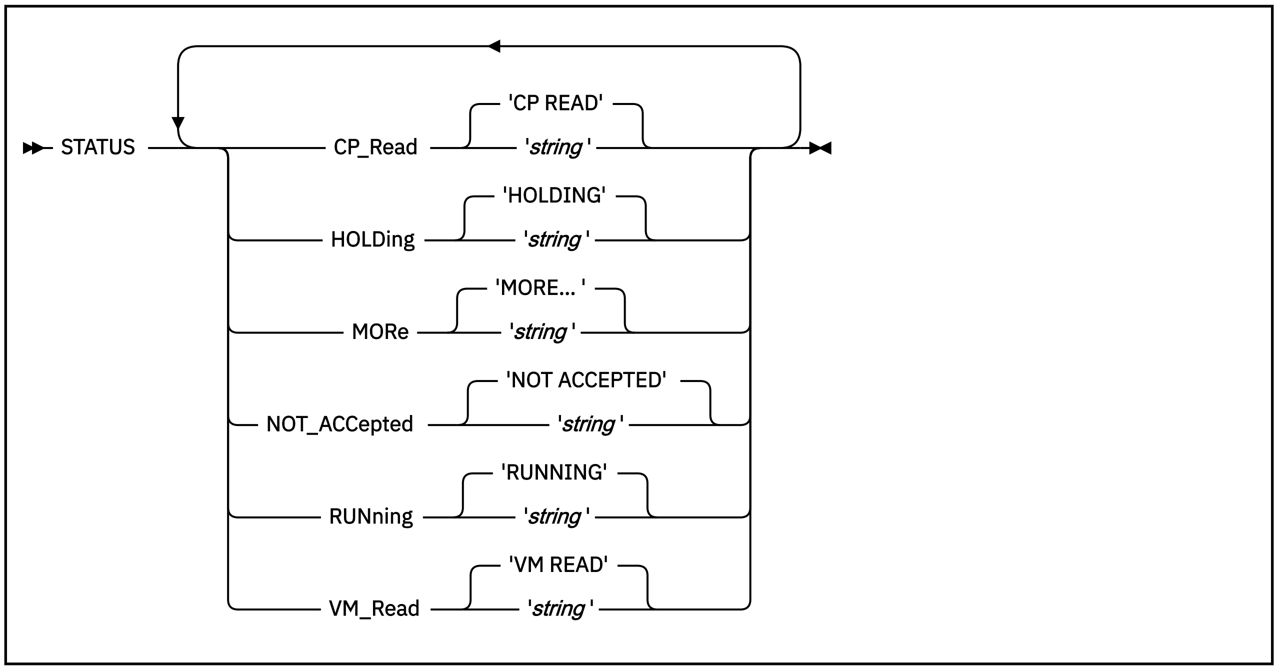
1. If the online message file contains more than one record or is wider than 78 characters, CP will use the online message defined in HCPBOX instead of the one defined in the file.
2. If the online message text is wider than the screen on which it will be displayed, CP truncates it.

Examples

1. To use the TOP MESSAGE file to define the online message section, use the following ONLINE_MESSAGE statement:

```
Online_Message  Top Message             /* The Online Message Can be found */
                                        /*    in file "TOP MESSAGE"        */
```

STATUS Statement



Purpose

Use the STATUS statement to redefine the default text of the status fields displayed in the bottom right corner of the screen.

How to Specify

Include as many statements as needed; they are optional. You can place STATUS statements anywhere in the logo configuration file. If you specify more than one statement with the same operands, the last operand definition overrides any previous specifications.

Operands

CP_Read *string*
: identifies the string to be displayed in the status area when a virtual machine has issued a CP read. The string can be as long as eight characters. The default string is 'CP READ'.

HOLDing *string*
: identifies the string to be displayed in the status area when there is more information to be displayed and the current screen cannot be cleared because highlighted output appears on it, or when the user presses ENTER when the screen is in MORE... status. The string can be as long as eight characters. The default string is 'HOLDING'.

MORe *string*
: identifies the string to be displayed in the status area when there is more information to be displayed and there is no highlighted text on the current screen. The string can be as long as eight characters. The default string is 'MORE... '.

NOT_ACCepted *string*
: identifies the string to be displayed in the status area when CP is executing a command that prevents it from accepting any more input from the command line. The string can be as long as 12 characters. The default string is 'NOT ACCEPTED'.

RUNning *string*
: identifies the string to be displayed in the status area when none of the other status conditions exists. The string can be as long as eight characters. The default string is 'RUNNING'.

VM_Read *string*
identifies the string to be displayed in the status area when a virtual machine has issued a VM read. The string can be as long as eight characters. The default string is 'VM READ'.

Usage Notes

1. You may have applications that depend on the default status fields. Changing the values of the fields may cause these applications to fail.
2. If you want to include imbedded blanks or preserve mixed case in your character strings, enclose the strings in single quotation marks.
3. The string may be enclosed in single or double quotation marks unless it is a single word entered with no imbedded blanks, in which case the entire word will be capitalized.

   If you need to include a single quotation mark in the string, enclose the whole string with double quotation marks. If you need to include a double quotation mark in the string, enclose the whole string with single quotation marks. Also, you may use two consecutive double quotation marks ("") to represent a " character within a string delimited by double quotation marks. Similarly, you may use two consecutive single quotation marks ('') to represent a ' character within a string delimited by single quotation marks.

Examples

1. You can code the following STATUS statement to change all of the status fields to mixed case:

```
    Status CP_Read 'CP Read',
           Holding 'Holding',
           More 'More… ',
           Not_Accepted 'Not accepted',
           Running 'Running',
           VM_Read 'VM Read'
```

Chapter 8. Setting Up Service Virtual Machines

This chapter describes how to set up virtual machines for:

- Accounting, error recording, and symptom record recording
- Communication controller support
- Service pool virtual machines
- Data storage management.

Notes:

1. All virtual machines that contain data retrieval applications and use the IUCV facility to connect to the CP accounting, error recording, or symptom record recording system services must include an IUCV directory statement that identifies the appropriate CP recording system service. In addition, when the virtual machine user ID is identified to CP in the appropriate SYSTEM_USERIDS system configuration file statement (ACCOUNT1, ACCOUNT2, EREP1, EREP2, SYMPTOM1, or SYMPTOM2), data is queued for the application even when the virtual machine is not logged on and CP is started with the COLD option. CP logs on these virtual machines as part of its initialization.
2. Consider adding a LOGONBY statement to the virtual machine definition for service machines. It lets the system administrators and system programmers log on to the service machine using their own log on password rather than that of the service virtual machine. This eliminates the need for many users to know the service virtual machine's log on password.
3. Consider adding the STGEXEMPT operand to the OPTION directory statement for service virtual machines. This operand ensures that the user ID is not subject to being stopped or forced due to the amount of free storage it causes CP to consume. STGEXEMPT is recommended to protect special purpose user IDs which are vital to the installation, user IDs running trusted code, and user IDs which should never be forced off the system.

Setting Up Virtual Machines for Accounting

The z/VM System DDR media or DVD includes:

- A sample virtual machine definition for an accounting virtual machine. This definition contains the required IUCV authorization for connecting to the CP accounting system service. The user ID for the virtual machine is DISKACNT.

  Note: One or two virtual machines can be set up as accounting virtual machines.
- A sample system configuration file that defines the user ID for the accounting virtual machine as DISKACNT. The user ID for the accounting virtual machine is defined as part of the SYSTEM_USERIDS statement in the system configuration file so that it is automatically logged on by CP.
- A sample PROFILE EXEC for the accounting virtual machine. The PROFILE EXEC is supplied on user ID DISKACNT's 191 A-disk.

If the accounting virtual machine is not logged on, use the XAUTOLOG command to log on the accounting virtual machine automatically. To do this for the *diskacnt* user ID, enter:.

```
xautolog diskacnt
```

Setting Up Virtual Machines to Collect Accounting Records

Your installation can set up multiple virtual machines to collect the accounting records. There are two methods that you can use to specify to CP which virtual machines will be used to retrieve accounting data. These are as follows:

- At CP generation time in the system configuration file, code the SYSTEM_USERIDS statement to specify the virtual machine IDs (up to two) for which the CP accounting system service (*ACCOUNT) is to accumulate records. If these users have a virtual machine definition in the user directory, they will be logged on as part of CP initialization. See "SYSTEM_USERIDS Statement" on page 294 for information about the SYSTEM_USERIDS statement.
- Additional virtual machines can also collect accounting records by following the instructions in sections:
  - See "Specifying a New Accounting Virtual Machine" on page 354.
  - See "Starting Manual Retrieval of Accounting Records" on page 354.

Specifying a New Accounting Virtual Machine

Normally your installation sets up particular virtual machines to be the accounting virtual machines. If you want to have additional virtual machines collect accounting records, do the following:

- Make sure the new accounting virtual machine has an A-disk available to receive the accounting records.
- Make sure the system programmer authorizes the new virtual machine to receive the records in the IUCV directory statement.

  The user entering the RETRIEVE utility must have a virtual machine definition containing an IUCV directory statement authorizing the virtual machine to connect to the CP system service which supports the type of record being collected. *ACCOUNT must be specified for ACCOUNT records.
- Follow the steps outlined in "Starting Manual Retrieval of Accounting Records" on page 354.

Starting Manual Retrieval of Accounting Records

Ordinarily, the accounting virtual machine starts retrieving automatically when you bring up z/VM. But there may be times when you need to start retrieval manually. For example, you may find out through error messages or by entering the QUERY RECORDING command that the accounting virtual machine has stopped retrieving. Or your installation may not set up the accounting virtual machine to start retrieval automatically. In either case, you may start retrieval manually, as follows:

1. If necessary, disconnect from the operator's virtual machine by entering:

   ```
   disconnect
   ```
2. Log on the accounting virtual machine.
3. Clear any activity in the virtual machine (including retrieval) by entering:

   ```
   #cp system reset
   ```
4. Load CMS into storage by entering:

   ```
   ipl 190
   ```

   or, if your installation has installed CMS in a named saved system,

   ```
   ipl cmsname
   ```

   where *cmsname* is the name of your CMS system.
5. Make sure there is room on the accounting virtual machine's A-disk for more accounting records. To find out how full the A-disk is, enter:

   ```
   query disk a
   ```

   If the A-disk is almost full, process some of the accounting records. Then erase the old files to free space on the A-disk.
6. Start retrieval of accounting records by entering:

```
retrieve account
```

7. Disconnect the accounting virtual machine by entering:

   ```
   #cp disconnect
   ```

   You may now use the display for another virtual machine.
8. If necessary, reconnect the operator's virtual machine.

Disassociating a User ID from the Retrieval of Accounting Records

1. Stop the recording of ACCOUNT records for this user ID. To do this from a user ID authorized for the class **A** or **B** version of the CP RECORDING command, enter:

   ```
   recording account off qid userid
   ```

   Or, the user ID being removed (must be authorized for the class **C**, **E**, or **F** version of the recording command), enter:

   ```
   recording account off
   ```

2. If there are accounting records queued in host storage for this user ID and you want to save the data, log on the user ID to be deleted and enter:

   ```
   retrieve account
   ```

   This will place the accounting records in the proper file for later processing. Disconnect this user by entering:

   ```
   #cp disconnect
   ```

   Log back on the authorized user ID.
3. If the accounting records queued in host storage for this user ID are not wanted, they may be purged. To do this from a user ID authorized for the class **A** or **B** version of the CP RECORDING command, enter:

   ```
   recording account purge qid userid
   ```

   Or, the user ID being removed (must be authorized for the class **C**, **E**, or **F** version of the recording command), enter:

   ```
   recording account purge
   ```

   This will PURGE all the accounting records queued in host storage for this user ID.
4. Verify the record count is zero and recording is off for this user ID. To do this, enter:

   ```
   query recording
   ```

   Note: The deleted user ID remains in the warm start data and in the output of the QUERY RECORDING command until VM is restarted with the COLD option.
5. If the user ID is specified on the SYSTEM_USERIDS statement in the system configuration file, remove that user ID.

   Note: The user ID OPERACCT is the default if no user ID for accounting is specified in the system configuration file.

Setting Up Virtual Machines for Error Recording

Setting up virtual machines for error recording is similar to setting up virtual machines for accounting. (One or two virtual machines can be used for error recording.) The z/VM System DDR media or DVD includes:

- A sample virtual machine definition for one error recording virtual machine. The user ID for the virtual machine is EREP.

  Note: The sample virtual machine definition for the error recording virtual machine (EREP) supplied on the z/VM System DDR media or DVD contains the required IUCV authorization to connect to the CP *LOGREC system service.
- A sample system configuration file that defines the user ID for an error recording virtual machine as EREP. The user ID for an error recording virtual machine is defined as part of the SYSTEM_USERIDS statement in the system configuration file. This allows the user ID to be automatically logged on by CP initialization and records queued to it.
- A sample PROFILE EXEC for one error recording virtual machine. The PROFILE EXEC is supplied on user ID EREP's 191 A-disk.

If the error recording virtual machine is not logged on, use the XAUTOLOG command to automatically log on the error recording virtual machine. To do this for the EREP user ID, enter:

```
xautolog erep
```

Setting Up Virtual Machines to Collect EREP Records

Your installation can set up multiple virtual machines to collect the EREP records. There are two methods that you can use to specify to CP which virtual machines will be used to retrieve error recording data. These are as follows:

- At CP generation time in the system configuration file, code the SYSTEM_USERIDS statement to specify the virtual machines (up to two) for which the CP Error Logging system service (*LOGREC) is to accumulate records. If these virtual machines are defined in the user directory, they will be logged on as part of CP initialization. See "SYSTEM_USERIDS Statement" on page 294.
- Additional virtual machines can also collect error recording data by following the instructions in sections:
  - See "Specifying a New EREP Virtual Machine" on page 356.
  - See "Starting EREP Record Retrieval Manually" on page 357.

For information about processing the EREP records, see "Processing EREP Records Collected by the EREP Virtual Machine" on page 358.

Specifying a New EREP Virtual Machine

Normally, your installation sets up particular virtual machines to be the EREP virtual machines. If you want to have additional virtual machines collect EREP records, then you must do the following:

1. Make sure the new EREP virtual machine has an A-disk available to receive the EREP records.
2. Make sure the system programmer had authorized the new virtual machine to receive the records in the IUCV directory statement.

   The virtual machine definition of the user entering the RETRIEVE utility must have an IUCV directory statement authorizing the virtual machine to connect to the CP system service which supports the type of record being collected. *LOGREC must be specified for EREP records.
3. Follow the steps outlined in the next section, "Starting EREP Record Retrieval Manually" on page 357.

Starting EREP Record Retrieval Manually

Ordinarily, the EREP virtual machine starts retrieving automatically when you bring up z/VM. But there may be times when you need to start retrieval manually. For example, you may find out through error messages or by entering the QUERY RECORDING command that the EREP virtual machine has stopped retrieving. Or your installation may not set up the EREP virtual machine to start retrieval automatically. In either case, you can start retrieval manually, as follows:

1. If necessary, disconnect from the operator's virtual machine by entering:

   ```
   disconnect
   ```

2. Log on the EREP virtual machine.
3. Clear any activity in the virtual machine (including retrieval) by entering:

   ```
   #cp system reset
   ```

4. Load CMS into storage by entering:

   ```
   ipl 190
   ```

   or, if your installation has installed CMS in a named saved system:

   ```
   ipl cmsname
   ```

   where *cmsname* is the name of your CMS system.
5. Make sure there is room on the EREP virtual machine A-disk for more EREP records. To find out how full the A-disk is, enter:

   ```
   query disk a
   ```

   If the A-disk is almost full, process some of the EREP records and erase the old files to free space on the A-disk. See "Processing EREP Records Collected by the EREP Virtual Machine" on page 358.
6. Start EREP record retrieval by entering:

   ```
   retrieve erep
   ```

7. Disconnect the EREP virtual machine by entering:

   ```
   #cp disconnect
   ```

   You may now use the display for another virtual machine.
8. If necessary, reconnect the operator's virtual machine.

Disassociating a User ID from the Retrieval of EREP Records

1. Stop the recording of EREP records for this user ID. To do this from a user ID authorized for the class **A** or **B** version of the CP RECORDING command, enter:

   ```
   recording erep off qid userid
   ```

   Or, the user ID being removed (must be authorized for the class **C**, **E**, or **F** version of the recording command), enter:

   ```
   recording erep off
   ```

2. If there are EREP records queued in host storage for this user ID and you want to save the data, log on the user ID to be deleted and enter:

   ```
   retrieve erep
   ```

This will place the EREP records in the proper file for later processing. Disconnect this user by entering:

```
#cp disconnect
```

Log back on the authorized user ID.

3. If the EREP records queued in host storage for this user ID are not wanted, they may be purged. To do this from a user ID authorized for the class **A** or **B** version of the CP RECORDING command, enter:

```
recording erep purge qid userid
```

Or, the user ID being removed (must be authorized for the class **C**, **E**, or **F** version of the recording command), enter:

```
recording erep purge
```

This will PURGE all the accounting records queued in host storage for this user ID.

4. Verify the record count is zero and recording is off for this user ID. To do this, enter:

```
query recording
```

Note: The deleted user ID remains in the warm start data and in the output of the QUERY RECORDING command until VM is restarted with the COLD option.

5. If the user ID is specified on the SYSTEM_USERIDS statement in the system configuration file, remove that user ID.

Note: The user ID OPEREREP is the default if no user ID for EREP is specified in the system configuration file.

Processing EREP Records Collected by the EREP Virtual Machine

If you need to process the EREP records that have been collected by the EREP virtual machine, either to clear some space on the EREP A-disk or to format the records for problem determination, you can use the following procedure to process and format the records.

Note: You can define a temporary disk for the output of the CPEREPXA utility if the A-disk is full. In that case, specify the file mode of the temporary disk in the FILEDEF commands for all files other than ACCIN.

1. Log on the EREP virtual machine.
2. End the retrieval of EREP records in the virtual machine by entering:

```
#cp external
```

Then enter the following response to the prompt:

```
ENTER END OR SUMMARY
end
```

3. The EREP virtual machine is now in CMS, and you can check the A-disk for an XAEREPIO RECORD file that has been used by EREP to accumulate records. This is the input file which contains the raw error records to be processed. If another RECORD file exists on the A-disk instead of XAEREPIO RECORD, use its name in place of XAEREPIO in the following steps.
4. Enter the following CMS FILEDEF commands to define the files to be used by the CPEREPXA utility to process the input file and create a history file:

```
FILEDEF ACCIN DISK XAEREPIO RECORD A (RECFM VB BLKSIZE 20000
FILEDEF ACCDEV DISK TESTACC RECORD A (RECFM VB BLKSIZE 20000
FILEDEF TOURIST DISK PRINTAL TOURIST A
FILEDEF EREPPT DISK PRINTAL OUTPUT A
```

ACCIN defines the input file, ACCDEV defines the accumulation file, TOURIST defines the file for messages and diagnostics, and EREPPT defines the file that will contain the report output. Creating the ACCIN, ACCDEV, and EREPPT file definitions before invoking CPEREPXA allows you to override the CPEREPXA defaults.

You can place these commands into a REXX exec to be used for processing future records.

To confirm that the file definitions were successfully created, issue the following command:

```
query filedef
```

5. Create a CPEREPXA control file (PRNTHIST PARM A) containing the following EREP parameters to be used for the first CPEREPXA invocation.

   Parameter
   Meaning

   PRINT=NO
   No reports will be generated.

   ACC=Y
   The selected error records will be accumulated in an output file.

   ZERO=Y
   The error recording files will be cleared after the error records are processed.

6. If desired, make a copy of the XAEREPIO RECORD file, as it will be deleted as part of the next step.
7. Invoke the CPEREPXA utility with the PRNTHIST control file to convert the raw file (XAEREPIO RECORD) to a history file (TESTACC RECORD). This history file will be used as the input file in the second CPEREPXA invocation. The CPEREPXA utility is on the S-disk (MAINT 190).

   ```
   cperepxa prnthist parm a
   ```

   Check the A-disk to make sure the TESTACC RECORD file was created if there were raw records to process.

8. Enter the following CMS FILEDEF command to define TESTACC RECORD as the input file for the second CPEREPXA invocation:

   ```
   FILEDEF ACCIN DISK TESTACC RECORD A
   ```

   To confirm that the ACCIN file definition was successfully changed, issue the following command:

   ```
   query filedef
   ```

9. Create a CPEREPXA control file (PRINTAL PARM A) containing the following EREP parameters to be used for the second CPEREPXA invocation.

   Parameter
   Meaning

   PRINT=AL
   All of the detail print reports will be generated.

   HIST=Y
   The error records to be processed are in a history file that was created as an accumulation file in a previous CPEREPXA invocation.

   ACC=N
   The selected error records will not be accumulated in an output file.

   LINECT=60
   Specifies the maximum line count for each page of the report output.

10. Invoke the CPEREPXA utility with the PRINTAL control file to process the error records and generate the print reports:

    ```
    cperepxa printal parm a
    ```

11. The PRINTAL OUTPUT A file will now contain all of the formatted records, which can be used as needed.

    The PRINTAL OUTPUT, PRINTAL TOURIST, and TESTACC RECORD files should be deleted prior to restarting error record retrieval in the virtual machine. This will ensure that no error records will be combined in successive processing.
12. Run the default PROFILE EXEC on the EREP user ID to restart the EREP record retrieval:

    ```
    profile
    ```
13. Disconnect the EREP virtual machine by entering:

    ```
    #cp disconnect
    ```

Setting Up Virtual Machines for Symptom Record Recording

Symptom record recording support allows your installation to record symptom records in CMS files separate from dumps or processor and I/O (LOGREC) errors. CP provides additional information in a symptom record when appropriate. You can use the Dump Viewing Facility VIEWSYM command to view, compare, and select multiple symptom records in CMS files.

A data retrieval application can connect to the CP *SYMPTOM system service through the IUCV facility to collect symptom records. A virtual machine that contains such an application must have an IUCV directory statement in its virtual machine definition that identifies the CP *SYMPTOM system service. In addition, when you identify the virtual machine user ID to CP in either the SYSTEM_USERIDS SYMPTOM1 or SYMPTOM2 statement in the system configuration file, data is queued for the virtual machine even when the virtual machine is not logged on and the z/VM system is started with the COLD option. The virtual machine is also logged on by z/VM as part of initialization.

Setting up virtual machines for symptom record recording is similar to setting up virtual machines for accounting and error recording. (One or two virtual machines can be used for symptom record recording.) The z/VM System media or DVD includes:

- A sample virtual machine definition for one symptom record recording virtual machine that contains the required IUCV authorization to connect to the CP *SYMPTOM system service. The user ID for the virtual machine is OPERSYMP.
- A sample system configuration file that *does not* define the user ID for a symptom record recording virtual machine. The user ID for the symptom record recording virtual machine therefore defaults to OPERSYMP.
- A sample PROFILE EXEC for one symptom record recording virtual machine. The PROFILE EXEC is supplied on user ID OPERSYMP's 191 A-disk.

If the symptom record recording virtual machine is not logged on, use the XAUTOLOG command to automatically log on the symptom record recording virtual machine. To do this for the *opersymp* user ID, enter:

```
xautolog opersymp
```

Setting Up Virtual Machines to Collect Symptom Records

Your installation can set up multiple virtual machines to collect the symptom records. There are two methods that you can use to specify to CP which virtual machines will be used to retrieve symptom data. These are as follows:

- At CP generation time in the system configuration file, code the SYSTEM_USERIDS statement to specify the virtual machine IDs (up to two) for which the CP symptom system service (*SYMPTOM) is to accumulate records. If these users have a virtual machine definition in the user directory, they will be logged on as part of CP initialization. See "SYSTEM_USERIDS Statement" on page 294 for information about the SYSTEM_USERIDS statement.

- Additional virtual machines can also collect symptom records by following the instructions in sections:
  - See "Specifying a New Symptom Record Recording Virtual Machine" on page 361.
  - See "Starting Manual Retrieval of Symptom Records" on page 361.

Specifying a New Symptom Record Recording Virtual Machine

Normally your installation sets up particular virtual machines to be the symptom virtual machines. If you want to have additional virtual machines collect symptom records, do the following:

- Make sure the new symptom virtual machine has an A-disk available to receive the symptom records.
- Make sure the system programmer authorizes the new virtual machine to receive the records in the IUCV directory statement.

  The user entering the RETRIEVE utility must have a directory containing an IUCV directory statement authorizing the virtual machine to connect to the CP system service which supports the type of record being collected. *SYMPTOM must be specified for SYMPTOM records.
- Follow the steps outlined in "Starting Manual Retrieval of Symptom Records" on page 361.

Starting Manual Retrieval of Symptom Records

Ordinarily, the symptom virtual machine starts retrieving automatically when you bring up z/VM. But there may be times when you need to start retrieval manually. For example, you may find out through error messages or by entering the QUERY RECORDING command that the recording virtual machine has stopped retrieving. You may also start retrieval manually, as follows:

1. If necessary, disconnect from the operator's virtual machine by entering:

   ```
   disconnect
   ```
2. Log on the symptom virtual machine.
3. Clear any activity in the virtual machine (including retrieval) by entering:

   ```
   #cp system reset
   ```
4. Load CMS into storage by entering:

   ```
   ipl 190
   ```

   or, if your installation has installed CMS in a named saved system,

   ```
   ipl cmsname
   ```

   where *cmsname* is the name of your CMS system.
5. Make sure there is room on the symptom virtual machine's A-disk for more symptom records. To find out how full the A-disk is, enter:

   ```
   query disk a
   ```

   If the A-disk is almost full, process some of the symptom records. Then erase the old files to free space on the A-disk.
6. Start retrieval of symptom records by entering:

   ```
   retrieve symptom
   ```
7. Disconnect the symptom virtual machine by entering:

   ```
   #cp disconnect
   ```

   You may now use the display for another virtual machine.
8. If necessary, reconnect the operator's virtual machine.

Disassociating a User ID from the Retrieval of Symptom Records

1. Stop the recording of SYMPTOM records for this user ID. To do this from a user ID authorized for the class **A** or **B** version of the CP RECORDING command, enter:

   ```
   recording symptom off qid userid
   ```

   Or, the user ID being removed (must be authorized for the class **C**, **E**, or **F** version of the recording command), enter:

   ```
   recording symptom off
   ```

2. If there are symptom records queued in host storage for this user ID and you want to save the data, log on the user ID to be deleted and enter:

   ```
   retrieve symptom
   ```

   This will place the symptom records in the proper file for later processing. Disconnect this user by entering:

   ```
   #cp disconnect
   ```

   Log back on the authorized user ID.
3. If the symptom records queued in host storage for this user ID are not wanted, they may be purged. To do this from a user ID authorized for the class **A** or **B** version of the CP RECORDING command, enter:

   ```
   recording symptom purge qid userid
   ```

   Or, the user ID being removed (must be authorized for the class **C**, **E**, or **F** version of the recording command), enter:

   ```
   recording symptom purge
   ```

   This will PURGE all the symptom records queued in host storage for this user ID.
4. Verify the record count is zero and recording is off for this user ID. To do this, enter:

   ```
   query recording
   ```

   Note: The deleted user ID remains in the warm start data and in the output of the QUERY RECORDING command until VM is restarted with the COLD option.
5. If the user ID is specified on the SYSTEM_USERIDS statement in the system configuration file, remove that user ID.

   Note: The user ID OPERSYMP is the default if no user ID for SYMPTOM is specified in the system configuration file.

Setting Up a Virtual Machine for Communication Controller Support for Emulator Program (EP)

To manage a 3745 communication controller, one needs to set up a special service virtual machine. Unlike other service virtual machines such as the accounting virtual machine, the z/VM System DDR media or DVD does not include a sample virtual machine definition for a communication controller virtual machine. This support requires that Advanced Communications Function for Systems Support Programs (ACF/SSP) be installed on your system.

The networking facilities available with communication controllers are provided through VM/Pass-Through (PVM) licensed program. PVM requires a separate virtual machine to perform the loading and dumping of communication controllers.

The CCLOAD utility, which can be issued by the disconnected service machine, provides automatic loading for the communication controller. The CCDUMP utility can be used to format a communication controller dump file produced by CCLOAD. For the complete syntax of these utilities, see *z/VM: CP Commands and Utilities Reference*.

CCLOAD requires access to the read/write A-disk. Loading requires space for SYSIN and SYSPRINT files. If CCLOAD is used for dumping the communication controller either automatically or by user request, it requires enough disk space for the dump files. For direct access storage device (DASD) requirements, see the network program products publications.

Use CCDUMP to format and print the dump files produced by CCLOAD. If you provide another user with shared access to the A-disk of the service machine, CTLR dump files can be printed offline while the service machine monitors the CTLR. If you specify DISK, you must have enough space on the A-disk for the formatted output file. Refer to the network program products publication for DASD requirements. When you format the dump on a printer (PRINTER option), messages do not appear on your console but are included in the SYSPRINT file. If you specify DISK, messages included in the SYSPRINT file are displayed on the console.

To erase the input dump file, you must have write access to the disk where it resides. Do not specify the ERASE option if you are sharing access to files on another user's A-disk (for example, those generated by the CCLOAD EXEC on a disconnected service machine).

The following examples show you how to set up a service virtual machine to perform utility functions for a communication controller:

1. Create a virtual machine definition for each communication controller. Use the virtual machine definition in Figure 5 on page 363 as a sample, but make sure you tailor the directory statements to match your system configuration and network licensed program requirements.

```
USER EP0F7 NOPASS 1M 4M G
** This virtual machine requires only a minimal amount
** of storage and the G privilege class.
**
ACCOUNT 10
AUTOLOG AUTOLOG1 PVM OP1 MAINT
** Depending on how you set up your network, you will want
** this virtual machine to be autologged by either the
** system or another service virtual machine such as
** VM/Pass-Through (PVM).
**
IPL CMS
CONSOLE 01F 3215 T OP1
** If you want the capability to interrupt the service
** virtual machine to manually dump and then reload the
** communication controller, or to terminate the service
** virtual machine, specify a secondary user ID (OP1) on
** the CONSOLE statement.
**
SPOOL 00C 2540 READER A
SPOOL 00D 2540 PUNCH A
SPOOL 00E 1403 A
**
DEDICATE 0F7 0F7
** Dedicates the communication controller to this user.
**
LINK MAINT 343 343 RR RMAINT
** Provides access to ACF/SSP.
**
MDISK 191 3390 900 40 XADISK MR
** Provides space for dumps of the communication controller.
```

Figure 5. Sample Virtual Machine Definition for a Communication Controller

2. Set up a PROFILE EXEC for the EP0F7 user ID. Use the sample in Figure 6 on page 364 as a guideline.

```
access 343 b/a
** Access the ACF/SSP disk.
**
queue 'RUN0F7'
** Stack a command to execute the CCLOAD command.
**
```

Figure 6. Sample PROFILE EXEC for EP0F7 User ID

3. Set up the RUN0F7 EXEC to run the CCLOAD EXEC and inform the VM/Pass-Through virtual machine that the communication controller is active, as follows:

```
/**/'exec ccload 0f7 ep0f7 cmd(smsg pvm exec ep0f7)'
```

Note: You may need to stop CCLOAD processing while the service virtual machine is disconnected. For example, you may need to free space for dump files. To have the service virtual machine resume monitoring the communication controller, have the secondary user ID (OP1 in this case) issue SEND EP0F7 RUN0F7.

PVM Example

The following examples illustrate a possible use of the service virtual machine. Refer to PVM publications for details.

The PVM configuration file shown in Figure 7 on page 364 contains definition of EP (emulator program) lines.

```
…
*
* BSC lines on the EP communications controller
* whose base subchannel address is 0F7
*
LINK 030 B0F7A R3270
…
LINK 031 B0F7B R3270
…
LINK 032 B0F7C R3270
…
LINK 033 B0F7D R3270
…
LINK 034 B0F7E R3270
…
LINK 035 B0F7F R3270
…
LINK 036 B0F7G R3270
…
LINK 037 B0F7H R3270
…
```

Figure 7. Sample PVM Configuration File

PROFILE PVM additions to use this support are shown in Figure 8 on page 364.

```
* For each EP communications controller, whose lines are
* owned by PVM, authorize its utility service virtual
* machine to send commands to PVM and AUTOLOG the user ID.
*
AUTHORIZ EP0F7
CP XAUTOLOG EP0F7
…
```

Figure 8. PROFILE PVM Additions

The EP0F7 PVM file shown in Figure 9 on page 365 is used to start the EP lines automatically.

```
* START LINE commands for PVM to execute when signalled
* by EP0F7 userid executing CCLOAD for CTLR device 0F7
*
CP VARY ON 030
CP ATTACH 030 *
START LINE 030
CP VARY ON 031
CP ATTACH 031 *
START LINE 031
CP VARY ON 032
CP ATTACH 032 *
START LINE 032
CP VARY ON 033
CP ATTACH 033 *
START LINE 033
CP VARY ON 034
CP ATTACH 034 *
START LINE 034
CP VARY ON 035
CP ATTACH 035 *
START LINE 035
CP VARY ON 036
CP ATTACH 036 *
START LINE 036
CP VARY ON 037
CP ATTACH 037 *
START LINE 037
```

Figure 9. Sample EP0F7 File

When the 3745 is first loaded by the service machine and PVM issues the VARY and ATTACH commands, the following response is displayed:

```
xxxx varied online
Line xxxx attached to PVM xxxx
```

Following recovery of a 3745 failure, where the EP control program has been reloaded, CP still considers the EP lines online and attached. When the same PVM command list is executed by the service machine, the VARY command displays the following response message:

```
xxxx already online
```

ATTACH displays the following error message for each line:

```
HCPATR122E Line xxxx already attached to PVM
```

Setting Up Service Pool Virtual Machines

You can establish a collection of virtual machines that are logged on over extended periods. Certain applications use this collection to create tasks on behalf of a user. Such a collection is called a service pool.

The pool of virtual machines, each initialized by an application running on CMS, is available to handle requests for services from programmable workstations. The application makes a logical connection between the functions the virtual machines are executing. These services are managed through a VTAM application running on the Advanced Program-to-Program Communications/VM VTAM Support (AVS) of z/VM. The VTAM application uses an LU 6.2 APPC/VM session to invoke these services. For additional information on AVS, see *z/VM: Connectivity*. The application that controls the service pool is called the service pool manager. The service pool manager provides services such as autologging the service pool virtual machines and selecting one to handle a request from a workstation.

To define a range of virtual machines to be defined as a pool, code the POOL directory statement. This assigns a virtual machine definition for each virtual machine to be included in the pool. See "POOL Directory Statement" on page 581 for information on coding the POOL directory statement. If you code the POOL statement in a virtual machine definition, it is assigned the only statement other than a USER or IDENTITY statement in that virtual machine definition. Each virtual machine in the pool must have the same first three characters in its user ID, followed by a number in the range specified in the POOL statement.

You must code other directory statements that apply to the range of virtual machines in the pool in a profile entry in the directory (see "PROFILE Directory Statement" on page 591). The pool definition uses the PROFILE statement as the common user definition information. The PROFILE name must have been defined in the directory before any POOL statement that refers to it.

A password (AUTOONLY) specifies that a virtual machine can only be autologged. Thus, a user at a terminal cannot enter a LOGON command for a virtual machine with the AUTOONLY password. This provides a security and data integrity function for service pool virtual machines without having to administer different, changing passwords for the pool of virtual machines.

Setting Up Virtual Machines for Data Storage Management

The DFSMS/VM is the foundation for system-managed storage in the VM environment. A DFSMS/VM service machine uses the Interactive Storage Management Facility (ISMF) to reduce the amount of manual work that must be performed when data is migrating to newly installed DASD. ISMF automatically formats new minidisks, updates user directories, and automates other functions, providing both speed and predictability during data migration activities.

You can use a DFSMS/VM service virtual machine to provide information about other storage management tasks. For example, you can use DIAGNOSE X'08' to determine the devices allocated to the system. The VMUDQ macroinstruction queries user directory minidisk definitions.

You can configure a DFSMS/VM virtual machine to ensure data integrity and security when you move data to new storage locations. ISMF provides a check utility that performs an integrity evaluation of a CMS minidisk to verify whether its file structure is intact. To ensure data security during and after move operations, a DFSMS/VM machine can rely on the usual access authority checking mechanisms, such as ACI and password validation, as well as stable and exclusive link access options that require specific authorization to allow user access to data.

The LNKSTABL and LNKEXCLU options, specified with the OPTION directory statement for a particular virtual machine, give that machine authority to specify access modes that limit the ability of other users to obtain link access to a minidisk while the service machine is linked to that disk. However, the use of these access-mode options can potentially tie up the system if improperly or indiscriminately used and should be treated as a special resource.

These options are described in more detail in "Using Link Access Control Options" on page 383. In addition, further information on access modes can be found in "LINK Directory Statement" on page 533 and "MDISK Directory Statement" on page 550. The VMUDQ macroinstruction is described in *z/VM: CP Programming Services*.

Chapter 9. Planning for SNA Console Communication Services (SNA/CCS)

The Systems Network Architecture console communication services (SNA/CCS) provide a total data communication structure for transmitting information by using a communications network. SNA communication products perform functions traditionally handled by the main processor (for example, management of communication lines, device-dependent characteristics and control, and data formatting).

SNA/CCS provides full z/VM console capabilities to operators on SNA/CCS terminals. You can use SNA/CCS terminals as virtual machine consoles. The information is transparent. If you are planning to use SNA/CCS processing, you must consider the topics discussed in this chapter.

This chapter describes the SNA environment and tells you how to:

- Establish the SNA/CCS terminal environment
- Do tracing for SNA/CCS.

Structure of the SNA Environment

Three major components contribute to SNA/CCS terminal support:

- SNA console communication services (SNA/CCS)
- Inter-user communication vehicle (IUCV)
- One of the following:
  - VTAM SNA Console Support (VSCS)
  - VTAM Communications Network Application (VCNA) licensed program.

IUCV and SNA/CCS are part of z/VM.

SNA/CCS terminal support is provided through a virtual machine. The VTAM service machine (VSM) is the virtual machine that acts as an interface between SNA/CCS and the SNA network. It runs either VSCS or VCNA. VSCS and VCNA both work with the Advanced Communications Function for Virtual Telecommunications Access Method (ACF/VTAM) in doing their job.

Usually, ACF/VTAM is also in the VTAM service machine. VSCS, however, may be in a different virtual machine. ACF/VTAM manages the SNA network, and VSCS or VCNA acts as the interface to CP.

A VCNA VTAM service machine also requires one of the following operating systems with External Interrupt Support (EIS):

- VSE/Advanced Functions licensed program (latest level)
- OS/VS1 with Basic Programming Extensions licensed program.

VSCS does not require a guest operating system but does require GCS.

Establishing the SNA/CCS Terminal Environment

To allow SNA/CCS terminals to access z/VM:

- VSM must be defined to the system
- SNA/CCS must be defined to the VSM
- SNA communications must be enabled
- VSM must be started in its own virtual machine
- VSM must establish a global IUCV connection with SNA/CCS.

After these tasks are accomplished, individual SNA terminal users may access z/VM through the terminals managed by that VSM.

Defining the VSM to z/VM

The virtual machine definition for the virtual machine that contains the VSM:

- Must authorize the VSM to use IUCV to communicate with SNA/CCS using the IUCV directory statement.
- Should specify PRIORITY on the IUCV directory statement to allow the use of IUCV priority messages.
- May specify how many IUCV connections are allowed by using the MAXCONN operand on the OPTION directory statement. Note that the MAXCONN operand determines how many SNA/CCS terminal users may access z/VM through the VSM.
- May specify that the virtual machine can be automatically logged on by other virtual machines using the XAUTOLOG directory statement.
- May specify that a named saved system or device is to be loaded when the virtual machine is logged on using the IPL directory statement.
- May specify a secondary user ID on the CONSOLE directory statement if secondary console support is to be used in the operation of the VSM.

Defining Logos Used by VSMs

Logos used by VSMs are selected via the logo configuration file. You can choose to have one logo picture file for all VSMs, or you can choose a different logo picture for each VSM in your system. You use CHOOSE_LOGO statements in the logo configuration file to tell CP which logos to use for terminals logging on through VSMs.

For more information about logos used by VSMs, refer to Chapter 7, “The Logo Configuration File,” on page 337; for details, see “CHOOSE_LOGO Statement” on page 342.

Defining SNA/CCS to VCNA

The VCNA START parameters that relate to SNA/CCS are described below. They are all operands of the DTIGEN macroinstruction, which is optional.

CCSMLM
specifies the CP message limit. This is the maximum number of messages to be specified on the IUCV ACCEPT request.

LGNCMDS
specifies the CP commands that are valid before logon.

PACE
specifies the pacing value for communication between VM/VCNA and CP. This value is the maximum number of I/O requests per terminal from CP to VM/VCNA before VM/VCNA must return a pacing response. It is provided to prevent a buffer overrun in VM/VCNA.

RDSPTMR
specifies the redisplay timer value in sixteenths of a second. CP uses this value to force VM/VCNA to redisplay data entered and to unlock the keyboard after the prescribed time has elapsed.

TIMEINT
specifies an interval in seconds that VM/VCNA uses to determine how often timer functions such as MORE, HOLDING, and NOT ACCEPTED are to be performed. A smaller value gives greater accuracy with more timer overhead.

VCNAMLM
specifies the IUCV VM/VCNA message limit. This limit is the maximum number of messages specified by VM/VCNA on the IUCV CONNECT request.

Defining SNA/CCS to VSCS

The VSCS START parameters that relate to SNA/CCS are described below. They are all operands on the DTIGEN macroinstruction, which is optional.

LGNCMDS
specifies the CP commands that are valid before logon.

RDSPTMR
is the redisplay timer value. CP uses this value to force VSCS to redisplay entered data and to unlock the keyboard after the prescribed time.

W3767L
is the default line size for 3767 and 3777 terminals. This value is the initial setting of the CP TERMINAL LINESIZE command when the user logs on with a 3767 or 3777 terminal.

W2741L
is the default line size for 2741 terminals. This value is the initial setting of the CP TERMINAL LINESIZE command when the user logs on with a 2741 terminal.

WTWXL
is the line length for teletype keyboard printers.

VSAMLM
is the maximum number of messages specified on IUCV ACCEPT and CONNECT requests.

DPACE
is the pacing value to be used with display terminals.

KPACE
is the pacing value to be used with keyboard terminals and printers.

Enabling SNA Communication

Before a VSM can establish communication with SNA/CCS, you must enable SNA communication. In a multiple VSM environment, you must selectively enable or disable SNA communication for a particular VSM. Use the CP ENABLE and DISABLE commands to enable and disable SNA communications.

Starting the VTAM Service Machine

The VSM operator must initialize the VSM before a user can log on to a terminal controlled by the VSM. The VSM operator logs on to z/VM from a natively controlled terminal to create a virtual machine in which the VSM can be initialized. The VSM consists of the programs loaded into this virtual machine: the operating system, the access method, and VM/VCNA or VSCS. After the VSM is operational, the VSM operator can disconnect, and further commands to the VSM can be entered from a secondary console (perhaps the system operator's console), freeing the virtual machine terminal.

An alternative method, also using secondary console support, does not require a virtual machine terminal that is dedicated to VSM. The VSM is automatically logged on in disconnected mode during CP initialization using XAUTOLOG. Then a PROFILE EXEC (VCNA) or a PROFILE GCS EXEC (VSCS) can bring up the system (saved or newly IPLed), and the VSM can be operated from the designated secondary console from the start.

Improving SNA/CCS Performance

Because SNA/CCS is implemented using a service virtual machine, you can improve the performance of the SNA/CCS environment by allocating more real processor resources to the VSMs that control the terminals connected to CP. Use the CP SET SRM command or the SHARE directory statement to do this. This performance consideration is not specific to SNA/CCS VSMs but applies to all service virtual machines. For more information on the CP SET SRM command, see the *z/VM: CP Commands and Utilities Reference*. For more information on the SHARE statement, see "SHARE Directory Statement" on page 595. For more information on performance and the use of the CP SET SRM command and the SHARE statement, see the *z/VM: Performance* book.

Some installations may see a performance benefit by using the IPOLL function with GCS to poll for pending replies and messages. By allowing GCS to retrieve up to 102 interrupts on each IPOLL, a significant reduction in the number of interrupts to VSCS when it is flooded with incoming IUCVs can be expected. The benefit is greatest when the VSCS virtual machine is loaded with work. The GCS command SET IPOLL ON enables this function.

VSM Termination

You can shut down a VSM in the following ways:

- You can use normal CP shutdown processing.
- You can terminate VTAM by using the VTAM HALT command.
- You can deactivate VCNA or VSCS by using the ACF/VTAM commands.
- The VSM operator can enter the VCNA or VSCS QUIT command to terminate VCNA or VSCS.
- The z/VM operator or the VSM operator (if authorized) can enter a CP FORCE command to remove individual users. This, combined with entering the DISABLE SNA command to prevent any new SNA/CCS connections, in effect shuts down the VSM.

Chapter 10. Customizing the CP Message Function

PI

You can customize the CP message function for your installation by adding code to module HCPMSU, as described in this chapter. HCPMSU is an installation-wide exit.

CP Exit 1210 is provided to allow that same tailoring of the system. If this exit point is activated then the HCPMSU installation code will not be invoked. For more information on CP Exit 1210, see *z/VM: CP Exit Customization*.

HCPMSU Module

Module HCPMSU allows alteration of the following message function commands: MESSAGE, MSGNOH, SMSG, and WARNING. You can include code in this module to do any of the following:

- Add further verification
- Change the format or context of the message
- Alter parameters that control the way the message is displayed on the screen
- Process installation-defined message command options.

HCPMSU is included in the CP module at system generation. Later, during system operation, when a user issues a message command, the CP message function calls HCPMSU before sending the formatted message to the designated receiver. If no code has been added to HCPMSU, the module performs no meaningful function.

HCPMSU contains one entry point, HCPMSUEX. Figure 10 on page 371 provides an overview of the IBM-supplied HCPMSU module, showing the entry point and the area where you can add your own source code.

```
         COPY  HCPOPTNS
HCPMSU   HCPPROLG ATTR=(RESIDENT),BASE=(R12)
*
         COPY  HCPEQUAT               common system equates
         COPY  HCPSAVBK               savearea block
*
HCPMSUEX HCPENTER CALL,SAVE=DYNAMIC
****************************************************************
*
*   Installation-supplied code can be inserted here.
*
****************************************************************
         SLR   R15,R15                Set return code of 0
MSUEXEND DS    0H                     Load regs and exit
         ST    R15,SAVER15            Store return code
         L     R11,SAVER11            Restore register 11
         HCPEXIT EP=HCPMSUEX
         HCPDROP R13
*
         HCPEPILG
```

Figure 10. Module HCPMSU Overview

HCPMSU has the following requirements and restrictions:

1. HCPMSU is written in Assembler H coding language.
2. HCPPROLG, HCPENTER, HCPDROP, HCPEXIT, and HCPEPILG are IBM-supplied macros (in HCPPSI MACLIB) that must be preserved in HCPMSU and used only as shown.
3. The attributes of HCPMSU must be REENTRANT and RESIDENT.
4. Links and accesses from HCPMSU to other modules, data areas, and control blocks are *not* supported.

5. IBM reserves the right to modify the code associated with this exit. IBM will attempt to make any changes as transparent as possible to the customer.
6. Upon entry to HCPMSUEX, register 13 points to a save area mapped by HCPSAVBK, see CP Accounting Exit in *z/VM: CP Exit Customization*. The caller's registers are saved in the save area. A ten-fullword work area, beginning at label SAVEWRK, is available for use by installation-supplied code.
7. Register 11, which points to the address of the dispatched VMDBK, and Register 13, which points to the address of the save area, must not be modified by installation-supplied code.

Entry Point HCPMSUEX

Entry point HCPMSUEX supports installation modifications to the MESSAGE, MSGNOH, WARNING, and SMSG commands. If the specified receiver of a message from one of these commands is logged on and has not turned off display of that message type (using the SET MSG OFF, SET SMSG OFF, or SET WNG OFF command), the CP message function verifies and formats the message, sets up the proper sending parameters, and calls HCPMSUEX before sending the message.

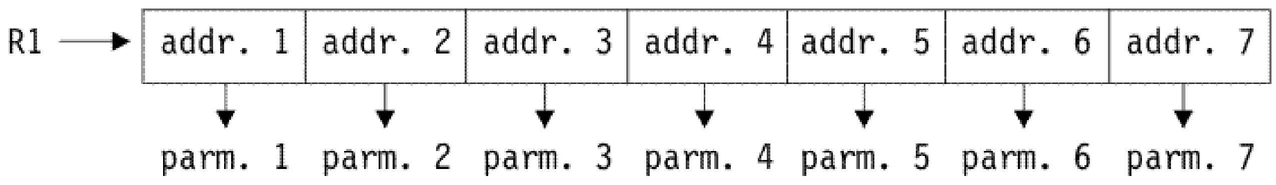
The CP message function uses general-purpose register 1 for input to entry point HCPMSUEX. Register 1 contains the address of a PLIST, which contains a list of addresses. Each fullword address points to a parameter being passed to the module. See "Parameter Values" on page 372.

Upon return to the CP message function from entry point HCPMSUEX, general-purpose register 15 must contain one of the following return codes:

Value	Meaning
0	Normal processing continues.
nnnn	An error was detected during HCPMSUEX exit processing. The installation-defined return code *nnnn* can be in the range of 0001 to 9999. The CP message function stops processing and issues the following message: `HCPMFS6600E  An error was detected by installation-wide exit HCPMSU – return code nnnn.` See *z/VM: CP Messages and Codes* for further information on this message.

Parameter Values

General-purpose register 1 contains the address of a PLIST that contains a list of addresses, each pointing to a parameter, as shown in the following figure.



The parameters contain the following values:

1. The first 8 bits of this 32-bit field control the way the message is displayed. More than one bit may be set. You can change the settings to change the way the message is displayed. The possible values are:

Value	Meaning
X'80______'	A time stamp is appended to the message.
X'40______'	No time stamp is appended to the message.
X'20______'	The message is highlighted when displayed.
X'10______'	The console alarm sounds when the message is displayed.
X'08______'	This is a high-priority message and is displayed immediately.

Value	Meaning
X'04______'–X'______01'	Reserved for IBM use.

Note: If neither of the first two bits, which control the time stamp, are set, the receiving user's value is used. If both bits are set, a time stamp is appended.

The supplied display values for the message commands are:

Command	Value	Meaning
MESSAGE	X'B0'	Time stamp is appended; message is highlighted; console alarm sounds.
MSGNOH	X'70'	No time stamp is appended; message is highlighted; console alarm sounds.
SMSG	X'00'	This type of message is not displayed.
WARNING	X'B8'	Time stamp is appended; message is highlighted; console alarm sounds; message is high-priority, displayed immediately.

2. This 32-bit field consists of three subfields:

   a. The first 8 bits indicate the type of message being processed. You can change the setting to change the way the message is processed. Note that if you change this setting, you might also need to change other parameters. The possible values are:

Value	Meaning
X'80'	MESSAGE command processing
X'40'	MSGNOH command processing
X'20'	SMSG command processing
X'10'	WARNING command processing
X'08'–X'01'	Reserved for IBM use

   Note: If none of these bits are set, the message type defaults to MESSAGE command processing. If more than one bit is set, the message processing is set to the type indicated by the first bit set.

   b. The next 8 bits indicate which IBM class commands the issuer of the message is allowed to execute. More than one bit may be set. The settings are dependent on the privilege class of the issuer of the message as well as the privilege class of the command entered. This field is included for reference only. Changes made to this field do not affect the way the command is processed. The possible values are:

Value	Meaning
X'80'	IBM class A commands
X'40'	IBM class B commands
X'20'	IBM class C commands
X'10'	IBM class D commands
X'08'	IBM class E commands
X'04'	IBM class F commands
X'02'	IBM class G commands
X'01'	IBM class H commands
X'00'	IBM class ANY

   c. The final 16 bits are reserved for IBM use.
3. This 8-byte field contains the user ID of the issuer of the command. This field is included for reference only. Changes made to this field do not affect the way the command is processed.
4. This 32-bit field contains the length of the message being processed. It includes the length of any installation-defined option that has been specified. Changes to this field might affect the way the message is displayed.
5. This 32-bit field contains the address of the message being processed. For each type of message, the message buffer looks as follows:

Message Type	Buffer Contents
MESSAGE	`* MSG FROM` *userid* `:` *text*
MSGNOH	*text*
SMSG	*text*
WARNING	`* WNG FROM` *userid* `:` *text*

   Note: The MESSAGE and WARNING messages always contain an initial blank and provide eight spaces for the user ID before the colon regardless of the actual length of the user ID.

   Following the text is 100 bytes of unused space which can be used to add to or modify the existing text. If the length of the message is changed, the length field described above should be changed to reflect this or not all text will be displayed.

   You can change the address of the message (and also the length of the message, if needed) for such purposes as, for example, pointing around the message header so it is not displayed. However, when the CP message function releases the storage it uses for the message, it releases the storage based on the address passed to HCPMSUEX, not the updated value.
6. This 8-byte field contains the user ID of the receiver of the message. If the value of the receiver's user ID is changed, the message is sent to the new receiver. However, if an error message is issued during normal message-function processing, and the error message includes the receiver's user ID, the user ID used in the error message is the one that was passed to HCPMSUEX, not the changed value.
7. This 32-bit field contains the length of the header for the message being processed. Each message is made up of a header and a text portion. The header includes the header information and the blank delimiter which precedes the text.

   The header length is used by the message processing function when messages are sent across the *MSG, *MSGALL, and VMCF services. Only the text of the message is sent. The header is discarded. The header length allows the message function to determine where the message text begins.

   Each message type is considered to have a header. For MESSAGE and WARNING types, the header length on entry to HCPMSUEX is 22 characters long. For MSGNOH and SMSG, the header length on entry to HCPMSUEX is 0 characters long.

   The header can be increased or decreased by this exit or by later message processing (for example, when a time stamp is added). SMSG never transmits the header while the other types display a header at the terminal if one is present.

Changing and Adding to the CP Message Function

There are two categories of CP message command modifications:

- A global change is one that takes effect each time the command is used.
- A local change is an installation-defined command option, which takes effect only when the option is used. If you define a new option, you determine where you want the option to appear in the command string. For example:

  ```
  MESSAGE userid option text
  ```

The CP message function processes the *option* as part of the text and passes it to HCPMSUEX as such.

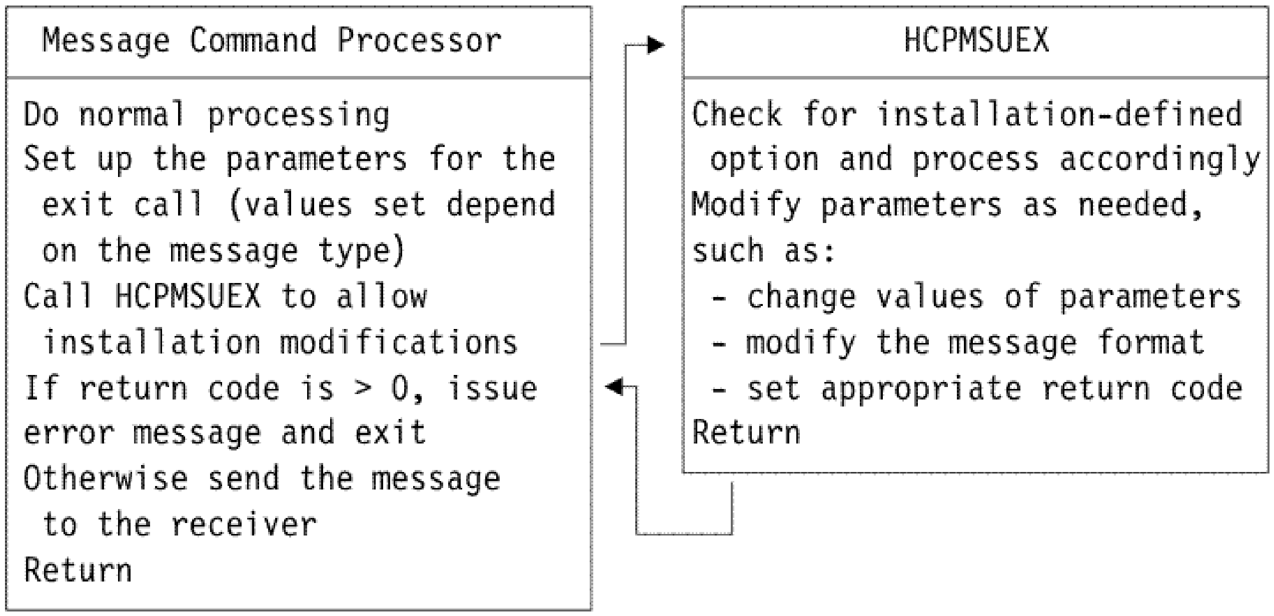

Message Command Processor
Do normal processing
Set up the parameters for the exit call (values set depend on the message type)
Call HCPMSUEX to allow installation modifications
If return code is > 0, issue error message and exit
Otherwise send the message to the receiver
Return
HCPMSUEX
Check for installation-defined option and process accordingly
Modify parameters as needed, such as:
- change values of parameters
- modify the message format
- set appropriate return code
Return

Figure 11. Flow of CP Message Function Customization

Figure 11 on page 375 illustrates the process of customizing the CP message function. This process is described in the following steps:

1. The message command processor sets up the parameters for the exit call. General-purpose register 1 contains the address of a PLIST that contains a list of addresses, each pointing to a parameter. Parameter values depend on the message type.
2. The message command processor calls entry point HCPMSUEX:
   a. If you have defined a new message command option, add code to HCPMSUEX to parse the option from the message text and process it accordingly.
   b. Add code to HCPMSUEX to do your global modifications. The processing required might depend on the values of the parameters passed to HCPMSUEX, which are dependent on the type of message being processed. You can change these values, reformat the message, and do additional verification. See "Parameter Values" on page 372 for a description of each field and possible values.
   c. Set the appropriate return code in general-purpose register 15.
   d. Return to the message command processor.
3. If general-purpose register 15 is zero, the message is sent to the receiver.
4. If general-purpose register 15 is nonzero, the message is not sent to the receiver and error message 6600E is issued.

PI end

Chapter 11. Security and Integrity in z/VM

Protection against attempts to breach the security of your system and against inadvertently compromising the integrity of your system and data should be part of the planning and administration for your z/VM system. z/VM has built-in facilities and support for products to aid you in this task. This chapter gives you some points to consider and recommendations to follow that will help you to improve your system security and integrity. In particular, this chapter discusses:

- IBM software products available to enhance the security and integrity of your z/VM system
- Points and recommendations to consider in enhancing the security and integrity of your z/VM system
- z/VM facilities available for detecting and foiling attempts to break system security
- Considerations for maintaining system integrity.

Note that these recommendations are optional and whether you follow them depends on the level of security that your installation requires.

Security-Enhancing Products

The following IBM software products supported by z/VM can be used to enhance the security and integrity of your system:

- Resource Access Control Facility (RACF®) for z/VM:

  RACF is offered as an optional feature of z/VM. RACF can:

  – Help your installation implement its security policy
  – Identify and authenticate each user
  – Control each user's access to sensitive data
  – Log and report events that are relevant to the system's security

  For more information, see *z/VM: RACF Security Server General User's Guide*.
- Directory Maintenance Facility (DirMaint)

  DirMaint is offered as an optional feature of z/VM. DirMaint provides a safe, efficient, and interactive way to maintain the z/VM user directory. You can manage the directory with DirMaint through the use of its commands. Thus, with DirMaint, you avert errors that are often made during direct updating of the directory source file. You can also use DirMaint to audit security of relevant tasks that it performs.

  For more information, see *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*.
- Interactive System Productivity Facility (ISPF) for VM

  ISPF is an IBM licensed program that manages interactive applications and provides services to them. It does this by controlling the interactions of a dialog. In z/VM, it can be used to manage DirMaint and RACF dialogs.

Security Considerations and Guidelines

To enhance the security and integrity of your z/VM system, consider the following points and recommendations:

- Limit access to READ/WRITE saved segments.
- Between the time you initialize z/VM and the time you initialize RACF/VM, there is a lapse in security. In particular, the OPERATOR and AUTOLOG1 virtual machines run totally unrestrained and without audit. To prevent anyone from taking advantage of this, take the following precautions:

  – Initialize RACF/VM immediately after you initialize z/VM.

  - Make sure that AUTOLOG1 enables only the RACF service virtual machine and no other. Any other virtual machine that must be autologged can be enabled by AUTOLOG2 (for example, the DATAMOVE virtual machine, which is part of DirMaint).
  - Do not enable general-use console terminals until after you install and initialize RACF/VM. Enable only the system console and keep it under strict physical security.
  - Do not perform any major tasks until you install and initialize RACF/VM.
- Use the XAUTOLOG function to log on the DATAMOVE virtual machine. When a minidisk is detached by its owner, it still contains data the owner placed on it. Before you reallocate the same minidisk space to another user, this residual data should be removed. The DATAMOVE virtual machine automates this process.

  Make sure that the DATAMOVE virtual machine is logged on whenever z/VM is initialized. Place the CP XAUTOLOG command for DATAMOVE in the PROFILE EXEC of the AUTOLOG2 virtual machine.
- Limit and audit access to the system and logo configuration files.
- Limit and audit access to the parm disk and all CP-accessed disks. Use of these disks should be limited to system administrators or system programmers. If the minidisk is simply integrated into the administrator's or programmer's virtual machine, no auditing is possible. One way to audit the disks is to place them in a virtual machine whose password is NOLOG. This forces the administrator to use the CP LINK command—an auditable and controllable event—to access the disks. You can also restrict access and control auditing using an external security manager, like RACF.
- Format any space before converting it to minidisk space. If you convert DASD space (such as spool space, temporary disk space, or paging space) to permanent minidisk space, you should clear the entire space of all data. This prevents a new owner of the space from seeing any residual data left by the former owner. Use the CPFMTXA command to do this.
- Avoid DIAGNOSE code X'98'. DIAGNOSE code X'98' lets an authorized virtual machine lock and unlock virtual pages in storage. It also lets the virtual machine execute its own real channel programs. Thus, the virtual machine could inspect and alter CP or some other virtual machine. Because DIAGNOSE code X'98' is such a powerful function, you should take the following precautions:
  - Audit the use of DIAGNOSE X'98'
  - Remove the DIAG98 operand from the OPTION statement of any virtual machine definition unless it is really necessary for the virtual machine to use this function.
- Limit and audit the use of DIAGNOSE code X'08'. DIAGNOSE code X'08' lets an authorized virtual machine issue a CP command from a program. In some cases, multiple CP commands can be imbedded in what appears to be a single CP command when the commands are separated by X'15' characters. You should take the following precautions:
  - Add the D8ONECMD directory statement to the virtual machine definitions for server virtual machines. With this statement, any time a server virtual machine issues a single CP command that contains imbedded CP commands, the activity can be logged, rejected, or no action taken (the default).
  - Audit the use of DIAGNOSE X'08' using the SET D8ONECMD command. This auditing, however, can result in many audit records and degrade system performance.
- Avoid directly accessing the z/VM user directory. A user who has acquired the authorization can directly access the user directory source file and modify it. Without the DirMaint feature, or equivalent directory manager product, this is the only way to maintain the directory. Manually updating the z/VM user directory could compromise system security and integrity.
- When using both the DirMaint and the RACF/VM features, see the instructions in Step 5. Select RACF-Specific Characteristics of *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*.
- Limit DirMaint privileged users. Strictly limit the number of user IDs you list in the AUTHFOR CONTROL file. The users you list here can enter DirMaint privileged commands.
- Limit access to the DIRMAINT TRANSLOG file. The DIRMAINT TRANSLOG file, also known as the DirMaint history file, is a time-stamped, sequential listing of DirMaint commands entered against the z/VM user directory and the results of certain other DirMaint procedures. This file contains an audit

record of every significant change made to the z/VM user directory and to the DIRMAINT virtual machine. Thus, it is a very sensitive file.

Also, a system administrator can record a message or note in the history file using the DIRMaint LOGMSG command. This can be quite useful, but, in the wrong hands, it could introduce erroneous or deliberately misleading information into the history file.

To limit the number of users with access to the DIRMAINT TRANSLOG file, limit the number of user IDs you authorize for use of command sets A, D, H, or S in the AUTHFOR CONTROL file. By default, users authorized for these command sets can enter the DIRMaint LOGMSG command.

Note: Each use of the DIRMaint LOGMSG command appears in the DirMaint audit trail.

- Restrict the use of the single console image facility (SCIF). Also, restrict the use of the CP SET SECUSER command. See *z/VM: CP Commands and Utilities Reference* for information about the CP SET SECUSER command. This facility allows a virtual machine to control several other disconnected virtual machines simultaneously.
- Consider disabling the use of the DIAL and MESSAGE commands before logon. RACF/VM can be used to disable these commands. See the *RACF Command Language Reference* book for more information. When DIAL and MESSAGE have been disabled before logon, any user attempting to use them before logon will receive an error message. Some things to consider when deciding whether or not to disable these commands before logon are your site's use of DIAL to get to second level systems and the amount of accountability your site desires for the sending of messages.
- Restrict and audit full-pack minidisks. Full-pack minidisks that overlay other individual minidisks should be given to system administrators only. The use of full-pack minidisk should be one of the audited privileges of an administrator. However, if the minidisk is integrated into the administrator's virtual machine, no auditing is possible.

  One way to audit a full-pack minidisk is to place it in a virtual machine with a password of NOLOG. This forces the administrator to use the LINK command – an auditable and controllable event – to access the minidisk.

  You should also audit DIAGNOSE code X'E4', the full-pack minidisk overlay function. If you have RACF installed, the resource name DIAG0E4 must be identified to RACF to authorize the protected functions of DIAGNOSE code X'E4'. Users can request information about their own minidisks without RACF authorization.

  Full-pack minidisks can also be created by using the DEFINE MDISK command. If you have RACF installed, the DEFINE.MDISK profile in the VMCMD class can be used to restrict and audit the DEFINE MDISK command.
- Clear all tapes and DASD areas before reassignment. You should set rules that govern the reassignment of tapes and DASD space. For example, minidisks and temporary disks should be reformatted and tapes erased. (Note that temporary disks can be cleared automatically—see “Clearing Temporary Disk Space” on page 381.)
- Audit the MODIFY commands. The MODIFY commands let you reredefine privilege classes. The privilege class structure defines the subset of CP commands and DIAGNOSE code functions that each user is allowed to enter. Installations may need to modify the privilege class system to accommodate local needs. The ability to modify the privilege class structure is important and highly sensitive.
- Limit and audit the use of the SET PRIVCLASS command. The SET PRIVCLASS command lets you control future SET PRIVCLASS commands or to temporarily change the set of privilege classes for a logged-on user. The ability to dynamically change privilege classes is important and highly sensitive, therefore, you should limit the number of people that you allow to issue this command. This command can be issued only if the FEATURES ENABLE SET_PRIVCLASS statement has been specified in the system configuration file.
- Restrict the system operator. Certain restrictions on the system operator can increase system security; for example:
  - Ensure a high level of physical security over the IPL system console for the processor. Also, use the programmable operator facility to control what the system operator can do at this console and what the system does for the system operator. The programmable operator facility can increase

the efficiency and security of your operations by intercepting all messages and requests directed to the system operator and by handling them according to the rules you have established. For more information on the programmable operator facility, see *z/VM: CMS Planning and Administration*.
  - Make sure the system operator is aware that DASD areas and tapes must be cleared before they are assigned to new owners.

- With the ability to define the command access to suit your installation's security and system integrity requirements, you have great flexibility and control over each user's access to CP commands. You can use this to enhance security and system integrity at your installation by restricting access to system resources and information controlled by commands, DIAGNOSE codes, or system functions. The CP configuration statement USER_DEFAULTS with the POSIXOPT of querying the database has a default of ALLOW. If you do not want to allow others to query the database you will need to change this setting. However, when you change the privilege class of commands and make changes to user access, be careful not to inadvertently compromise security or system integrity by allowing users to use commands that could provide access to unauthorized information or that could affect system operation.

z/VM Security Facilities

z/VM facilities for detecting and foiling attempts to break system security and for detecting and preventing integrity exposures are:

- An interface which allows installation of an external security manager (ESM) to perform the following functions:
  - Auditing of commands, diagnose codes, and security-relevant system functions
  - Protection of selected commands, diagnose codes, and system function
  - Disabling the DIAL and MESSAGE command before logon.
  - Authorizing access to POSIX database and the information it contains
- Storage access verification functions.
- The ability to provide automatic clearing of temporary disks after use to prevent the current user from accessing data left from a previous user of the temporary disk, with the ENABLE CLEAR_TDISK operand on the FEATURES system configuration file statement.
- The ability to bypass directory password authorization.
- Journaling that allows you to monitor, record, and act on possible attempts to gain unauthorized access to system resources.
- Automatic deactivation of passwords that keeps restricted passwords from being assembled in the object directory.
- The ability to grant a virtual machine authority to restrict temporarily link access to minidisks by other users.
- The ability to suppress the display of passwords entered as part of the LOGON and LINK commands.
- The ability for multiple virtual machines to gain cryptographic capability through access to the Crypto Express optional hardware feature and the Central Processor Assist for Cryptographic Function (CPACF), which is included on the server.

In addition to the above facilities, the CMS shared file system includes various kinds of authority checking. Users can, for example, grant and revoke authority on their files and directories. When a user tries to use a file or directory, the shared file system checks whether the user is authorized to do so. More information about shared file system security can be found in *z/VM: CMS File Pool Planning, Administration, and Operation*.

Using an External Security Manager for Auditing and Protecting

An external security manager (ESM) is a service virtual machine used to maintain z/VM security and integrity. An IBM application you can use for this purpose is RACF/VM. CP can call upon RACF not only to protect certain system resources but also to audit security-relevant events such as CP commands, DIAGNOSE code functions, and communication among virtual machines.

Although all events can be audited, not all events will be audited. They are audited only if you choose. Use RACF to specify which of those events, if any, you care to audit. Any audit task involves longer path lengths, substantial input and output, and heavy use of DASD. Thus, auditing tends to degrade performance of the system. For performance considerations, do not audit more events than necessary.

RACF also provides various forms of authorization control for a subset of CP commands and DIAGNOSE codes. For additional information on using RACF/VM to audit and control CP commands, see the *RACF Security Administrator's Guide* and the *RACF Auditor's Guide*.

If you choose not to use RACF but want to write your own security application, you can use this interface to provide your own auditing. For details on DIAGNOSE code X'A0', see *z/VM: CP Programming Services*.

Verifying Storage Access

Many CP functions allow a user to access the user's own virtual storage. z/VM has functions that prevent such an access from straying outside the space the user is entitled to. Whenever a user invokes one of these CP functions that checks user's authority, for example, IUCV DECLARE BUFFER or DIAGNOSE code X'A0', CP checks whether that user is entitled to access the storage area assigned to the buffer. CP prevents the requested buffer from overlaying any area in which the user has no authority. For example, the space may be protected because it is already occupied by a CMS module. If the requested buffer is in storage-protected space, the user receives a protection exception.

For additional information on the DIAGNOSE codes and the IUCV DECLARE BUFFER function, see *z/VM: CP Programming Services*.

Clearing Temporary Disk Space

In some previous VM systems, CP cleared only cylinder 0, track 0, of a temporary disk before allocating temporary disk space. As a result, data owned by the previous user was available to the current user of the temporary disk. This created a data integrity exposure. With z/VM, temporary disk space can be optionally cleared completely to avoid this integrity exposure.

To eliminate user access to sensitive data remaining on a temporary disk, use the CLEAR_TDISK operand on the FEATURES system configuration file statement to specify whether temporary disk space is to be cleared. If space is to be cleared, it is cleared at the following times:

- At IPL time
- When CP volumes that contain temporary disk space are attached to the system
- When a temporary disk is detached from a user

if the ENABLE CLEAR_TDISK operand is specified on the FEATURES system configuration file statement.

Class B system resource operators can query the status of temporary disk clearing by using QUERY TDISKCLR. See *z/VM: CP Commands and Utilities Reference* for information about the QUERY TDISKCLR responses.

Note: If I/O errors occur during clearing of the DASD cylinders, the cylinders not cleared are not marked available for temporary disk space. This prevents the allocation of TDISK space that has not been cleared.

Performance Note: Because clearing takes place asynchronously, a period of time may elapse before the actual clearing occurs. The clearing of temporary disk space happens independently of the tasks that require it. Thus, it does not specifically affect these tasks, such as IPL, except for contention for the processor and I/O. There should be no significant impact on performance. Nevertheless, the space may not be immediately available.

Permitting Bypassing of Directory Password Authorization

With z/VM, you can bypass directory password authorization to allow specially designated virtual machines to:

- Link to any other virtual machine's virtual DASD without performance of minidisk password authorization
- Use a subset of the DIAGNOSE X'84' subfunctions to update a virtual machine definition without performance of logon password authorization.

To provide one or both of these functions to a virtual machine, code either or both of the LNKNOPAS and D84NOPAS operands on the OPTION directory statement of the virtual machine definition. For additional information on these operands of the OPTION statement, see "OPTION Directory Statement" on page 572.

Journaling the LOGON, AUTOLOG, XAUTOLOG, and LINK Commands

LOGON, AUTOLOG, XAUTOLOG, and LINK journaling detects and records certain occurrences of the LOGON, AUTOLOG, XAUTOLOG, and LINK commands. Using the recorded information, you can identify attempts to log on to CP by users who enter invalid passwords. Also, you can identify any user who successfully enters the LINK command to a protected minidisk that user does not own.

Briefly, LOGON, AUTOLOG, XAUTOLOG, and LINK journaling works like this. While journaling is turned on, CP monitors all occurrences of the LOGON, AUTOLOG, XAUTOLOG, and LINK commands. CP counts how many times a user enters one of these commands with an invalid password. CP can be set to take one or more of these actions when the count reaches a threshold value:

- Write a record to the accounting data set to record the incident
- Reject subsequent LOGON, AUTOLOG, XAUTOLOG, and LINK commands entered by the user
- Lock that terminal for a designated period
- Send a message to a designated user ID to alert the installation to the incident.

While journaling is turned on, CP creates an accounting record each time it detects that a user has successfully entered a LINK command to a protected minidisk not owned by that user. A protected minidisk is a minidisk whose password is anything but ALL for the type of LINK attempted, or a minidisk protected by an external security manager.

For a description of the accounting records that CP writes for LOGON, AUTOLOG, XAUTOLOG, and LINK journaling, see Accounting Record Formats in *z/VM: CP Programming Services*.

To make LOGON, AUTOLOG, XAUTOLOG, and LINK journaling available and to specify options, you may use the JOURNALING statement in the system configuration file. For more information, see "JOURNALING Statement" on page 183. To turn journaling on or off, use the class A SET command. To determine whether journaling is on or off, use the class A or C QUERY command. For additional information on these commands, see *z/VM: CP Commands and Utilities Reference*.

Automatic Deactivation of Restricted Passwords

A facility is available to validate user logon passwords against a set of predefined restricted passwords. The facility automatically validates other user passwords to prevent the accidental use of a restricted (published) password for or by a user. It provides a file that contains a list of IBM restricted passwords. This list, called the RPWLIST DATA, is a CMS file that is included on MAINT*vrm* 2C2 (where *vrm* is the z/VM version, release, and modification level) if z/VM was loaded to minidisk. It is on `VMPSFS:MAINTvrm.CPDV.SAMPLE`, if z/VM was loaded to filepool. When you run the DIRECTXA command to convert the source directory to an object directory, passwords are checked against the passwords in this list. You do not need to perform any action to use this list.

You may add your own restricted passwords to this list or remove some of the IBM-restricted passwords from the list as the need arises. To modify the RPWLIST DATA file, you simply edit it and create a private copy to be kept with the source directory. You can find this file by using the CMS search order.

You must use the following formatting rules when you edit the RPWLIST DATA file:

- The RPWLIST DATA file must have a fixed-record-length format.
- The records must be at least 8 characters long but no longer than 80 characters.

- The passwords to be restricted must begin in the first column of each record.
- Columns 1 through 8 must contain only one restricted password and nothing else.
- There may be only one restricted password per record.
- Column 9 must be blank.
- Columns 10 through 80 may be used for comments.

If you do not want automatic deactivation of passwords, edit the RPWLIST DATA file and delete all the passwords. Leave one dummy record of an asterisk (*) in column 1.

Using Link Access Control Options

You can set up a service virtual machine with the authority to control user access to minidisks and databases. This authority is useful when running database applications and performing data migration tasks where data stability and integrity are essential. To do this, the service machine is configured with directory statement options that allow it to use certain link access modes that temporarily restrict other users from gaining link access to the data the service machine is using.

Two types of access modes can be used to ensure data integrity when linking to another minidisk for a limited period of time. The stable access modes, when requested by an authorized virtual machine, prevent other users from obtaining write access to a disk while the stable access is in effect. The exclusive access modes grant an authorized virtual machine sole access to a minidisk, preventing both read and write access by any other users.

The LNKSTABL and LNKEXCLU operands of the OPTION directory statement authorize a user to use the stable or exclusive access modes of the LINK command or DIAGNOSE X'E4'. This global authority allows a virtual machine to perform a stable link to any minidisk for which it has password level authorization. You can also specify stable and exclusive authority to a specific minidisk using the mode suffix letter (S or E) on the MDISK or LINK directory statements. While stable or exclusive links to a minidisk are in effect, access to that minidisk by any other user will be restricted or denied. For more information on the LNKSTABL and LNKEXCLU options of the OPTION directory statement, see "OPTION Directory Statement" on page 572. See "LINK Directory Statement" on page 533 and "MDISK Directory Statement" on page 550 for more information about access modes and suffixes.

Suppressing Passwords Entered on the Command Line

CP can be set to reject LOGON or LINK commands that have the password entered on the same line as the command. Rejecting these commands prevents passwords from being displayed or printed without masking. (Masking a password means overprinting the password so it cannot be read.)

This capability is also available to virtual machines that issue LINK commands by DIAGNOSE code X'08'. For a description of DIAGNOSE code X'08', see *z/VM: CP Programming Services*.

To request password suppression, specify it as an option on the FEATURES statement in the system configuration file. Once requested, password suppression is always on; an operator cannot turn it off. See "FEATURES Statement" on page 158 for information on how to use the FEATURES PASWORDS_ON_CMDS operand.

Cryptographic Acceleration

The Crypto Express features of IBM Z® servers are designed to satisfy high-end server security requirements. They can be configured as coprocessors for secure key transactions or as accelerators for Secure Sockets Layer (SSL) acceleration, providing significant improvements in the performance of cryptographic algorithms used for encryption and public-private keypair generation and verification. z/VM makes Crypto Express available to guests with either dedicated access for use for both secure-key and clear-key operations, or with shared access for clear-key operations. Information on making Crypto Express available to a virtual machine can be found in the description of the APVIRTUAL and APDEDICATED operands in the "CRYPTO Directory Statement" on page 490.

The CP Assist for Cryptographic Function (CPACF) is a part of each processor in the IBM Z server. It provides a set of cryptographic functions that focuses on the encryption/decryption function of SSL, Virtual Private Network (VPN), and data-storing applications. The CPACF is used by SSL/TLS functions included in the z/VM Lightweight Directory Access Protocol (LDAP) client and server, and by the SSL functions provided by the z/VM SSL server. Any virtual machine can access the functions of the CPACF by using the Message-Security Assist (MSA) extensions of the Z processor architecture. No explicit z/VM authorization or configuration is required. Information on MSA instructions can be found in z/Architecture Principles of Operation (https://publibfp.dhe.ibm.com/epubs/pdf/a227832d.pdf).

For more information on the specific capabilities of Crypto Express and CPACF, consult the documentation for your processor.

Pervasive Encryption for z/VM

Beginning with the IBM z14, the z/VM hypervisor can exploit host-level encryption. The z/VM Control Program can use programming interfaces to encrypt and decrypt data. This capability provides broad encryption for data in the hypervisor layer. It represents host-level cryptography and may not be a replacement for encryption services at the hardware or guest level. The paging subsystem exploits this new capability. Encryption of data sent to and/or from paging volumes can be enabled in the hypervisor layer. All data ciphered in this way uses a common ephemeral key, which lasts until the next system IPL. Since paging data is not meant to persist past an IPL, these keys are discarded at system termination. For more information about pervasive encryption, see “ENCRYPT Statement” on page 152.

Guest Secure IPL

The z/VM hypervisor can exploit hardware-secure boot to validate a guest's boot code. A guest can ask the machine to validate the guest's boot code, which must be signed by the customer or the code's supplier. The machine loader validates that the boot code matches one of the keys that the customer loaded into the HMC certificate store.

For more information, see the following topics:

- Guest Secure IPL in *z/VM: Running Guest Operating Systems*
- IPL Directory Statement in *z/VM: CP Planning and Administration*
- IPL in *z/VM: CP Commands and Utilities Reference*

Maintaining System Integrity

Your z/VM system has integrity if it can prevent the circumvention, subversion, and disabling of its security mechanisms. In general, this is achieved by keeping users separate from each other, separate from the operating system, and limiting their access only to data which they need to do their jobs.

Quite simply, system integrity is your system's ability to:

- Resist compromise of its security controls through misuse and manipulation
- Ensure that its resources can only be accessed through authorized routes by authorized users.

To put it another way, system integrity is the inability of any program running in your system to:

- Obtain control in real supervisor state, with privilege class authority or directory capabilities greater than assigned
- Use CP to circumvent the system integrity of any guest operating system which itself has system integrity
- Circumvent z/VM real storage protection or auxiliary storage protection
- Access, without authority, a z/VM password-protected resource
- Access, without authority, a resource protected by an external security manager (ESM), like RACF.

The IBM z/Architecture®, upon which z/VM is based, is at the center of the system's ability to maintain integrity. One crucial aspect of this is your system's ability to keep each virtual machine absolutely isolated from every other virtual machine. This isolation especially extends to CP, which is logically separate from all virtual machines in the system.

For the sake of integrity, z/VM exploits the z/Architecture in several other ways:

- The addresses in a virtual machine are virtual addresses. They have no meaning outside the virtual machine in which they are generated and used. Whenever required, these virtual addresses are translated into real addresses by ART (access register translation) and DAT (dynamic address translation), for the address space referenced by the user. Using ART and DAT, the system keeps these address spaces absolutely separate from one another. This means that it is impossible for one user to access an address space of another user unless the owner allows the other user to do so. Additional information about accessing data spaces can be found in *z/VM: CMS Application Development Guide*.
- z/VM translates the addresses in all channel programs, except those initiated by DIAGNOSE X'98'. Channel programs are programs built and run by virtual machines that request auxiliary storage devices to perform input and output tasks. z/VM identifies the storage device and performs the I/O operation on behalf of the virtual machine.
- Every z/VM virtual machine runs in interpretive-execution mode which processes most privileged and non-privileged instructions and handles virtual storage address translation without requiring intervention of z/VM.
- z/VM uses page protection to prevent read-only saved segments from being modified. A saved segment is a block of data or reentrant code in virtual, shared storage that many users can share simultaneously. However, if a user has a legitimate reason for wanting to change a read-only saved segment, the user must specifically request an exclusive copy of the saved segment and be authorized to do so in the user directory. The unmodified code remains shared among the other virtual machines. See "Protection of Shared Storage" under "Storage Handling" on page 391 for more information.

z/VM System Integrity Requirements

There are many things z/VM provides to support system integrity. You should be aware of them and make certain that all programs and users cooperate with them.

System Modification Requirements:

A program added to the z/VM system does not weaken the latter's integrity as long as it:

- Uses only documented, unrestricted z/VM interfaces
- Runs only with privilege class G authority
- Does not supply services to more than one user from within a single virtual machine
- Does not require the use of special CP user directory options
- Does not share VM or other passwords with any other program running in any virtual machine.

Your installation probably will add several programs to the system to fulfill local needs. Because the integrity of z/VM can be affected by these, consider the following questions before you add any to your system:

Does the program (or any of its parts):

- Run with a user ID which, if an ESM is installed, requires special privileges?
- Require the use of CP DIAGNOSE functions that are restricted to classes other than G or ANY?
- Require CP directory options?
- Modify z/VM code?
- Modify z/VM by adding or changing a z/VM command?
- Run as a multiuser service machine?

If the answer to any of these questions is yes, the program could seriously weaken the integrity of z/VM. Carefully reconsider whether you should add the program to your system. If you must add a program that threatens to weaken your system's integrity, keep the following in mind:

- z/VM maintains storage protection in one of two ways, depending upon the nature of the object:
  - Access register translation (ART) and dynamic address translation (DAT)

    protect the storage of nonshared segments; that is, storage that is reserved exclusively for one user. ART and DAT are hardware facilities used by z/VM during the execution of any instruction to translate a virtual address into the corresponding real address. The system uses ART and DAT to provide secure, separate address spaces for each virtual machine in the system. This means that it is impossible for one user to access an address space of another user unless its owner allows the other user to do so. Additional information about accessing data spaces can be found in *z/VM: CMS Application Development Guide*.
  - z/VM's shared segment protection mechanism gives any user that tries to alter a read-only saved segment a protection exception. This preserves the integrity of data or code being shared simultaneously by other users.
- You may wish to use an ESM to protect sensitive system data and work areas from access by unauthorized users and to protect any proprietary information automatically stored on external media. Such protection requires several things of the user:
  - The user must protect proprietary information from all other users except those properly authorized. Proprietary information includes passwords, cryptographic keys, user register contents, user data areas, and buffers.
  - The user must clear all proprietary data from any storage device or buffer before it is released. Temporary disks (T-disks) are the exception. If ENABLE CLEAR_TDISK is specified on the FEATURES system configuration file statement, z/VM clears each T-disk before it is assigned to anyone.
  - The user must not use z/VM commands and DIAGNOSE codes that bypass (or allow the bypassing of) security checks, integrity controls, or validation procedures. That is, limit the use of commands and functions that allow direct and uncontrolled access to basic system resources. (See "Sensitive z/VM Commands to Restrict" and "Sensitive DIAGNOSE Functions to Restrict" under "z/VM Integrity and CP Function" on page 387.).) Use z/VM privilege classes, DIRMAINT, or an ESM to control access to these resources.

User Identification Requirements: To identify a user means to firmly establish who is using the system to perform a particular act. Every command, DIAGNOSE, and other security-relevant event must be directly attributable to a user whose identity has been well-established. With POSIX, however, you can have multiple user IDs associated with a single UID. Authorities can be given to the one UID and therefore multiple user IDs. Determining the identity becomes difficult. For this reason, it is not recommended that you have multiple user IDs associated with a single UID. User identification requires the following:

- System and user resources must be separated from one another and identified. Otherwise, system resources may be counterfeited or one system resource may be substituted for another.
- Before a privileged program passes user data to another privileged program, it is necessary to define who is responsible for validating the user data. Then, of course, the data must be validated.

Validation Requirements: Validation is an important term that can mean several things:

- z/VM routines that access areas of storage based on user-supplied addresses must first validate that the user has the appropriate kind of access (READ, WRITE, and so forth).
- During the validating process, the system can use only previously validated and protected data. Using non-validated or unprotected data during the validation process invalidates the process itself.
- When a user wants to access an area that spans a page boundary, the system must confirm that the user has authority to access the entire range of addresses involved.
- z/VM routines that simulate CPU instructions or translate channel programs must do so according to the architecture of the CPU or device involved.
- Parameters passed to the system must be validated for the purpose intended.

Serialization Requirements: Serialization is a method used to prevent the asynchronous altering of variables, whose validity must be checked, until after the operation for which they are validated is complete.

A gap can occur between the time when a variable is checked and approved and the time when the variable is actually used in an operation. During this time, the variable is vulnerable to change by some unauthorized, asynchronous function. The solution is to protect each validated variable from asynchronous alteration until it is used.

Note: z/VM system integrity does not specifically include the protection of data among several users of a single CMS batch system nor does it apply to virtual machines using the nondisruptive transition (NDT) support.

z/VM Integrity and CP Function

There are several CP resources that can affect system integrity:

Privilege Classes: A privilege class is a subset of z/VM commands and DIAGNOSE codes. Each user is assigned to a particular privilege class, depending upon his responsibilities, level of skill, and place in the organization's hierarchy. Every user is a member of at least one privilege class.

If a user attempts to issue a command that is outside his privilege class, the system ignores the command. The main benefit of this is that no user can alter the system in any way that goes beyond his expertise and authority.

IBM designed the structure of z/VM privilege classes with the typical organizational hierarchy at the typical computing installation in mind. If you find the IBM privilege class structure inappropriate to your organization's needs, you can modify it or completely replace it. It is possible for your organization to precisely define up to 32 privilege classes of its own that partly or completely redefine the privilege class structure that comes with your system. For more information, see Chapter 19, “Redefining Command Privilege Classes,” on page 449.

Sensitive z/VM Commands to Restrict: Very few CP commands and DIAGNOSE functions have no security and integrity relevance. Some affect the security and integrity of your system more than others, and there are some CP commands and DIAGNOSE functions that allow direct, uncontrolled access to basic system resources.

Change the CP privilege classes or use an external security manager to limit the use of the following sensitive CP commands:

CPACCESS
: identifies a CMS-formatted minidisk to CP and makes the files on that minidisk available to CP by establishing a file mode letter for the files.

CPCACHE
: causes CP to cache a file on a CP-accessed minidisk.

CPRELEASE
: releases a CP-accessed minidisk.

DEFINE CPOWNED
: lets a user define new entries or to change existing entries in the list of CP-owned DASD volumes.

DEFINE TIMEZONE
: lets a user define a new time zone or change an existing time zone definition.

DISPLAY (Host Storage)
: displays the contents of host storage at the user's terminal.

DUMP (Host Storage)
: prints the contents of host storage at the spooled virtual printer.

LOCATE
: determines the address of a particular user's CP control block, the address of a virtual device, or the address of a real device.

SET D8ONECMD
: lets a user change the D8ONECMD settings for their or other's virtual machine. The D8ONECMD settings control whether CP will accept multiple commands imbedded in a single command and separated by X'15' characters.

SET MDCACHE SYSTEM
turns minidisk caching on and off for the whole system. Turning minidisk caching off at the system level disables caching enabled at the device or record level.

SET OBSERVER
changes the observer user ID associated with your virtual machine or another user's virtual machine.

SET PRIVCLASS
controls future SET PRIVCLASS commands or temporarily changes the set of privilege classes for a logged-on user.

SET RDEVICE
changes or adds devices to the system's definition of a set of real devices.

SET SECUSER
changes the secondary user ID associated with your virtual machine or another user's virtual machine.

SET SYSOPER
changes the user ID of the primary system operator on your system.

SET TIMEZONE
changes the system's active time zone ID and time zone offset.

SET VDISK SYSLIM
limits the total resource available for allocating virtual disks in storage. Users may suddenly find that they cannot create as many virtual disks in storage as they had been able to.

SHUTDOWN
systematically ends all system functions and checkpoints the system for an eventual warmstart. May also perform an automatic warmstart of the nucleus.

SNAPDUMP
takes a dump identical to a hard abend dump, without shutting down the system. Could stop the system long enough to cause communication lines to be dropped.

STORE (Host Storage)
allows the user to alter the contents of host storage.

Sensitive DIAGNOSE Functions to Restrict: Change the CP system of privilege or use an external security manager to limit the use of the following sensitive DIAGNOSE functions:

DIAGNOSE X'04'
allows the user to examine host storage.

DIAGNOSE X'08'
allows a virtual machine running in supervisor state to issue CP commands. Usually, this presents no danger to your system's security or integrity. Often, z/VM commands are issued on behalf of authorized users by privileged server virtual machines. However, if a server virtual machine were to issue a CP command on behalf of an unauthorized virtual machine, your system's security and integrity would be threatened. Each server should check carefully the identity and authorization of each machine for whom it does work, like DIRMAINT does.

DIAGNOSE X'28'
allows channel programs modified after STARTIO (but before the I/O operation is complete) to execute correctly.

DIAGNOSE X'3C'
allows the user to dynamically update the CP user directory.

DIAGNOSE X'4C'
allows a user with the ACCT option in his virtual machine definition to generate accounting records.

DIAGNOSE X'7C'
allows a program to drive a logical 3270 terminal as though it were a real, locally attached 3270.

DIAGNOSE X'84'
allows a user to replace certain data in the CP user directory.

DIAGNOSE X'98'
allows a virtual machine authorized to use it to lock and unlock virtual pages in storage. It also allows the virtual machine to execute its own real channel programs.

Data Structures That Can Enhance System Integrity

z/VM maintains many directories, tables, and control blocks that manage the system. Many of these data structures contribute to your system's security and integrity.

System Configuration File: The system configuration file contains definitions for your system and how it should operate. The items in this file that pertain to security and integrity include:

- FEATURES ENABLE CLEAR_TDISK statement, specifying that all previously written data and directory areas on temporary disk DASD space should be cleared automatically.
- FEATURES PASSWORDS_ON_CMDS statement, specifying whether the facility that suppresses the password on the command line should initially be part of the system.
- JOURNALING statement, specifying whether the facility that journals events should be part of the system, whether the system can set and query the journaling facility, and what to do if someone tries to log on to the system or link to a disk without a valid password.

CP User Directory: The CP user directory is a table maintained on DASD, which contains virtual machine definitions that describe the configuration of particular virtual machines in your system. To put it another way, the CP user directory tells z/VM exactly which system resources are available to each virtual machine, how each is built, and how access to each is governed.

The items in each virtual machine definition that pertain to security and integrity include:

- CP log on passwords
- Minidisk passwords
- LINK statements
- CLASS statement, describing what kind of user this is and what class of commands and instructions the user will be allowed to issue
- DIAG98 option on the OPTION directory statement, letting the user invoke the DIAGNOSE X'98' function.
- LNKNOPAS and D84PAS options on the OPTION directory statement.
- POSIXINFO statement for the UID and GID|GNAME
- POSIXGLIST statement for listing the groups that the user belongs to

RSCS Configuration File Options: The RSCS configuration file defines the RSCS network, and includes such information as:

- Local node identifier
- Identifier of each node with which the local node can communicate
- Network links over which the local node communicates with the remote nodes.

VM/Pass-Through Directories: VM Pass-Through (PVM) allows a user to interactively access a virtual machine in some remote system. Your system's PVM virtual machine maintains a directory called a configuration file. This file includes:

- Local and remote node identifiers
- Link and routing definitions
- Authorization of certain users to issue restricted commands.

Directory Update-in-Place: DIAGNOSE X'84' gives a privilege class B user the ability to replace data in the CP user directory. Note that this doesn't authorize the user to add new entries or to delete existing ones.

As z/VM is delivered, all privilege class B users can issue DIAGNOSE X'84'. For the sake of security, strictly limit the number of privilege class B users in your system. Of course, your organization may have chosen to change the privilege class structure that IBM has built into your system. If so, be certain to limit the number of users who can issue DIAGNOSE X'84'.

Dumps: A dump is a snapshot, taken at a particular moment, of the contents of a computer's storage. Most dumps are created by highly privileged users to help them solve system problems.

There is no way to predict what storage contains when the dump is created. Passwords or poorly encrypted data may be present, or the dump may contain other material that your organization considers proprietary and confidential. Therefore, take great care that only authorized personnel handle the dumps generated by the system. What's more, never send a dump to anyone outside your organization unless you are certain it contains nothing proprietary or confidential.

Composite Reader File: The composite reader file is a spool file that contains a group of CMS files. From the CMS point of view, the composite reader file is many files; from the z/VM point of view, however, it is all one spool file. This difference in perspective can create security and integrity problems.

To illustrate, consider the following sequence of commands issued by a user named MARY to create a composite reader file in the virtual reader of a user named JOHN.

```
CP SPOOL PUNCH JOHN CONT
PUNCH PROFILE EXEC
PUNCH PROFILE XEDIT
PUNCH CALENDAR DATA
⋮
PUNCH NOVEMBER REPORT
CP SPOOL PUNCH CLOSE NOCONT
```

If JOHN issues a CP QUERY READER command, the response indicates that there is only one file in his reader, NOVEMBER REPORT. But because this is a composite spool file, there are several other files present that are hidden from any QUERY. The presence of these other files is neither implicit nor apparent.

Now, assume that JOHN already has a file called PROFILE EXEC. When he reads in NOVEMBER REPORT, all the other files are read in, too. JOHN's PROFILE EXEC disappears, overwritten by the new PROFILE EXEC. JOHN may or may not consider that desirable. At the very least, he should be informed of the loss of his file.

To prevent security and integrity problems like this from arising, z/VM provides several options to the DISK LOAD, READCARD, RECEIVE, and DEFAULTS commands:

FULLPROMPT
: Specifies that a prompt message is issued for each file.

MINPROMPT
: Specifies that a prompt message is issued when the name of the first file differs from the name of the spool file. The prompt message for the first file does not appear when it has the same name as the spool file.

NOPROMPT
: Specifies that a prompt is not issued when a file is received.

REPLACE
: Specifies that if a file with the same file name and file type already exists, then it is to be replaced with this one.

NOREPLACE
: Specifies that no file ever overlays an existing file on the receiving disk.

z/VM Options That Can Enhance System Integrity

z/VM offers several options that can help you to enhance the integrity of your system.

Real Channel Program Support: DIAGNOSE X'98' lets an authorized virtual machine lock and unlock virtual pages in storage. It also allows the virtual machine to execute its own real channel programs. That would make it possible for the virtual machine to access real storage beyond the legitimate end of its address space. It would then be possible for the virtual machine to inspect and alter CP or some other virtual machine. That would not be sound security practice.

The ability to invoke this function is an extremely powerful privilege, which the system administrator should either restrict or eliminate.

There are two ways to prevent the use of DIAGNOSE X'98':

- Remove DIAG98 from the OPTION statement of each virtual machine definition in the CP user directory in which it appears.
- Set DIAGNOSE X'98' to an unused privilege class.

If an unauthorized user attempts to use DIAGNOSE X'98', he receives an operation exception.

Storage Handling

Your z/VM system maintains the security and integrity of its storage, as follows:

Protection of Host Storage: z/VM provides both fetch and store protection for host storage. No user may store in or fetch from a segment of host storage unless the user's key matches the protection key of that storage.

z/VM protects host storage using the privilege class structure. As the system is delivered, only privilege class C users can alter host storage. Of course, your organization may have chosen to change the privilege class structure that IBM has built into your system.

Allocation of Host Storage: Host storage is allocated to a user when the user first refers to a virtual page for which a backing frame has not yet been allocated. Allocation continues, as needed, up to and including the maximum storage size indicated in the user's virtual machine definition.

Before z/VM allocates a frame of real storage, that frame is cleared of any data that may have been left behind by another user. This prevents a user from having access to data for which that user has no authorization. z/VM maintains a frame table that is used to identify the usage state of every frame in host real storage.

Paging: In z/VM, many users share host storage simultaneously. But not every page of storage could possibly be active every second of the time. When a resident page of guest storage is not being actively accessed by a user, the system transfers its contents to DASD. This is known as paging out. The space in host real storage that it abandons then becomes available to another user who really needs it. When and if that same guest page is needed again, the system transfers the contents from DASD back to host real storage. This is known as paging in.

z/VM maintains a set of page and segment tables that contain precise information on the location of every page and segment of guest storage, whether paged in or paged out. The system follows them rigorously, helping to maintain the integrity of the system.

z/VM also allows for guest page data to be encrypted when written to a paging disk. Encryption prevents inadvertent disclosure of guest data to system administrators who have access to these volumes. Note that you should exercise caution when enabling encryption in CP. For more information, see "Precautions for Using the ENCRYPT PAGING REQUIRED Option" on page 711.

Protection of Shared Storage: Virtual storage space is routinely shared by users of z/VM. These shared segments help the system manage its storage more efficiently.

Shared segments can be read/only or read/write blocks of storage containing code or data shared by many users. Sharing one copy of something, like XEDIT, is preferable to giving each user his own copy.

As z/VM is delivered, only a user with privilege class E can define and save a shared segment. Of course, your organization may have chosen to change the privilege class structure that IBM has built into your system. If so, take care that the ability to define and save shared segments is not given out indiscriminately.

Clearing Residual Data from Minidisks: You can tailor DIRMAINT to automatically clear DASD space whenever a minidisk is detached. This applies when you release an entire minidisk and when you permanently reduce the size of a minidisk. This prevents the next user to whom the space is allocated from seeing information he may not be authorized to see.

Clearing Residual Data from Tapes: Sometimes, a user finds that he no longer needs the data on a reel or cartridge of tape. This does not mean that the data is no longer sensitive. Hence, you must establish procedures to ensure that each reel or cartridge of tape is degaussed before it is assigned to a new user.

Program Stack, Security, and Integrity

While executing, many programs and EXECs place data on the program stack. Ideally, this data should be used to accomplish useful work and then promptly discarded. In reality, though, it may be left behind in the stack after the program terminates. In fact, many programs intentionally leave data on the program stack so that they can pass information to other routines. While this practice may be clever, it is not risk free.

As a rule, when a program or EXEC terminates or relinquishes control, the program stack should be exactly as it was when the program or EXEC gained control. This should apply to both normal and abnormal termination. The program or EXEC that terminates before restoring the program stack can cause serious security and integrity problems.

For example, suppose a privileged server virtual machine runs a program that obtains data from a virtual reader or other communication channel and places it in the program stack. An unscrupulous party comes along and feeds his own commands to the program through the communication channel. Dutifully, the program places them on the program stack until the attacker somehow forces the program to fail. If the program is designed to restore the program stack, no harm is done. However, if the program does not restore the stack, then the attacker's commands are executed by the privileged server machine!

Many programmers prudently avoid using the program stack altogether by using GLOBALV variables to pass information to other routines. Some programmers use other techniques, such as managing the program stack with the CMS MAKEBUF and DROPBUF commands and the StackBufferCreate and StackBufferDelete CSL routines. These commands and routines enable a user to manage a new, separate buffer that can be used without affecting the contents of any buffers previously defined.

Study the following examples.

A Simple EXEC that Manages the Stack with MAKEBUF and DROPBUF

```
ADDRESS COMMAND
SIGNAL ON HALT          /* FORCE ANY INTERRUPTION TO COME OUT
                           OF MAIN EXIT */
'MAKEBUF'               /* GET A BUFFER TO HOLD ANY GARBAGE */
BUFNO = RC              /* KEEP ITS NUMBER */
⋮
(MAIN PROCESSING)
⋮
HALT:                   /* MAIN EXIT PROM PROGRAM */
SRC = RC                /* KEEP LAST "RC" FOR EXIT */
'DROPBUF' BUFNO         /* DISCARD GARBAGE */
EXIT SRC
```

A More Complex EXEC Calling a Subroutine

```
ADDRESS COMMAND
SIGNAL ON HALT          /* FORCE ANY INTERRUPTION TO COME OUT
                           OF MAIN EXIT */
'MAKEBUF'               /* GET A BUFFER TO HOLD ANY GARBAGE */
BUFNO = RC              /* KEEP ITS NUMBER */
⋮
(MAIN PROCESSING)
⋮
'MAKEBUF'               /* GET ANOTHER BUFFER TO PASS DATA
                           TO SUBROUTINE */
QUEUE 'SOMETHING FOR SUBROUTINE'
CALL SUBROUTINE
SRC = RC                /* KEEP "RC" FROM SUBROUTINE */
'DROPBUF' BUFNO + 1     /* DISCARD ANYTHING NOT CLEARED BY SUBROUTINE */

                        /* AT THIS POINT, TEST RETURN CODE FROM
                           SUBROUTINE, USING "SRC" INSTEAD OF "RC" */
⋮
(MORE PROCESSING)
⋮
HALT:                   /* MAIN EXIT FROM PROGRAM */
```

```
SRC = RC                /* KEEP LAST RC FOR EXIT */
'DROPBUF' BUFNO         /* DISCARD GARBAGE */
EXIT SRC
```

An EXEC Called as a Subroutine and Returning Data in the Program Stack

```
ADDRESS COMMAND
SIGNAL ON HALT          /* FORCE ANY INTERRUPTION TO DO
                           FINAL PROCESSING */
'MAKEBUF'               /* GET A BUFFER TO HOLD ANY GARBAGE */
BUFNO = RC              /* KEEP ITS NUMBER */
⋮
(MAIN PROCESSING)
⋮
SRC = RC                /* KEEP LAST "RC" FOR EXIT */
'DROPBUF' BUFNO         /* DISCARD GARBAGE */
QUEUE 'SOMETHING FOR CALLING ROUTINE'
EXIT SRC                /* MAIN EXIT FROM PROGRAM */
HALT:                   /* SPECIAL EXIT FROM PROGRAM IF
                           INTERRUPTED BY "HI" */
'DROPBUF' BUFNO         /* DISCARD GARBAGE */
EXIT
```

Reporting z/VM Integrity Problems

The authorized program analysis report (APAR) is the formal mechanism for reporting to IBM problems you are having with z/VM. These problems may include:

- Integrity exposures
- Logic errors in the programming
- Documentation errors
- Program distribution problems.

You should never use the APAR process to:

- Comment on or suggest improvements to z/VM
- Report minor stylistic, grammatical, spelling, or punctuation errors in a z/VM document
- Report on the packaging or quality of the items you received from IBM.

IBM also accepts security APARs involving z/VM or any product that runs on it. A security APAR reports problems in existing z/VM security mechanisms where the problem is not quite an integrity problem but does represent a threat to the security of the system as a whole or to one of its components.

If you discover an integrity problem in z/VM, proceed as follows:

1. Consider your problem in light of the discussion of system integrity under "Maintaining System Integrity" on page 384. If it is truly a system integrity problem, proceed to the next step.
2. Confirm that you have the current version, release, and modification level of each of the software products in your system.
3. Examine the appropriate documentation to see if there is a solution to your problem.
4. Remove any modifications you may have applied to z/VM to be certain that this is not the cause.
5. If your modification has caused the problem, do not replace it until it is repaired. If none of your modifications is the problem (or if you have made none), try to isolate the problem to a particular component of z/VM.
6. Gather and retain the necessary materials that document your problem: dumps, output listings, tapes, messages, return codes, and so forth. Be certain that none of this material is considered proprietary by your organization. If it is, try to replicate the problem resulting in data that is not proprietary.
7. Notify IBM service.

To resolve your problem, IBM may:

- Correct IBM code or documentation
- Document a temporary or permanent restriction

- Notify you that it is aware of the problem (IBM either provides an immediate solution or waits until the next release of the program or document)
- Notify you that someone or something at your installation is the cause of the problem (IBM then advises you on a solution).

If, to solve your problem, IBM must correct its code or documentation, the company alerts all z/VM customers to the problem and provides a program temporary fix (PTF). A PTF is a temporary solution to or bypass of a problem identified by IBM as the result of a defect in a current, unaltered release of the software product. Your system administrator is responsible for installing all PTFs; IBM provides any necessary PTF documentation.

Chapter 12. The Stand-Alone Dump Utility

CP can produce several formats of dumps: a CP dump, a snapdump, a stand-alone dump and a virtual machine dump. Dumps are especially helpful in analyzing problems such as wait states, infinite loops, and abends. This chapter describes how to create a stand-alone dump device and use it. For more information on debugging various types of problems, the other dump types, and other problem diagnosis tools, see *z/VM: Diagnosis Guide*.

The stand-alone dump utility creates a dump that contains only storage that is in use by CP. It is able to create a CP hard abend format dump which is usually much smaller than a storage dump and the dump will be written to either ECKD or SCSI DASD. It also supports larger memory sizes.

z/VM includes a CMS-based utility, SDINST, that you will use to create an IPL'able stand-alone dump DASD device that is tailored according to your installation’s configuration. After you install z/VM, you should create the stand-alone dump device for emergency use. This is usually done by the system programmer, not a general user. If, after a system failure, CP cannot create an abend dump, the operator can IPL the stand-alone dump device to dump CP storage.

This chapter describes the stand-alone dump installation utility, SDINST, and provides information about using the stand-alone dump device to create a dump.

Creating the Stand-Alone Dump Utility

Your installation can generate a stand-alone dump device customized to your system configuration. This gives you control over the DASD used to IPL the stand-alone dump program and the additional output devices for the dump. To install the stand-alone dump program, a CMS-based utility, SDINST EXEC, prompts for setup information and then formats and initializes the IPL and DUMP DASD.

To use the SDINST utility, your user ID must have the following:

- Access to the SDINST EXEC, which usually resides on MAINT 190
- At least 512 MB of virtual storage
- A virtual reader at device number 00C and a virtual punch at 00D
- No Class N reader or punch spool files
- A read/write 191 minidisk that is accessed as file mode A with at least ten 4K blocks of free space
- A read-only 400 minidisk, preferably MAINT 400, which contains the following files:
  - SADU73 IMAGE
  - SSPJ73 IMAGE
  - SSPK73 IMAGE
  - SSPP73 IMAGE

SDINST can run interactively to build its configuration file, SDINST DUMPCONF, or it can use an existing configuration file, which is useful if you are reinstalling to the same devices.

1. Upon invocation, the utility will check for an existing SDINST DUMPCONF file. If the file exists, you will be given the option of re-using that file or creating a new one. If the file is re-used, the utility verifies the information in the dump configuration file before formatting and initializing the DASD.
2. If a new file is to be created, the user is first asked for the virtual device number of a 3390 DASD or FCP device desired for the stand-alone dump IPL disk. This disk is where the stand-alone dump program will be written and will also contain the first part of the dump.
   a. The utility will then ask for the real device number of the 3390 or FCP that will be used at dump time.

   b. If the device is a 3390, the utility then asks if the volume label found is correct. The label will be useful to verify the correct disk will be formatted at install time, as well as when the dump is written at dump time. The DASD will be checked to ensure it is accessible and writeable.
   c. If the device is an FCP, the utility will continue by asking for the WWPN and LUN. These values are used at install time by the DASD formatting program and also at dump time. Only one FCP / WWPN / LUN combination is allowed per LUN.
   d. After gathering information about the IPL device, the utility asks if there are additional dump devices. If there are, the previous steps for obtaining device information are repeated. There can be a maximum of 30 dump devices, including the IPL device.
3. Once the utility has all the information, it will create the dump configuration file on the user’s 191 disk which must be accessed as file mode A. Then the utility punches the DASD formatting program files to the user's virtual reader. After the user confirms that they are ready to format the DASD, SDINST will IPL the reader and format and initialize the DASD. When it completes, the user must re-IPL CMS.

An example of the prompts and replies that appear on the virtual machine console during SDINST EXEC execution is shown in “Examples for Generating the Stand-Alone Dump Utility” on page 397.

Additional Information about Stand-Alone Dump

Mixing DASD types is not allowed. If you use 3390 DASD as the IPL DASD, all additional dump DASD must also be 3390. If you use SCSI DASD as the IPL DASD, the additional dump DASD must be SCSI as well.

If using 3390 DASD, the device must already have a label, which may be the result of the DASD being CP-formatted, CMS formatted or previously formatted by SDINST. If your device does not have a label, you can quickly put a label on it by using the CMS FORMAT command and formatting just 1 cylinder.

The IPL and all additional DUMP devices must be available in read/write mode.

For dumping a first level system (LPAR), each dump device is an entire 3390 or SCSI LUN. Currently, a 3390 used by stand-alone dump is limited to 65520 cylinders when allocated on an IBM TotalStorage DASD subsystem. That is about 45 GB of data. The maximum SCSI device support is limited to about 1 TB because VM's EDEVICE support is used when loading the dump.

When using the stand-alone dump program to dump a second level system, minidisks can be used. The minidisk used for the IPL device must be at least 75 cylinders. Minidisks for additional dump devices could be as small as 1 cylinder but that is not a reasonable size for dump devices. The minimum SCSI LUN sizes are equivalent: 53 MB for the IPL device and 0.7 MB for additional dump devices.

Stand-alone dump does not support list-directed IPL from ECKD and does not support secure IPL from SCSI or ECKD. Therefore, you cannot set the following DUMPDEV parameters to IPL z/VM stand-alone dump in a virtual machine:

- ECKD
- SECURE

The size of the dump produced by stand-alone dump will be approximately the same size as a SNAPDUMP or CP hard abend dump of the same system if the dump includes all PGMBKs and the entire CP frame table. To determine how many dump devices will be needed, use the output of the CP QUERY DUMP command from the running system under a heavy load, multiplied by 1.5, to get an estimate of the number of 4K pages that the dump will contain. This estimate should be used as a minimum value and may not be sufficient in some situations, such as if dumping a system that had a "memory leak" problem. To be safe and cover every situation, allocate dump space equal to the configured storage size of the LPAR. Each cylinder on a 3390 DASD holds 180 4K pages. Note that the first part of the IPL device (60 cylinders on a 3390, or approximately 42 MB for SCSI) is used by the stand-alone dump program and the rest of the IPL device will hold the first piece of the dump. All of the space on any additional dump devices is used to hold the dump.

If dumping storage for a problem that occurred during CP IPL, such as a disabled wait state, the dump size will be approximately 2% of the LPAR's real memory plus 32 MB. The 32 MB accounts for 16 MB for the CP nucleus plus 16 MB for the System Execution Space Page Management Table.

Examples for Generating the Stand-Alone Dump Utility

The following is an example of generating the stand-alone dump utility. In this example:

- The IPL and DUMP DASD are on 3390 DASD with an address of 555.
- There is only one device for IPL and DUMP.

Under CMS, enter:

```
SDINST
```

The exec replies:

```
HCPSDI8682I STAND-ALONE DUMP INSTALLER 7.3
ENTER 'QUIT' AT ANY PROMPT TO TERMINATE EXECUTION.
HCPSDI8654A PLEASE ENTER THE VIRTUAL DEVICE ADDRESS OF THE IPL DASD
```

Enter:

```
555

HCPSDI8655A PLEASE ENTER THE REAL DEVICE ADDRESS FOR THIS DASD
OR ENTER Y TO USE 0555
Y

HCPSDI8674A DASD HAS EXISTING VOLID DSD555.
DO YOU WISH TO CONTINUE? (Y/N)
Y

HCPSDI8658A ARE THERE ANY ADDITIONAL DUMP DASD TO SPECIFY? (Y/N)
N

SSPKnn   IMAGE    F1 F     80     113581     1738 yyyy-mm-dd hh:mm:ss
SSPPnn   IMAGE    F1 F     80          1        1 yyyy-mm-dd hh:mm:ss
SSPInn   IMAGE    F1 F     80      76478     1494 yyyy-mm-dd hh:mm:ss
SADUnn   IMAGE    F1 F     80      15655      306 yyyy-mm-dd hh:mm:ss

CP SPOOL PUNCH CL N TO * NOHOLD NOEOF NOKEEP
RDR FILE 1146 SENT FROM MYUSERID PUN WAS 1146 RECS 114K CPY 001 N ...
RDR FILE 1147 SENT FROM MYUSERUD PUN WAS 1147 RECS 0001 CPY 001 N ...
RDR FILE 1148 SENT FROM MYUSERID PUN WAS 1148 RECS 076K CPY 001 N ...

CP SPOOL RDR   CL N NOHOLD EOF
SPOOL RDR OPTIONS HAVE BEEN CHANGED TO CLASS N, NOHOLD AND EOF.
IF FOLLOWING IPL IS SUCCESSFUL, YOU WILL NEED TO RESTORE SPOOL
RDR OPTIONS MANUALLY BECAUSE CONTROL DOES NOT RETURN TO THIS EXEC.

EVERYTHING IS READY.  ANSWER Y TO CONTINUE AND IPL THE DASD FORMATTING
PROGRAM.  ANY OTHER RESPONSE EXITS WITHOUT DOING THE IPL.
Y

CP IPL 00C CLEAR PARM loglevel=0 sspmod=SADUnn:400
nnnnnnnn FILES CHANGED
HCPSDI8660I SETTING DEVICE 0191 ONLINE.
HCPSDI8662I FORMATTING 3339 CYLINDERS ON DEVICE 0555.
HCPSDI8664I WRITING PARTITION TABLE TO DEVICE 0555.
HCPSDI8666I CREATING FILE SYSTEM FOR IPL PARTITION ON DEVICE 0555.
HCPSDI8668I INSTALLING STAND-ALONE DUMP TO THE IPL PARTITION ON
DEVICE 0555.
HCPSDI8670I STAND-ALONE DUMP INSTALLATION IS COMPLETE.
YOU MAY RE-IPL CMS.
HCPGSP2629I The virtual machine is placed in CP mode due to a
SIGP stop from CPU 00.
```

Do not be concerned by the FILES CHANGED message. The program issues a CHANGE RDR ALL KEEP NOHOLD command to prevent accidental deletion of any reader files.

Taking the Stand-Alone Dump

To take a stand-alone dump:

1. Stop the system. If the system is at a disabled wait state, it is already stopped. Otherwise use the Stop All function on the HMC. For a second level system, use the CP STOP CPU ALL command.
2. Use the Store Status function on the HMC or check the **Store Status** box when you do the Load function in the next step. Type STORE STATUS on the CP command line if dumping a second level system.
3. IPL the dump device. Use the Load Normal function on the HMC. Do not use the CLEAR option and erase any data that is in the Load Parameter field. If dumping a second level system, do not specify the CLEAR and LOADPARM operands.

The dumper displays messages on the system console as it progresses. For example, these messages were displayed when dumping a 10 GB second level system to a 3390 DASD at device number 555.

```
CP STOP CPU ALL

STORE STATUS
Store complete.

TERM CONMODE 3215

IPL 555
Booting default...
Starting Stand-alone Dump
Done initializing disks
Dumping Frame Table
Dumping memory:
  00000000 / 00000047 MB
  00000008 / 00000047 MB
  00000016 / 00000047 MB
  00000024 / 00000047 MB
  00000032 / 00000047 MB
  00000040 / 00000047 MB
  00000047 / 00000047 MB
Stand-alone Dump was successful.
HCPGSP2629I The virtual machine is placed in CP mode
due to a SIGP stop from CPU 00.
```

Processing the Stand-Alone Dump Data

Re-IPL the z/VM system. To read the dump from the dump devices, use the DUMPLD2 utility and specify the DASD operand. Attach the dump devices to a userid before invoking the DUMPLD2 utility. The following example shows loading a dump from virtual device 777 to filemode B and naming the dump TEMPLOAD.

```
ATTACH 555 * 777
DASD 0555 ATTACHED TO OPERATNS 0777 WITH DEVCTL
Ready;

DUMPLD2  DASD  OUTFILE TEMPLOAD TO B
HCPDLD8278A Enter virtual device number of first dump device.
777
DUMP ON 0777 WAS CREATED yyyy-mm-dd hh:mm:ss

  TOTAL PAGES 32626  DUMPER RC 0

DASD  DUMPLD2         DUMP PAGES DUMP DUMP
TYPE   VDEV   VOL-ID   ON DEVICE RDEV  RC
----  ------- ------  ---------- ---- ----
3390   0777   DSD55     32626    0555    0

A total of 1 file(s) (32626 records each) will be created.
HCPDLD8210I 1 TEMPLOAD MDMPxxxx file(s) will be created on disk B
Continue - Y/N ?
y
HCPDLD8213I Created TEMPLOAD MDMP0001 B
Ready;
```

The DUMPLD2 utility requires all of the devices that are part of the stand-alone dump configuration to be attached at the same virtual device numbers that were used when SDINST formatted them. If this is not the case, error messages are displayed and the dump is not loaded. As shown in the above example, the

DUMPLD2 utility displays a device table that shows, under the DUMPLD2 VDEV column, the virtual device (VDEV) address where the real device (RDEV) should be attached.

After loading the dump to CMS DASD, use the VM Dump Tool to process the dump. For details on using the DUMPLD2 utility, see *z/VM: CP Commands and Utilities Reference*. For details on using the VM Dump Tool, see *z/VM: VM Dump Tool*.

Example for Generating the Stand-Alone Dump Utility on SCSI

The following is an example of installing the stand-alone dump utility on to SCSI devices. In this example, the IPL SCSI device uses FCP device number 1F01 and the additional dump device uses FCP 1F02. A separate FCP device must be used for each DASD.

```
vary on 1f01
1 device(s) specified; 1 device(s) successfully varied online
Ready;

att 1f01 * 1401
FCP 1F01 ATTACHED TO OPERATNS 1401
Ready;

vary on 1f02
1F02 varied online
1 device(s) specified; 1 device(s) successfully varied online
Ready;

att 1f02 * 1402
FCP 1F02 ATTACHED TO OPERATNS 1402
Ready;

SDINST
HCPSDI8682I STAND-ALONE DUMP INSTALLER 7.3
ENTER 'QUIT' AT ANY PROMPT TO TERMINATE EXECUTION.
HCPSDI8654A PLEASE ENTER THE VIRTUAL DEVICE ADDRESS OF THE IPL DASD.
1401

HCPSDI8655A PLEASE ENTER THE REAL DEVICE ADDRESS FOR THIS DASD
OR ENTER Y TO USE 1F01
y

HCPSDI8656A PLEASE ENTER THE 16 HEXADECIMAL DIGIT WWPN FOR THIS DASD.
50050763040313DE

HCPSDI8657A PLEASE ENTER THE 16 HEXADECIMAL DIGIT LUN FOR THIS DASD.
4011402900000000

HCPSDI8658A ARE THERE ANY ADDITIONAL DUMP DASD TO SPECIFY? (Y/N)
y

HCPSDI8654A PLEASE ENTER THE VIRTUAL DEVICE ADDRESS OF THE ADDITIONAL DUMP DASD
1402

HCPSDI8655A PLEASE ENTER THE REAL DEVICE ADDRESS FOR THIS DASD
OR ENTER Y TO USE 1F02
y

HCPSDI8656A PLEASE ENTER THE 16 HEXADECIMAL DIGIT WWPN FOR THIS DASD.
50050763040313DE

HCPSDI8657A PLEASE ENTER THE 16 HEXADECIMAL DIGIT LUN FOR THIS DASD.
4011402A00000000

HCPSDI8658A ARE THERE ANY ADDITIONAL DUMP DASD TO SPECIFY? (Y/N)
n

SSPKnn   IMAGE    A1 F    80     113581    1738 yyyy-mm-dd hh:mm:ss
SSPPnn   IMAGE    A1 V    80          1       1 yyyy-mm-dd hh:mm:ss
SSPInn   IMAGE    A1 F    80      79108    1546 yyyy-mm-dd hh:mm:ss
SADUnn   IMAGE    A1 F    80      15779     309 yyyy-mm-dd hh:mm:ss

CP SPOOL PUNCH CL N TO * NOHOLD NOEOF NOKEEP
RDR FILE 0004 SENT FROM OPERATNS PUN WAS 0004 RECS 114K CPY  001 N N...
RDR FILE 0005 SENT FROM OPERATNS PUN WAS 0005 RECS 0001 CPY  001 N N...
RDR FILE 0006 SENT FROM OPERATNS PUN WAS 0006 RECS 079K CPY  001 N N...

CP SPOOL RDR   CL N NOHOLD EOF
SPOOL RDR OPTIONS HAVE BEEN CHANGED TO CLASS N, NOHOLD AND EOF.
```

```
IF FOLLOWING IPL IS SUCCESSFUL, YOU WILL NEED TO RESTORE SPOOL
RDR OPTIONS MANUALLY BECAUSE CONTROL DOES NOT RETURN TO THIS EXEC.

EVERYTHING IS READY.  ANSWER Y TO CONTINUE AND IPL THE DASD FORMATTING
PROGRAM.  ANY OTHER RESPONSE EXITS WITHOUT DOING THE IPL.
y

CP IPL 00C CLEAR PARM loglevel=0 sspmod=SADUnn:191
     NO FILES CHANGED
HCPSDI8660I SETTING DEVICE 0191 ONLINE.
HCPSDI8664I WRITING PARTITION TABLE TO DEVICE 1401.
HCPSDI8666I CREATING FILE SYSTEM FOR IPL PARTITION ON DEVICE 1401.
HCPSDI8664I WRITING PARTITION TABLE TO DEVICE 1402.
HCPSDI8668I INSTALLING STAND-ALONE DUMP TO THE IPL PARTITION ON
DEVICE 1401.
HCPSDI8670I STAND-ALONE DUMP INSTALLATION IS COMPLETE.
YOU MAY RE-IPL CMS.
HCPGSP2629I The virtual machine is placed in CP mode due to a
SIGP stop from CPU 00
```

Do not be concerned by the FILES CHANGED message. The program issues a CHANGE RDR ALL KEEP NOHOLD command to prevent accidental deletion of any reader files.

Taking the Stand-Alone Dump on SCSI

To take a stand-alone dump:

1. Stop the system. If the system is at a disabled wait state, it is already stopped. Otherwise use the Stop All function on the HMC. For a second level system, use the CP STOP CPU ALL command.
2. To IPL from SCSI on the real processor, access the **Load** window on the processor console and click the **SCSI dump** box. To IPL from SCSI in a virtual machine, you will specify the virtual device number of the FCP device and use the DUMP operand on the IPL command. Also, you will use the CP SET DUMPDEV command to specify the additional information that you specify on the **Load** HMC panel.

   * Specify the load address of the FCP device of the IPL DASD.
   * Specify all blanks in the Load Parameter field.
   * Specify the world wide port name (PORT on the SET DUMPDEV command)
   * Specify the logical unit number (LUN)
   * Specify 0 for the boot program selector (BOOTPROG)
   * Specify 0 for the boot record logical block address (BR_LBA)
   * Specify all blanks in the OS specific load parameters field (SCPDATA)

   To continue the example from "Example for Generating the Stand-Alone Dump Utility on SCSI" on page 399, specify 1F01 as load address, WWPN as 50050763040313DE and LUN as 4011402900000000.
3. To see the progress of the stand-alone dump, open the SYSC console on the HMC (Operating System Messages). If using z/VM, type `CP TERM CONMODE 3215`.
4. IPL the dump device. For dumping a second level system, use the CP IPL command with the DUMP operand.

   The dumper displays messages on the system console as it progresses. For example, these messages were displayed when dumping a 200 GB first level system.

   ```
   HCPLDI2816I Acquiring the machine loader from the processor controller.
   HCPLDI2817I Load completed from the processor controller.
   HCPLDI2817I Now starting the machine loader.
   MLOEVL012I: Machine loader up and running (version v2.3).
   MLOPDM003I: Machine loader finished, moving data to final storage location.
   Starting Stand-alone Dump
   Done initializing disks
   Dumping Frame Table
   Dumping memory:
   00000000 / 00001747 MB
   00000291 / 00001747 MB
   00000582 / 00001747 MB
   00000874 / 00001747 MB
   00001165 / 00001747 MB
   ```

```
00001456 / 00001747 MB
00001747 / 00001747 MB
Stand-alone Dump was successful.
```

Processing the Stand-Alone Dump Data on SCSI

Re-IPL the z/VM system. To read the dump from the dump devices, use the DUMPLD2 utility and specify the DASD operand. For this procedure, the dump devices will be accessed using z/VM's EDEVICE support. The edevices must be defined and then attached to the userid as virtual devices before trying to load the dump. If you do not know all of the dump devices that were used, run DUMPLD2 twice. The first time, specify the first device and DUMPLD2 will find the dump and display information about the dump devices. When it asks if it should continue, type n and then attach the additional dump devices as necessary. Run DUMPLD2 again and answer y when asked to continue.

Note: The EDEVs must be attached to the user using consecutive virtual device numbers, such as 1401, 1402, and so on.

The following example shows loading a dump from virtual EDEVs 1401 and 1402 to filemodes C and D and naming the dump MYSAD. The same FCP devices, WWPNs and LUNs that were used in the previous sections are used here.

```
vary on 1f01-1f02
1F01 varied online
1F02 varied online
2 device(s) specified; 2 device(s) successfully varied online
Ready;

CP SET EDEVICE 1401 TYPE FBA ATTR 2107 FCP_DEV 1F01
   WWPN 50050763040313DE LUN 4011402900000000
EDEV 1401 was created.
Ready;

CP SET EDEVICE 1402 TYPE FBA ATTR 2107 FCP_DEV 1F02
   WWPN 50050763040313DE LUN 4011402A00000000
EDEV 1402 was created.
Ready;

vary on 1401
1401 varied online
1 device(s) specified; 1 device(s) successfully varied online
Ready;

vary on 1402
1402 varied online
1 device(s) specified; 1 device(s) successfully varied online
Ready;

att 1401 * 1401
DASD 1401 ATTACHED TO OPERATNS 1401
Ready;

att 1402 * 1402
DASD 1402 ATTACHED TO OPERATNS 1402
Ready;

dumpld2 dasd small outfile mysad to C D
HCPDLD8278A Enter virtual device number of first dump device.
1401
DUMP ON 1401 WAS CREATED yyyy-mm-dd hh:mm:ss
     TOTAL PAGES 1184452  DUMPER RC 0

DASD   DUMPLD2  DUMP PAGES                                               DUMP
TYPE    VDEV     ON DEVICE    FCP       WWPN              LUN           RC
----   -------  ----------    ----  ----------------  ----------------   ----
SCSI   1401       1040375     1F01  50050763040313DE  4011402900000000    0
SCSI   1402        144077     1F02  50050763040313DE  4011402A00000000    0

A total of 19 file(s) (65536 records each) will be created.
HCPDLD8210I 15 MYSAD MDMPxxxx file(s) will be created on disk C
HCPDLD8210I 4 MYSAD MDMPxxxx file(s) will be created on disk D
Continue - Y/N ?
y
HCPDLD8213I Created MYSAD MDMP0001 C
.
.
```

```
.
HCPDLD8213I Created MYSAD MDMP0019 D
Ready;
```

Chapter 13. Creating and Modifying Image Libraries for Printers

This chapter describes how to:

- Create text decks for IBM 3800 and impact printers
- Install and modify image libraries
- Display information about image libraries
- Purge image libraries
- Find information on how to back up image libraries
- Find more information about printers.

For a printer to work properly, it must have access to an *image library*. This image library is a set of modules that define the spacing, characters, and copy modification data the printer needs to format and print information. For each printer type there can be a number of IMAGE libraries, but the IMAGE library used for any one file must contain both the UCS images and the FCB images required for the printer to properly format and print the file.

An image library requires a *text deck* as its input file. The procedure for creating a text deck for a 3800 is different from the procedure for creating a text deck for an impact printer. This chapter explains both methods.

If your installation dedicates a printer to a virtual machine, use the facilities of the operating system you run in the virtual machine to build the appropriate image libraries. If your installation is going to use a printer as a CP spooling device (that is, a device that CP uses to print data for one or more virtual machines), you can follow the procedures described in this book to install image libraries.

Creating Text Decks

A text deck is the input file required to produce an image library. A text deck contains forms control buffers (FCBs), copy modifications, character arrangement tables, graphic character modifications, or library character sets.

Creating Text Decks for the 3800

IBM supplies a 3800 image library feature tape. This feature tape is preinstalled on the z/VM System DDR, source is excluded. Control files for a 3800 image library are included.

Character sets supplied by IBM are supplied as separate files on the S disk; they are named XTB1*xxxx* TEXT S. These fonts are described in the *IBM 3800 Printing Subsystem Programmer's Guide*.

If you are modifying a 3800 image library or creating your own 3800 image library, you need to use the GENIMAGE utility to create a text deck for that image library.

The following is an example of the command to initiate GENIMAGE:

```
genimage sysin file * sysprint listing a1
```

sysin file *
is the file name, file type, and file mode of the CMS input file. (The file mode can represent an accessed minidisk or an accessed shared file system directory.)

sysprint listing a1
is the file name, file type, and file mode of the file in which GENIMAGE places a message listing. (The file mode can represent an accessed minidisk or an accessed shared file system directory.) GENIMAGE produces two groups of output files: a message listing and a text deck.

For more information on the GENIMAGE utility, see *z/VM: CP Commands and Utilities Reference*. For information on creating your own FCBs, copy modifications, and library character sets for a 3800, see the *IBM 3800 Printing Subsystem Programmer's Guide*.

Creating Text Decks for Impact Printers

With the z/VM System DDR media or DVD, IBM supplies text decks, assemble files, and control files, which you can use to create image libraries for impact printers. Impact printers supported by z/VM are listed in *z/VM: General Information*.

If you are modifying an image library or creating your own image library, you need to create a text deck for that image library. To do so, you must:

1. Code the appropriate macroinstructions to add universal character sets (UCSs) and FCBs to produce assemble files. For more information on the naming restrictions and conventions for FCBs and UCSs, see "Universal Character Sets and FCBs Supplied by IBM" on page 404, "Forms Control Buffers Supplied by IBM" on page 410, and "Naming Conventions for UCS Buffer Images and FCBs" on page 408.
2. Process the assemble file with the VMFHASM utility. The output of VMFHASM is a text deck. If you have the H Assembler then use the VMFHASM command. If you have the High Level Assembler then use the VMFHLASM command.

Universal Character Sets and FCBs Supplied by IBM

The IBM-supplied buffer images for each printer are located in an assemble file of the form *nnnnxxxx* ASSEMBLE (*nnnn* is the printer type, *xxxx* is the buffer image name).

For example, the AN buffer image associated with the 3203 is located in 3203AN ASSEMBLE.

IBM supplies the following UCS buffer images for the 3203 printer:

Name	Meaning
AN	Normal alphanumeric notation character set arrangement
HN	Normal hexadecimal notation character set arrangement
PCAN	Preferred alphanumeric notation character set arrangement
PCHN	Preferred hexadecimal notation character set arrangement
PN	PL/I—60 graphics
QN	PL/I—60 graphics
QNC	PL/I—60 graphics
RN	FORTRAN, COBOL commercial
SN	Text printing 84 graphics
TN	Text printing 120 graphics
YN	High-speed alphanumeric

IBM also supplies the following UCS buffer images for devices that emulate the 3211 printer.

Name	Meaning
A11	Standard commercial
H11	Standard scientific
G11	ASCII
P11	PLI
T11	Text printing

IBM supplies the following UCS buffer images for the 3262 printer:

Name	Meaning
P48	USA EBCDIC
P52	Austria/Germany character set
P63	USA EBCDIC
P64	USA EBCDIC
P96	USA EBCDIC
P128	Katakana character set

IBM supplies the following FCB images for the 3203, 3262, 4245, and 4248 printers and devices that emulate the 3211 printer:

Name
Meaning

FCB1
Space—6 lines per inch; Length of page—66 lines

Line Represented	Channel Skip Specification
1	1
3	2
5	3
7	4
9	5
11	6
13	7
15	8
19	10
21	11
23	12

Line Represented	Channel Skip Specification
64	9

FCB8
Space—8 lines per inch; Length of page—68 lines

Line Represented	Channel Skip Specification
1	1
4	2
8	3
12	4
16	5
20	6
24	7
28	8
32	10
36	11
63	12
66	9

FCBS
Space—8 lines per inch; Length of page—68 lines

Line Represented	Channel Skip Specification
1	1
54	2
55	3
56	4
57	5
58	6
59	7
60	8
61	10
62	11
63	12
64	9

For the exact contents of these buffer images, see the *IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide*.

Notes:

1. During print line buffer (PLB) loading, the 3211 attaches to the end of the UCS a 64-byte associative field. This buffer is used to ensure that each character that is loaded into the PLB for printing is also on the print train.

For the 3203, the dualing and uncomparable table (DUCT) serves a similar purpose. For more information on coding the DUCT, see the *IBM 3203 Printer Model 5 Component Description and Operator's Guide*.

2. The forms control buffer (FCB) for a virtual 3203, 3262 Model 5, 4245, 4248 or a device that emulates a 3211 should be compatible with the FCB that is loaded in the real counterpart; otherwise, the results will be unpredictable.

Adding New Universal Character Set Buffer Images

If the UCS buffer images that IBM supplies do not meet your needs, you can change a buffer image or create a new buffer image.

With z/VM, two UCS macroinstructions define a UCS for any impact printer. To create the text deck to add a new print buffer image to z/VM, you must:

1. Provide a buffer image name and a 12-byte header for the buffer load by coding the UCSI macroinstruction.
2. Provide the exact image of the print chain.
3. Code the UCSICCW macroinstruction and produce the appropriate ASSEMBLE file. The UCSICCW macroinstruction marks the end of the print image and optionally creates a CCW to print the image.
4. Process the assemble file with the VMFHASM utility. The output of VMFHASM is a text deck. For more information on VMFHASM, see "Using VMFHASM or VMFHLASM to Create a Text Deck" on page 414.

Note: Only one UCS may be defined in an ASSEMBLE file.

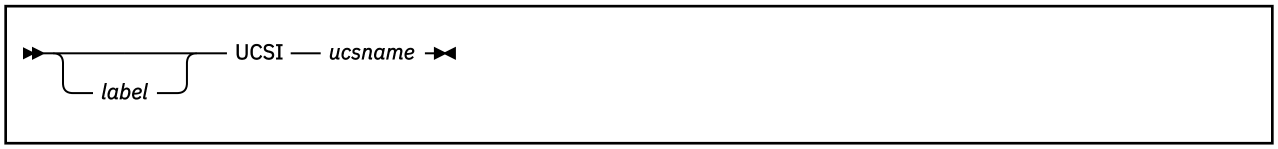
Macros are available that make the process of adding new print buffer images relatively easy. The following procedures tell you how to use these macroinstructions. Using them also helps you avoid error.

The UCSI Macro

The UCSI macroinstruction creates a 12-byte header record for the buffer load for any supported impact printer. CP uses the header record during processing of the LOADBUF and START commands.

Note that the UCS buffer contains up to 512 characters.

The syntax of the UCSI macroinstruction is:

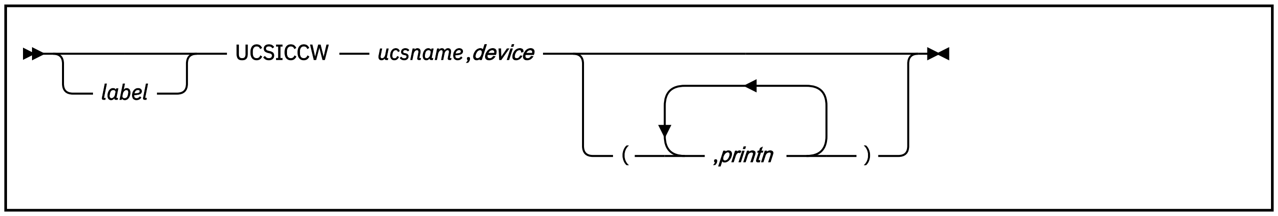


ucsname
is a 1- to 4-character name that is assigned to the buffer load.

After you have coded the UCSI macroinstruction, you must supply the exact print image. To supply the print image, code DCs in hexadecimal or character format. The print image may consist of several DCs and may be up to 512 characters in length. The image consists of a train image and possibly a dualing and uncomparable character table (DUCT). The length of the data for each of these areas varies depending on the the printer type. Refer to the appropriate hardware manual for details. A list of some useful hardware manuals appears near the end of this chapter.

UCSICCW Macro

After you have supplied the print image, code the UCSICCW macroinstruction. The UCSICCW macroinstruction marks the end of the print image and optionally creates a CCW string to print the buffer load image when the operator specifies VERIFY on the LOADBUF command. This macroinstruction applies to any supported impact printer. The syntax of the UCSICCW macroinstruction is:



ucsname
: is a 1- to 4-character name assigned to the buffer load by the UCS macroinstruction.

device
: is the device name of the printer with which the buffer is associated.

printn
: are the line lengths to be printed. The variable *print1* represents the number of characters printed on the first line, *print2* represents the number of characters printed on the second line, and so on. Each count specified must be between 1 and the maximum length of the print line for the printer you specify. Up to 12 print fields may be specified. However, the total number of characters to be printed may not exceed the maximum for that printer. If the lengths are not specified, the default is 48 characters per line for the entire UCS buffer image.

Naming Conventions for UCS Buffer Images and FCBs

When you modify or create a new UCS or FCB, follow the naming conventions listed below.

- The name of the file containing UCS or FCB macroinstructions is composed of the printer type followed by the name of the character set. For the 3203 printer with the PN character set, UCS macroinstructions are found in 3203PN ASSEMBLE. This naming convention means that, for a given printer, no UCS may have the same name as an FCB.
- The CSECT name in an assemble file is the name of the character set. In 3203PN ASSEMBLE, the CSECT name is PN.
- Text files have the same file name as the associated assemble file and have a file type of TEXT. The text file associated with 3203PN ASSEMBLE is 3203PN TEXT.
- Each UCS must be defined in a separate ASSEMBLE file.

Examples of coding assemble files for printers are given in this chapter. For information on creating text decks, see "Using VMFHASM or VMFHLASM to Create a Text Deck" on page 414.

Examples of New 3203 UCS Buffer Images

Example 1

This example defines a buffer image called EX01. The associated module in which you include the code for EX01 should be called 3203EX01 ASSEMBLE.

You do not have to specify the line length for verification of the buffer load. Insert the following code in 3203EX01 ASSEMBLE:

```
EX01     CSECT
         SPACE
         COPY HCPCWOEQ
         COPY HCPEQUAT
         SPACE
EX01     CSECT ,
         UCSI    EX01
         DC      5C'1234567890A...Z1234567890*/'
         DC      X'00101010101010101010004000404000'   240-255
         DC      X'40101010101010101010004040400000'   256-271
         DC      X'40401010101010101010004000000000'   272-287
         DC      X'10101010101010101010000000404000'   288-303
         UCSICCW EX01,3203
         END     EX01
```

The buffer image is five representations of a 48-character string containing:

- The alphabetic characters
- The numeric digits, twice
- The special characters asterisk (*) and slash (/).

Example 2

This example defines a buffer image called NUM1. The associated module in which you include the code for NUM1 should be called 3203NUM1 ASSEMBLE. Insert the following code in 3203NUM1 ASSEMBLE:

```
EX02     CSECT
         SPACE
         COPY HCPCWOEQ
         COPY HCPEQUAT
         SPACE
EX02     CSECT ,
         UCSI    NUM1
         DC      24C'1234567890'
         DC      X'00101010101010101010004000404000'   240-255
         DC      X'40101010101010101010004040400000'   256-271
         DC      X'40401010101010101010004000000000'   272-287
         DC      X'10101010101010101010000000404000'   288-303
         UCSICCW NUM1,3203,(60,60,60,60)
         END     NUM1
```

The NUM1 print buffer consists of twenty-four 10-character entries. If, after 3203NUM1 ASSEMBLE is reloaded, you enter the command:

```
loadbuf 00e ucs num1 ver
```

four lines of 60 characters (the 10-character string repeated six times) are printed to verify the buffer load.

Examples of New 3211 UCS Buffer Images

This example defines a buffer image called A11. The associated module in which you include the code for A11 should be called 3211A11 ASSEMBLE.

Enter the following code for the A11 UCS buffer image in 3211A11 ASSEMBLE:

```
ALL      CSECT
         SPACE
         COPY HCPCWOEQ
         COPY HCPEQUAT
         SPACE
ALL      CSECT ,
*      a11 standard commercial 48 graphics 3211
         UCSI    ALL
         DC      9C'1<.+IHGFEDCBA*$-RPQONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC      X'000000'                         433-435
         DC      X'000000000000000000000000101010' 436-450
         DC      X'101010101010100040404240004010' 451-465
         DC      X'101010101010101000404041000040' 466-480
         DC      X'401010101010101010004040000000' 481-495
         DC      X'101010101010101010100040404448' 496-510
         DC      X'0000'                           511-512
         UCSICCW ALL,3211,(48,48,48,48,48,48,48,48,48)
         END     ALL
```

Note that the DC specification contains 49 characters and the UCSICCW macroinstruction specifies 48 characters. The ampersand (&) must be coded twice to be accepted by the assembler. The single quotation mark (') must also be specified twice in order to be accepted.

Examples of New 3262 UCS Buffer Images

This example defines a buffer image called P48. The associated module in which you include the code for P48 should be called 3262P48 ASSEMBLE.

Enter the following code for the P48 UCS buffer image in 3262P48 ASSEMBLE:

```
P48      CSECT
         SPACE
         COPY HCPCWOEQ
         COPY HCPEQUAT
         SPACE
P48      CSECT ,
*                'usa ebcdic 48 character'
         UCSI  P48
         SPACE
*                 0 1 2 3 4 5 6 7 8 9 a b c d e f
         DC    X'F1F2F3F4F5F6F7F8F9F07B7C61E2E3E4' 000
         DC    X'E5E6E7E8E9506B6CD1D2D3D4D5D6D7D8' 010
         DC    X'D9605B5CC1C2C3C4C5C6C7C8C94E4B7D' 020
         DC    X'F1F2F3F4F5F6F7F8F9F07B7C61E2E3E4' 030
         DC    X'E5E6E7E8E9506B6CD1D2D3D4D5D6D7D8' 040
         DC    X'D9605B5CC1C2C3C4C5C6C7C8C94E4B7D' 050
         DC    X'F1F2F3F4F5F6F7F8F9F07B7C61E2E3E4' 060
         DC    X'E5E6E7E8E9506B6CD1D2D3D4D5D6D7D8' 070
         DC    X'D9605B5CC1C2C3C4C5C6C7C8C94E4B7D' 080
         DC    X'F1F2F3F4F5F6F7F8F9F07B7C61E2E3E4' 090
         DC    X'E5E6E7E8E9506B6CD1D2D3D4D5D6D7D8' 0A0
         DC    X'D9605B5CC1C2C3C4C5C6C7C8C94E4B7D' 0B0
         DC    X'F1F2F3F4F5F6F7F8F9F07B7C61E2E3E4' 0C0
         DC    X'E5E6E7E8E9506B6CD1D2D3D4D5D6D7D8' 0D0
         DC    X'D9605B5CC1C2C3C4C5C6C7C8C94E4B7D' 0E0
         DC    X'F1F2F3F4F5F6F7F8F9F07B7C61E2E3E4' 0F0
         DC    X'E5E6E7E8E9506B6CD1D2D3D4D5D6D7D8' 100
         DC    X'D9605B5CC1C2C3C4C5C6C7C8C94E4B7D' 110
         UCSICCW P48,3262,(48,48,48,48,48,48)
         END   P48
```

Forms Control Buffers Supplied by IBM

z/VM provides three FCB images: FCB1, FCB8, and FCBS. These FCBs are located in assemble files of the form *nnnnxxxx* ASSEMBLE, where:

- The variable *nnnn* is the printer type
- The variable *xxxx* is the buffer image name.

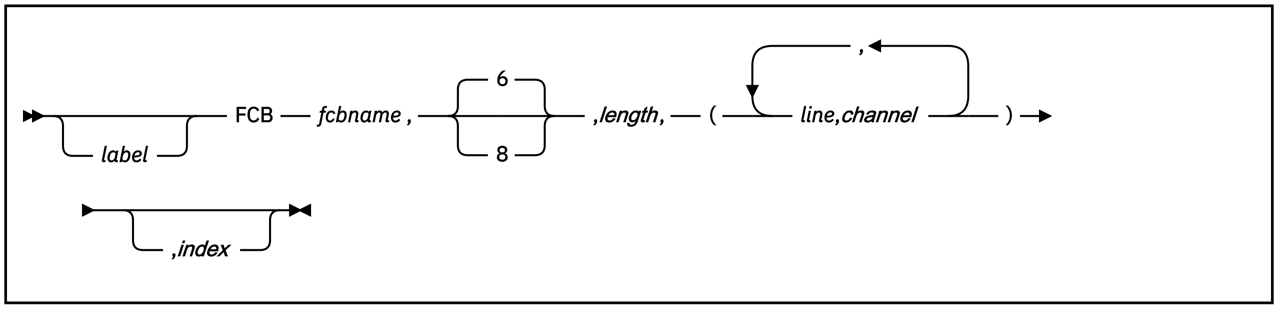
For example, the FCB1 buffer image associated with the 4248 printer is located in 4248FCB1 ASSEMBLE.

Adding a New Forms Control Buffer

To add a new forms control buffer for your real or virtual printer, use the FCB macroinstruction. This macroinstruction applies to the following printers:

3203 Model 5
Devices that emulate the 3211
3262 Model 5
4245
4248.

When you add a new FCB, make sure you follow the naming conventions given under "Naming Conventions for UCS Buffer Images and FCBs" on page 408. The format of the FCB macroinstruction for the printers listed above is:



fcbname
: is the name of the forms control buffer. The variable *fcbname* can be 1 to 4 alphanumeric characters in length.

space
: is the number of lines per inch. Valid specifications are 6 and 8. This operand can be omitted; the default is six lines per inch. When the space operand is omitted, a comma (,) must be coded. This operand has no meaning for a virtual printer.

length
: is the number of print lines per page or carriage tape (2 to 255).

(*line,channel*...)
: shows which print line (*line*) corresponds to each channel (*channel*). The values for *channel* range from 1 to 12. The entries can be specified in any order.

index
: is an index value from 1 to 31. The variable *index* specifies the print position that is to be the first printed position. It is valid only for devices that emulate the 3211 printer.

To create the text deck that will add a new FCB:

1. Code the FCB macroinstruction in the appropriate assemble file.
2. Process the assemble file with the VMFHASM utility. The output of VMFHASM is a text deck. If you have the H Assembler then use the VMFHASM command. If you have the High Level Assembler then use the VMFHLASM command. For more information on VMFHASM or VMFHLASM, see "Using VMFHASM or VMFHLASM to Create a Text Deck" on page 414.

Notes:

1. If you code the FCB macroinstruction with more than one channel designated for one print line, the macroinstruction includes only the last channel in the buffer for that print line. (A buffer byte can only be loaded with one channel code.)
2. The forms control buffer *must* have compatibility with channel 1; that is, channel 1 and line 1 must be the same physical line for all FCBs that are built. If they are not, the forms will be misaligned.
3. Each FCB assemble file should only contain one FCB or message HCPCSB241E will be issued when you try to load any FCB other that the first FCB into the printer.
4. Make sure you access the disk that contains the HCPOM2 MACLIB before assembling your file. The HCPOM2 MACLIB contains the CP version of the FCB MACRO. Otherwise, the CMS version of the FCB MACRO might be found in the DMSOM MACLIB and that will not create a valid text deck.

Example:

Code the FCB macroinstruction (using the name SPEC), if you want:

- Your printer to print 8 lines per inch
- Your printer to print 60 lines per page
- Channel 1 to correspond to print line 1
- Channel 12 to correspond to print line 40
- Channel 9 to correspond to print line 60

- Print position 10 to be the first print position.

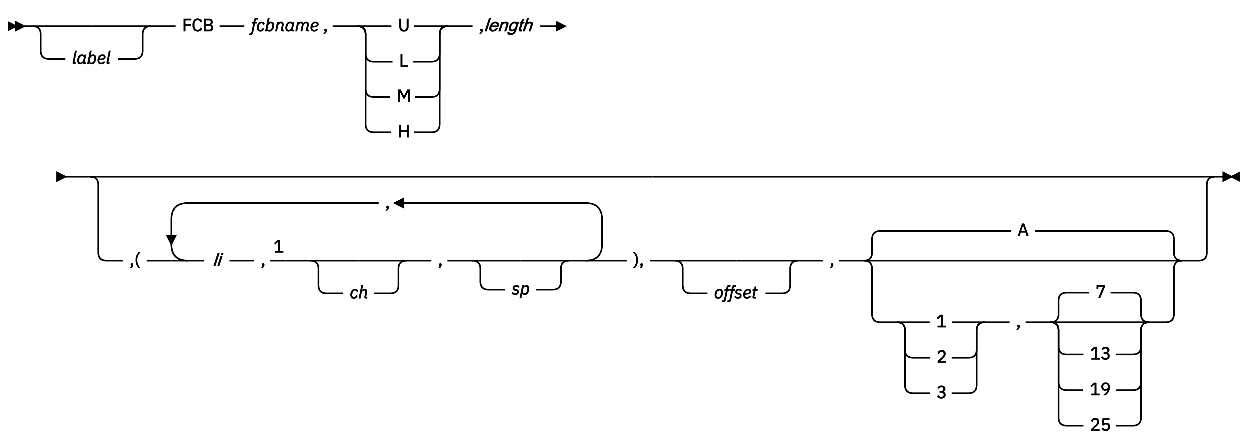
```
fcb spec,8,60,(1,1,40,12,60,9),10
```

If you want another forms control buffer, called LONG, to be exactly the same as SPEC with the exception that only six lines are printed per inch, code either of the following:

```
fcb long,6,60,(1,1,40,12,60,9),10
fcb long,,60,(1,1,40,12,60,9),10
```

Adding an Extended Forms Control Buffer for the 4248 Printer

To add an extended forms control buffer for your real or virtual 4248 printer, use the following format of the FCB macroinstruction:



Notes:

[1] The *li*,*ch*,*sp* values are positional; therefore, if the *ch* or *sp* or both values are omitted, the separator commas are required to maintain the proper positioning. However, if on the last line-channel-space definition the *ch* and *sp* values are omitted, the separator commas are not required, or if the *sp* value is omitted the separator comma is not required.

fcbname
is the name of the forms control buffer. The variable *fcbname* can be from 1 to 4 alphanumeric characters in length.

speed
is the speed at which the 4248 printer runs. The value of *speed* must be one of the following:

Value	**Meaning**
U	Unchanged
L	Low
M	Medium
H	High

length
is the number of lines on the output page (2 to 256).

li,ch,sp
is the information that defines a line in the FCB. You can specify multiple lines by separating the lines with commas. Enclose the entire line definition with one pair of parentheses.

li
specifies the line number on the page. This must be between 1 and the value you specified for length.

ch
specifies the channel code (1 to 12), or can be omitted. If you omit this operand, no channel is defined on this line.

sp
specifies the lines-per-inch (LPI) spacing. Spacing is variable with each line. Acceptable values are 6 and 8; or, you can omit this operand. If omitted, the line spacing already in effect does not change.

offset
specifies the character position in which the duplicate copy begins if you require the duplicate feature. If you omit this operand, no duplicate copy prints.

levels
controls the level of the paper stacker. Acceptable values are:

Value	Meaning
A	Automatic stacker level control, which is the default
1	Paper tray lowered 1 inch below automatic position
2	Paper tray lowered 2 inches below automatic position
3	Paper tray lowered 3 inches below automatic position

rate
specifies the paper stacker drop rate. Acceptable values are:

Value	Meaning
7	Drop the paper tray after seven sheets, which is the default
13	Drop the paper tray after 13 sheets
19	Drop the paper tray after 19 sheets
25	Drop the paper tray after 25 sheets

Note: If you specify the automatic stacker level control, the stacker drop rate is ignored.

Notes:

1. If you code the FCB macroinstruction with more than one channel designated for one print line, the macroinstruction includes only the last channel in the buffer for that print line. (A buffer byte can only be loaded with one channel code.)
2. The forms control buffer *must* have compatibility with channel 1; that is, channel 1 and line 1 must be the same physical line for all FCBs that are built. If they are not, the forms will be misaligned.
3. Each FCB assemble file should only contain one FCB or message HCPCSB241E will be issued when you try to load any FCB other that the first FCB into the printer.

4. Make sure you access the disk that contains the HCPOM2 MACLIB before assembling your file. The HCPOM2 MACLIB contains the CP version of the FCB MACRO. Otherwise, the CMS version of the FCB MACRO might be found in the DMSOM MACLIB and that will not create a valid text deck.

Example:

If you want:

- Your printer to run at medium speed, print 8 lines per inch, and 60 lines per page
- Channel 1 to correspond to print line 1
- Channel 12 to correspond to print line 40
- Channel 9 to correspond to print line 60
- To begin the duplicate copy at position 67
- To set the paper tray 2 inches below the automatic position
- To drop the paper tray after 25 sheets.

code the FCB macroinstruction (using the name SPEC) as:

```
fcb spec,m,60,(1,1,8,40,12,,60,9),67,2,25
```

If you want another forms control buffer, called LONG, to be exactly the same as SPEC with the exception that, at line 40, the spacing be 6 lines per inch, code the following:

```
fcb long,m,60,(1,1,8,40,12,6,60,9),67,2,25
```

Using VMFHASM or VMFHLASM to Create a Text Deck

After you have created the appropriate assemble files to modify a UCS or FCB buffer image, use the VMFHASM or VMFHLASM EXEC to create a text deck. If you have the High Level Assembler then use the VMFHLASM command. If you have the H Assembler then use the VMFHASM command. *Example*

To create a text deck from the file 3203PN ASSEMBLE, enter:

```
vmfhlasm 3203pn fn
```

where:

3203pn
is the file name of the 3203PN ASSEMBLE file.

fn
is the file name of a control file whose file type is CNTRL. You can create your own control file or you can use the HCPVM CNTRL file that is supplied by IBM. Refer to *z/VM: VMSES/E Introduction and Reference* for the format of a control file. The important thing about the control file that you use is that the CP maclibs should precede the CMS maclibs on the "MACS" statement. Specifically, HCPOM2 must precede DMSOM so that the correct FCB MACRO is used for the assembly.

VMFHASM or VMFHLASM creates the file 3203PN TEXT, which you can then use to create an image library.

For more information on the VMFHASM or VMFHLASM execs see the *z/VM: VMSES/E Introduction and Reference*. For information on image libraries, see "Installing Your Own Image Library" on page 415.

Image Libraries

This section describes the tasks associated with image libraries.

Default Image Library Names

The default image library used for a printer is IMAG*nnnn*, where *nnnn* is the device type of the printer.

Some types of printers may be configured to emulate another type of printer. In these cases, *nnnn* is the device type of the emulated printer, because that is the device type that is defined to CP. For example, a 4248 may emulate a 3211. In this case, the default image library is IMAG3211 because the printer is defined to CP as a 3211.

Whether you use the default image library name or override the default by specifying a name on either the RDEVICE statement or CP START command, before an image library can be used, it must be installed as described below.

Installing the Image Library That IBM Provides

As part of the z/VM System DDR media or DVD, IBM provides sample image library control files. For example, the image library control file for a 3800 printer is in a CMS file named IMAG3800 CNTRL. These control files are normal CMS files that can reside in an accessed Shared File System directory or on a minidisk. If you find that these control files do not meet your needs, you can change them or create your own. Whether you use the control files supplied by IBM or create your own, you must issue the IMAGELIB utility to install the image library associated with those control files. "Installing Your Own Image Library" on page 415 below describes how to do this.

For additional information on the IMAGELIB utility, refer to *z/VM: CP Commands and Utilities Reference*.

Installing Your Own Image Library

Use the IMAGELIB utility to install your own image library or the image library provided by IBM. Before you issue IMAGELIB, you must create a control file whose file name is the same as the image library you want to build and whose file type is CNTRL. The control file is a normal CMS file that can reside in an accessed shared file system directory or on a minidisk. The format of this file is one statement per text deck you want included, with names in columns 1 to 8.

The general format of IMAGELIB is as follows:

```
imagelib libname
```

The variable *libname* is the 1- to 8-character alphanumeric file name of the image library you want to install. (Keep in mind that image library members must be in files with a file type of TEXT, and the names of the library parts must be in a CNTRL file.)

Example 1:

To install an image library for which the components are listed in the file named 3800LIB1 CNTRL, enter:

```
imagelib 3800lib1
```

Example 2:

To install an image library for which the components are listed in the file named 3203LIB2 CNTRL, enter:

```
imagelib 3203lib2
```

For additional information about the IMAGELIB utility, refer to *z/VM: CP Commands and Utilities Reference*.

Modifying Image Libraries

Use the IMAGEMOD utility to alter existing image libraries or to create a map of an image library. For additional information about the IMAGEMOD utility, refer to *z/VM: CP Commands and Utilities Reference*.

Adding Files to an Image Library

Use the ADD operand of IMAGEMOD to add files to an existing image library. For example, to ADD the images XT10 and XT12, from the IBM supplied text decks XTB1XT10 and XTB1XT2, to the image library named IMAG3800, enter:

```
IMAGEMOD ADD IMAG3800 XTB1XT10 XTB1XT12
```

Note: For names of other IBM supplied images for the 3800, refer to "Creating Text Decks for the 3800" on page 403.

Deleting Members from an Image Library

Use the DEL operand of IMAGEMOD to delete members from an existing image library. For example, to delete the members named XTB1ODA and XTB1ONA from the image library named IMAG3800, enter:

```
IMAGEMOD DEL IMAG3800 XTB1ODA XTB1ONA
```

Replacing Members or Modules in an Image Library

Use the REP operand of IMAGEMOD to replace existing modules of an image library with modules of the same name. For example, to replace the modules named XTB1GS10, XTB1GS12, and XTB1GS15, enter:

```
IMAGEMOD REP IMAG3800 XTB1GS10 XTB1GS12 XTB1GS15
```

Creating an Image Library Map

Use the MAP operand of the IMAGEMOD command to create a map of an image library. Use the TERM, PRINT, and DISK options to specify where you want the map sent.

Example 1:

To create a map of an image library named IMAG3800 and send the map to your console, enter:

```
imagemod map imag3800 (term
```

Example 2:

To create a map of an image library named IMAG3211 and send the map to your virtual printer, enter:

```
imagemod map imag3211 (print
```

Example 3:

To create a map of an image library named IMAG3800 and place the map in a CMS file on your A disk, enter:

```
imagemod map imag3800 (disk
```

Note that the map will be contained in a file named IMAG3800 MAP A5.

Displaying Information about Image Libraries

Use the QUERY IMG command to display information about your image libraries. (Note that QUERY IMG can be entered by class A, B, C, D, and E users.)

For example, to display information about all of your image libraries, enter:

```
query img all
```

To display information about a particular image library, for example, the one named IMAG3800, enter:

```
query img name imag3800
```

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

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

For the meanings of these fields, refer to *z/VM: CP Commands and Utilities Reference*.

Purging Image Libraries

Use the PURGE IMG command to get rid of unwanted files that contain image libraries. (Note that PURGE IMG can be entered by class A, B, C, D, and E users.) For example, to purge a 3800 image library named IMAGTEST, enter:

```
purge img name imagtest
```

After this command is entered, CP purges IMAGTEST *unless* a printer is still using it. If a printer is still using IMAGTEST, CP places the file in a *pending-purge* state. CP purges the file as soon as the printer releases it.

To determine whether a file is in pending-purge state, enter the QUERY IMG command. If the file is in pending-purge state, CP's response shows that the file is class P.

Keeping Backup Copies of Image Libraries

CP uses system spooling space to store image libraries. Because you may not always be able to recover spooling space after a CP abnormal termination, you should use the CP SPXTAPE command to keep backup copies of image libraries on tape. (Note, however, that the system tries to preserve system data files [SDFs] over a cold start.) For more information, see *z/VM: System Operation* or *z/VM: CP Commands and Utilities Reference*.

Where to Find More Information about Printers

For information on using the GENIMAGE utility to create text decks that contain 3800 FCB images, copy modifications, character arrangement tables, graphic character modifications, and library character sets, see *z/VM: CP Commands and Utilities Reference*.

For information on operating a 3800 printer as a real spooling device, see *z/VM: System Operation*.

For detailed information on the 3800 printer itself, see:

- *Introducing the IBM 3800 Printing Subsystem and Its Programming*
- *IBM 3800 Printing Subsystem Programmer's Guide, OS/VS1, OS/VS2*
- *Reference Manual for the IBM 3800 Printing Subsystem*.

For more information on other printers mentioned in this chapter, see:

- *IBM 3203 Printer Model 5 Component Description and Operator's Guide*
- *IBM 3262-1/11 Printer Component Description*
- *IBM 3262-2/12 Printer Component Description*
- *IBM 3262-3/13 Printer Component Description*
- *IBM 3262-1/11 Printer Operator's Guide*
- *IBM 3262-2/12 Printer Operator's Guide*
- *IBM 3262-3/13 Printer Operator's Guide*
- *IBM 4245-1 Printer Component Description and Operator's Guide*
- *IBM 4248-1 Printer Operator's Introduction and Reference*
- *IBM 4248-2 Printer Operator's Introduction and Reference*.

Chapter 14. CCW Translation

This chapter describes how to code the macroinstructions used at installation time to change the way CP handles CCWs.

Using CCW Translation

An installation can control how CP handles specific CCWs. CP uses a set of tables to translate the guest's virtual channel programs into real channel programs that CP issues to the real device. These tables identify the characteristics of each CCW command (00 - FF) for the device class and type. An installation with only supported DASDs would not need to alter these tables. However, if the installation contains devices that act slightly different from supported DASD, a system installation programmer would want to slightly alter specific CCW entries in these tables.

The table name is the same as the file name of an ASSEMBLE file that must be updated, assembled, and incorporated in place of the standard table TEXT file when building the CP system.

Table 18 on page 419 shows the table module's file name and the macroinstruction name associated with each device class.

Table 18. Table and Macroinstruction Names for Device Classes

Table	Macroinstruction	Device Class
HCPTCS	HCPCHSCD	Check sorter
HCPTDD	HCPDDPCD	Dedicated DASD
HCPFBD	HCPDDPCD	Dedicated FBA DASD
HCPTUD	HCPDDPCD	DASD, unsupported
HCPTMD	HCPMDPCD	Minidisk
HCPFBM	HCPMDPCD	FBA Minidisk
HCPTTP	HCPTPPCD	Tape
HCPUTT	HCPTPPCD	Tape, unsupported
HCPTMT	HCPCONCD	Terminal, 3215
HCPTMT	HCPDSPCD	Terminal, 3270
HCPTMT	HCPTRMCD	Terminal, TERM
HCPTUT	HCPTRMCD	Terminal, unsupported
HCPTUR	HCPURPCD	Unit record
HCPUUR	HCPURPCD	Unit record, unsupported

Coding the Device Class Macroinstruction

To code a device macroinstruction to change the way that CP handles a specific CCW for the device:

1. Display the *filename* ASSEMBLE file for the device class. (The *filename* is the same as the table name in Table 18 on page 419 for the device class. For example, to view the information for minidisks, display the HCPTMD ASSEMBLE file.)
2. Locate the table name associated with the device type and model you want to change. The table name appears in the specifications section of the assemble file listing. For example, the table name for 9345 minidisks is TMD93451. The specification section of the assemble file contains other information and

notes that you may want to review; for example, the device macroinstruction parameters that are valid for the device class.

3. Locate the CCW entry within the table that you want to change.
4. Code the CCW exactly as it appears in the *filename* ASSEMBLE file but change the conditions to satisfy your needs. You must code a change for each CCW you want to redefine. See "Device Macroinstruction" on page 421 for the valid parameter choices.
5. When all CCW changes are coded, code one more entry with LAST=YES.
6. Assemble the changed table and include the generated text deck when building the CP system nucleus.

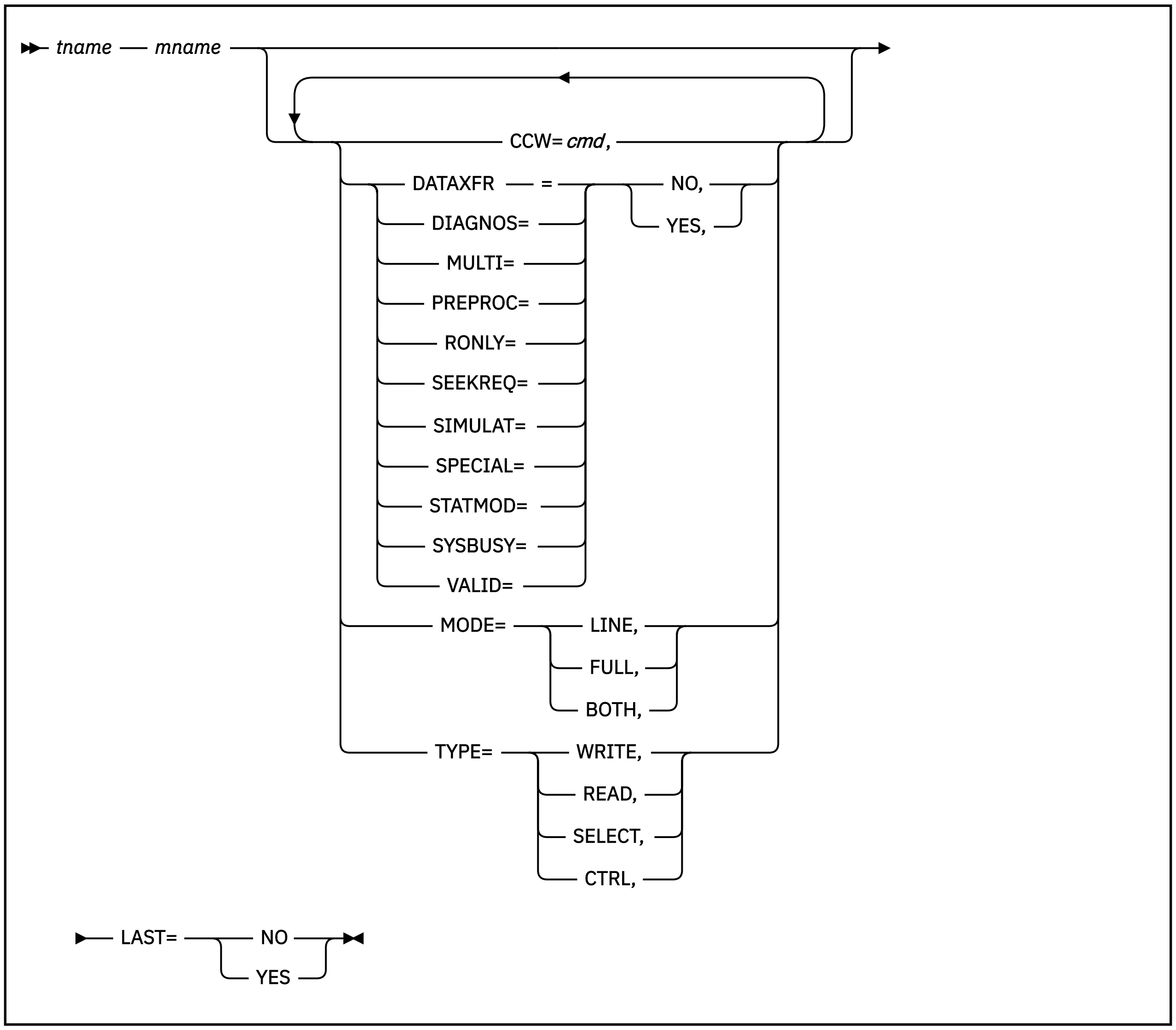
Note: If I/O is still rejected after making CCW table updates, contact z/VM service to determine if something else is preventing the request from being accepted.

Unsupported Devices Tables

The unsupported device tables each contain one table for how CP should treat that class of unsupported device. All unsupported devices of the same class are treated identically.

CP treats an unsupported DASD, for example, in many of the same ways that it treats a supported dedicated DASD. The only exceptions are those specified in the unsupported DASD table. CP handles channel program CCWs as you specified them in the table.

Device Macroinstruction



Parameters

tname mname
: identifies the device translation table and macroinstruction for the device class. These must match the table and macroinstruction names shown in Table 18 on page 419. These cannot be changed because CP code references these names.

CCW=*cmd*
: identifies the command code. The *cmd* is expressed in hex in the range X'00' to X'FF'.

DATAXFER=NO
DATAXFER=YES
: indicates if the command causes a data transfer on the channel.

DIAGNOS=NO
DIAGNOS=YES
: indicates if the command is a diagnostic CCW.

MULTI=NO
MULTI=YES
: indicates if the command is a multitrack operation. This is an internal CP flag.

PREPROC=NO
PREPROC=YES
indicates if preprocessing is required. This is an internal CP flag.

RONLY=NO
RONLY=YES
indicates if the command is prohibited (command rejected) on a read-only device.

SEEKREQ=NO
SEEKREQ=YES
indicates if orientation is required. That is, this CCW must have been preceded in the channel program by a SEEK CCW.

SIMULATE=NO
SIMULATE=YES
indicates if CCW simulation is required. This is an internal CP flag.

SPECIAL=NO
SPECIAL=YES
indicates that further checking is required, such as:

- Certain opcodes need no data translation
- Certain opcodes must be first in the channel program
- For certain opcodes, user class is checked.

This is an internal CP flag.

STATMOD=NO
STATMOD=YES
indicates if the device can set STATUS MODIFIED.

SYSBUSY=NO
SYSBUSY=YES
indicates if the command is allowed to be issued to a tape device that is currently marked as busy with a system function such as SPXTAPE. If it is not allowed, a unit check is returned with command reject set in the sense data.

VALID=NO
VALID=YES
indicates if the command is valid. VALID=NO causes a command reject for this CCW.

MODE=LINE
MODE=FULL
MODE=BOTH
indicates the screen mode.

TYPE=WRITE
TYPE=READ
TYPE=SELECT
TYPE=CTRL
indicates type of command.

LAST=NO
LAST=YES
indicates if this is the last invocation for a particular device table. It must be specified as YES for the last invocation in order to generate a device table.

Examples

Examples are not provided because you should be using the ASSEMBLE files as the base for your changes.

You cannot create new tables, labels, device classes, or supported device types. You can modify only existing tables and entries.

Chapter 15. z/VM HiperDispatch

z/VM's HiperDispatch support is designed to improve your workload's performance in the memory subsystem.

z/VM HiperDispatch improves performance by working with the IBM Z hardware to run virtual servers in a cache-advantaged way. This section first explains how the memory hardware is organized and why this is important for performance. Next, you learn how Processor Resource/System Manager (PR/SM) and z/VM work together to take advantage of the hardware configuration. Finally, you learn how to plan and configure your z/VM system to exploit HiperDispatch.

Background on IBM Z Cache Structure

CPUs are connected to memory through a layered cache structure.

First, each CPU has its own cache (L1 and L2). Several CPUs are then grouped together on a chip, and they share a cache at this level (L3). Chips are then grouped together into multi-chip modules. Each multi-chip module is put onto a book, which has its own cache (L4). Finally, the books contain memory. This organization of CPUs into chips and books with shared cache is referred to as topology.

The amount of time it takes to resolve a memory reference is directly related to the number of caches that must be traversed. References resolved in the L1 cache are the fastest, while references that must be found in memory take the longest.

Exact CPU topology and number of cache levels varies by machine. In the following example there is one book, which has its own L4 cache and memory. The book contains four chips, each chip sharing the L4 cache. The CPUs on each chip share the L3 cache. Each CPU on a chip has its own L1 and L2 cache.

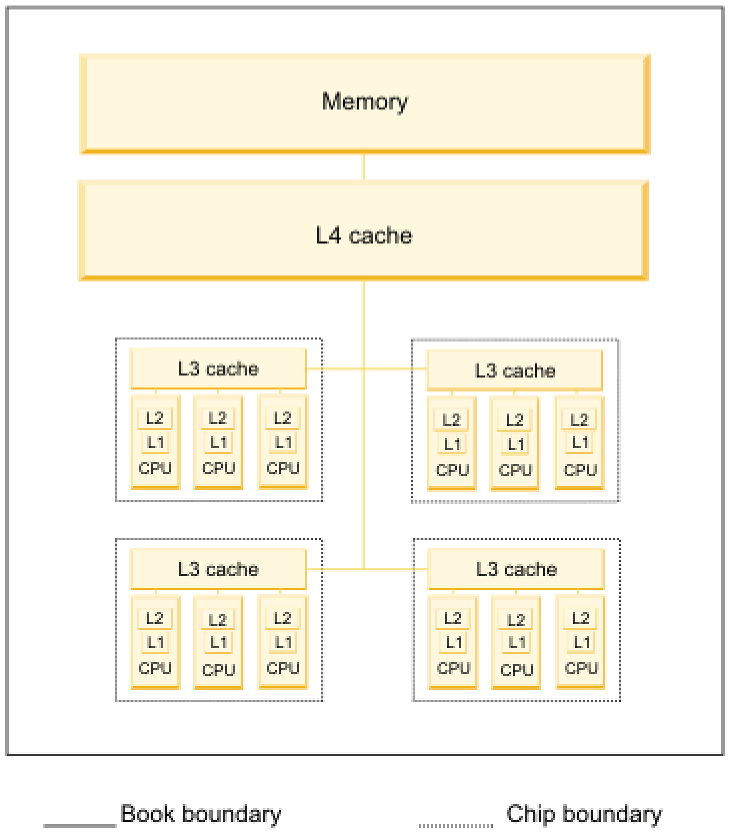


Figure 12. Example of a book with cache structures

Software Performance Considerations

Software gets the best performance when its memory references are resolved in the cache levels closest to the CPU.

A guest on a hypervisor, for instance, would benefit from its virtual CPUs being consistently run on the same, or nearby, real CPUs. Two real CPUs are considered to be nearby if they share all caches except for the per-CPU cache. Ideally, virtual CPUs from different guests would run on different real CPUs to avoid putting unrelated memory contents into the cache. In the world of virtualization, however, it would not be efficient, or even possible, to assign only one virtual CPU to each real CPU. Thus, the hypervisor must balance the cache benefits to guests against the efficient use of CPUs.

PR/SM Concepts

To understand HiperDispatch, you need to understand PR/SM entitlement, horizontal polarization mode, and vertical polarization mode.

Entitlement

To understand how PR/SM uses machine topology information, first you should know how PR/SM determines entitlement for a partition. Entitlement is the amount of real CPU time each partition is guaranteed. Entitlement for a partition with shared CPUs is a function of the LPAR's weight, the sum of the weights for all other shared partitions, and the number of shared physical CPUs in the Central Processing Complex (CPC). Entitlement is calculated separately for each CPU type as the number of shared CPUs multiplied by the ratio of an LPAR's weight to the sum of the weights of all shared partitions.

Figure 13 on page 424 shows a CPC with three shared LPARs and three physical CPUs (each of type CP) and the relationship between weight and entitlement on each LPAR.

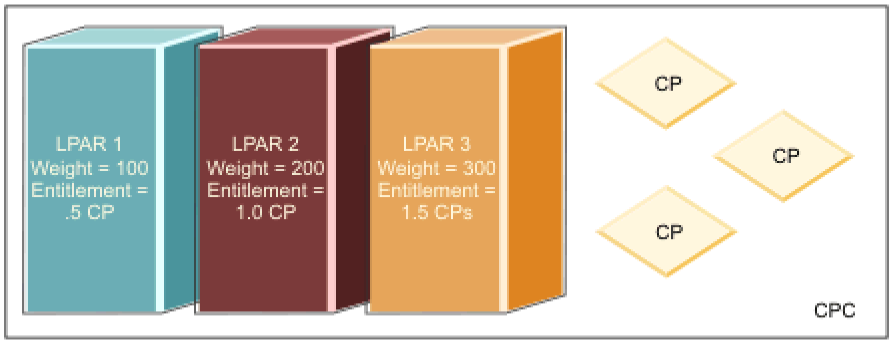


Figure 13. The relationship between weight and entitlement on each LPAR

While each partition is entitled to a certain amount of CPU time, it may use more or less than its entitlement. A partition might be running a workload that does not require all of its entitlement. This situation creates unused CPU time that other partitions can use if they need it. For example, partition LPAR 1 might be running a workload that needs more than its entitled CPU time. If another partition (LPAR 2) is using less than its entitlement, LPAR 1 might use the excess time. Partitions can use more than their entitlement only if they are not capped.

Entitlement for dedicated partitions is different than for shared ones. In a partition with dedicated logical CPUs, entitlement for each logical CPU equals 100% of one real CPU.

Horizontal Polarization Mode

Historically, z/VM did not use topology information and ran in a *horizontal polarization mode* partition. To understand horizontal polarization mode, consider a partition with 16 online logical CPUs of type CP that has an entitlement equal to 12 real CPUs worth of power. In this mode:

- PR/SM distributes entitlement equally across all logical CPUs, so each of the 16 logical CPs is entitled to 75% of a physical CPU. When all other partitions consume their entitlements, each of these 16 will get no more than 75% of a real CPU.
- PR/SM does not necessarily make much effort to run a z/VM partition's logical CPUs over and over on the same real CPUs. So the 75% of a real CPU that each logical CPU receives could be spread across several real CPUs. This weak affinity can have serious effects on caches, leading to longer memory references.
- Furthermore, even if the logical CPU were always mapped to the same real CPU, as much as 25% of the time the real CPU could be running completely unrelated work, possibly even consisting of a number of low-consuming logical CPUs from other partitions, diluting cache even more.

These cache effects can have serious consequences for guest performance. Clearly, a better solution is needed.

Vertical Polarization Mode

PR/SM is able to give operating systems two cache advantages. First, PR/SM can provide information about where logical CPUs are placed in the physical topology. Second, PR/SM can place logical CPUs to increase cache benefits. In *vertical polarization mode* PR/SM maps logical CPUs to real CPUs as closely to one another as possible and moves these mappings as little as possible.

PR/SM also tries to consolidate a partition's entitlement onto a subset of the logical CPUs that it places topologically near to one another. To do this, PR/SM divides logical CPUs into three types: vertical-highs, -mediums and -lows. Each vertical-high (Vh) CPU is entitled to 100% of one physical CPU, a vertical-medium (Vm) CPU is entitled to 50-100% of one physical CPU (except in cases where the partition's entitlement is less than 50%), and a vertical-low (Vl) CPU is entitled to 0% of one physical CPU. Vertical-medium and -low CPUs can consume CPU beyond their entitlement if other shared logical CPUs on the CPC are not consuming their entitlements.

Communication between PR/SM and the Operating System

PR/SM offers interfaces to allow an operating system to exploit topology information for its benefit. The operating system (such as z/VM) can request to change between vertical and horizontal modes or can query the current mode. The operating system can also sense its own entitlement, as well as the entitlement and consumption of other partitions, so the operating system can calculate whether there is any excess CPU available. Finally, the operating system can query the topology of the real CPUs to which its logical CPUs are assigned and whether this mapping has changed recently. A smart operating system or hypervisor can use this information to improve performance for itself and its guests.

The Objective of z/VM HiperDispatch

The objective of HiperDispatch is to help guest workloads get improved performance from the memory subsystem.

To accomplish this objective, z/VM can switch the partition to vertical polarization mode and exploit partition topology information. Topology awareness lets z/VM more efficiently use processors and run guest CPUs in topologically intelligent ways. Topological information about where the partition's entitlement is concentrated lets z/VM consolidate workloads onto a smaller set of logical CPUs.

Topology awareness has important advantages for the z/VM dispatcher. It can now group together virtual CPUs belonging to the same guest, so they are dispatched on logical, and in turn real, CPUs that share cache. z/VM also tries to keep guest CPUs running on the same logical CPUs, again, for cache benefits.

When the partition is running in vertical mode and global performance data (GPD) is enabled, z/VM makes estimates about future values of two distinct quantities that influence the decision about how many processors to unpark. First, z/VM tracks CPU consumption data in its own LPAR in order to project a "ceiling" (or maximum) value above which it is very unlikely that the consumption will rise in the very near future. This projected maximum is called the *consumption ceiling projection*. Second, z/VM tracks CPU usage and entitlement across the entire CPC in order to project a "floor" (or minimum) value on how much power PR/SM would let the z/VM LPAR consume if the LPAR tried in the very near future to consume as much as it possibly could. This projected minimum is called the *capacity floor projection*. z/VM refreshes the consumption ceiling projection and the capacity floor projection every two seconds.

z/VM uses the consumption ceiling projection, the SRM CPUPAD setting, and the capacity floor projection to decide how many processors to unpark. How those three numbers are combined is decided by an SRM setting called the *unparking model*, which is controlled by the CP SET SRM UNPARKING command. Three unparking models are supported: large, medium, and small. Each model makes the unparking decision differently, as follows:

- The *large* unparking model unparks processors according to the following rule.

  It unparks all of the vertical-high logical processors, all of the vertical-medium logical processors, and all the vertical-low logical processors it projects that PR/SM will "power", where *power* is defined by the capacity floor projection.
- The *medium* unparking model unparks processors according to the following rule.

  It unparks all of the vertical-high logical processors and all of the vertical-medium logical processors, and it also unparks vertical-low logical processors, but only according to this relation:

  ```
  min(vertical-lows-needed, vertical-lows-powered)
  ```

  In other words, in contrast to the large model, the medium model unparks only the vertical-low processors that will be "needed" and "powered", where *needed* is defined by the consumption ceiling projection plus the CPUPAD setting and *powered* is defined by the capacity floor projection.
- The *small* unparking model unparks processors according to the following rule.

  It unparks:

  ```
  min(logical-processors-needed, logical-processors-powered)
  ```

  where *needed* is defined by the ceiling projection plus the CPUPAD setting and *powered* is defined by the capacity floor projection. In other words, in contrast to the medium model, the small model parks vertical-high and vertical-medium logical processors it appears the workload will not need.

To get the best performance from PR/SM, one rule of thumb for an LPAR to follow is to leave its vertical-low processors parked unless it appears that the LPAR's workload needs them and there appears to be power available to run them. Thus, many z/VM installations will probably find that the medium unparking model and the small unparking model help PR/SM operate efficiently. Furthermore, to get the best leverage from its entitled logical processors, a good rule of thumb is that an LPAR should use all of its entitled logical processors to run the workload, unless there is a good reason to avoid some of them. Thus, most installations will probably find the medium unparking model to be the most suitable. The other two models are provided to help the installation deal with edge cases.

In some GPD-enabled situations, it might be necessary for the system programmer to forbid z/VM from using vertical-low logical processors, no matter whether there might be capacity to run them. In such situations, the CP SET SRM EXCESSUSE NONE command will help. When GPD is on and EXCESSUSE NONE is in effect, z/VM will always leave all vertical-low processors parked.

The z/VM system IPLs with UNPARKING MEDIUM and EXCESSUSE MEDIUM set, unless the system programmer uses the z/VM system configuration file to change these settings. To make such a change, use the SRM statement of the system configuration file. See "SRM Statement" on page 269 for more information.

When the partition is running in vertical mode and GPD is disabled, z/VM cannot query usage across the CPC. So, instead of keeping CPUs in use according to available capacity, z/VM does this according to apparent demand. z/VM tracks its own utilization, projects a utilization ceiling, adds to the projected ceiling a padding value, and then uses CPUs according to the calculated sum. In this way z/VM is prepared to cover the projected demand plus maintaining the pad for headroom. z/VM must assume that PR/SM will power all of the in-use logical CPUs. z/VM has no way to validate this because GPD is disabled.

When the partition is running in horizontal mode, z/VM keeps all logical CPUs in use all the time.

Unnecessary or superfluous logical CPUs are placed into a new state called *parked*. CPUs put into parked state are not assigned work and remain in wait state until unparked. CPUs are placed into parked or unparked state based on the projections mentioned above. Partition topology information is used to decide which CPUs are most cache advantageous to unpark.

Remember that usage and entitlement are calculated separately for each CPU type. Thus, if running a partition with some Central Processors (CPs) and some Integrated Facility for Linux processors (IFLs), there might be different numbers of each unparked, depending upon workload, entitlement, and excess CPU available. Unentitled (vertical-low) CPUs are parked first, but vertical-high and vertical-medium CPUs might also be parked, if it would be advantageous to the system's workload.

Figure 14 on page 427 is an example of a mapping of logical CPUs to real CPUs in a vertical partition with entitlement equal to 10.2 CPs and 2.7 IFLs, where there are 21 logical CPs and 7 logical IFLs defined. Each of the 9 vertical-high CPs and 2 vertical-high IFLs has an entitlement of 100% of a real CPU. Each vertical-medium CP will have an entitlement of 60%, for a total of 1.2 CPUs. The vertical-medium IFL has an entitlement of 70% of a real CPU. The calculated usage projection for this LPAR is 18 CPs and 5 IFLs, which is illustrated by the unparked (in use) CPUs. Because some vertical-low CPUs are unparked, this means there is excess CPU available on the CPC.

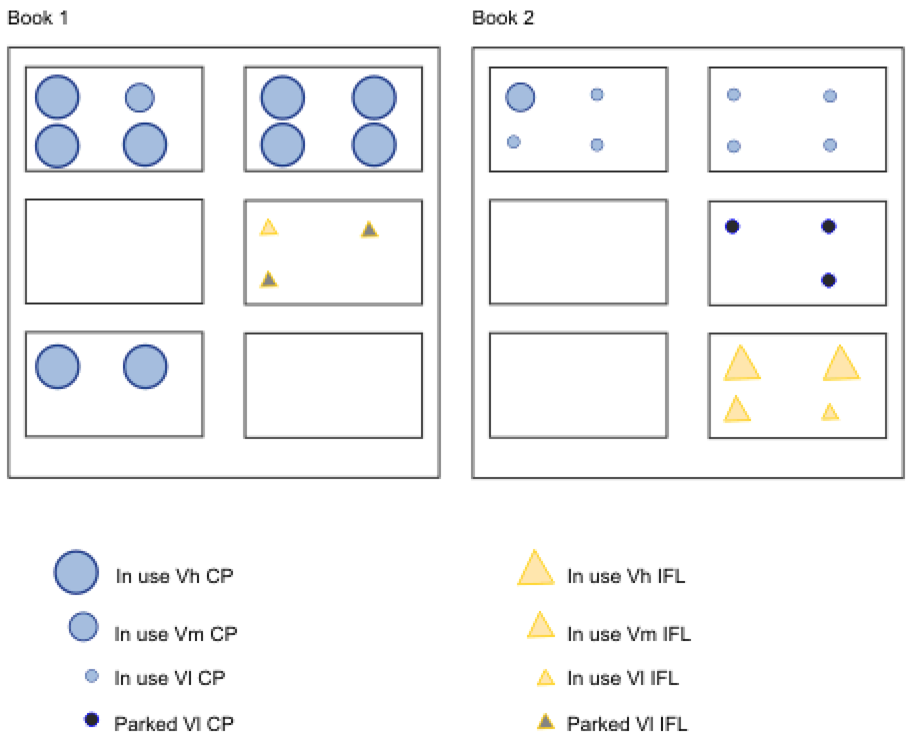


Figure 14. Example of a mapping of logical CPUs to real CPUs

Note: z/VM does not virtualize CPU topology or vertical polarization for guests. However, guests receive benefit from the z/VM topology-directed dispatching of virtual CPUs.

Questions and Answers: How to Take Advantage of HiperDispatch

The following are answers to your questions about configuring and administering your system to take advantage of HiperDispatch.

How do I know whether my workload would benefit from z/VM HiperDispatch?

For information about the characteristics of workloads that benefit from z/VM HiperDispatch, see the following on the web at IBM: VM Performance Resources (https://www.ibm.com/vm/perf/).

Is there any special way to set up my partition so that HiperDispatch works better?

If you decide to use vertical polarization mode, be sure to turn on global performance data (GPD) for the partition. GPD is a setting turned on in the partition's activation profile on the HMC (Hardware Management Console) or SE (Support Element). This setting allows PR/SM to tell a partition about usage on other partitions within the CPC. You can use the QUERY SRM command to see whether GPD is enabled.

How do I switch from horizontal polarization to vertical polarization mode (or vice versa)?

To switch between horizontal polarization and vertical polarization mode, use the POLARIZATION option on the SET SRM (System Resource Manager) command or the SRM system configuration statement.

Notes:

- When running second level, only horizontal polarization mode is allowed. If you attempt to go to vertical polarization mode you receive an error message.
- You can switch between horizontal polarization and vertical polarization mode at any time; however, there might be a short delay before the system begins to work optimally. When switching from vertical polarization mode to horizontal polarization, it can take up to two seconds to unpark all processors. Similarly, the switch from horizontal polarization mode to vertical can take up to two seconds for all appropriate processors to park.
- First level z/VM systems IPL in vertical polarization mode by default. To IPL in horizontal polarization mode, you must specify the SRM system configuration statement.

How do I query which mode is currently running?

Polarization mode appears in several places. You can use an option on the QUERY SRM command, POLARIZATION, to see which polarization mode is currently running. To view the polarization of each CPU, use INDICATE LOAD command or the QUERY PROCESSORS TOPOLOGY command (discussed further below). If you collect monitor records, you can see this information in the System, Monitor and Scheduler domains or on z/VM screens.

How do I know my partition topology and can I change it?

Use the TOPOLOGY operand on QUERY PROCESSORS to see the topology, but there are caveats. This operand is allowed only when topology is available; you get an error message when running second level. You see topology information when running first level in horizontal polarization mode, but remember the affinity between real and logical CPUs is not as strong as when running in vertical polarization mode.

You cannot change the mapping between real and logical CPUs. PR/SM determines this based on a partition's entitlement and number of logical CPUs. The mapping can change for a variety of reasons, including a new partition being activated, logical CPUs being varied off or on, or hardware errors.

Example:

```
q proc topo
TOPOLOGY
  NESTING LEVEL: 02  ID: 01
    NESTING LEVEL: 01  ID: 01
```

```
       PROCESSOR 01  ALTERNATE  CP    VH  0006
       PROCESSOR 02  ALTERNATE  CP    VM  0006
       PROCESSOR 03  ALTERNATE  CP    VH  0006
       PROCESSOR 04  ALTERNATE  CP    VH  0006
     NESTING LEVEL: 01  ID: 02
       PROCESSOR 05  MASTER     CP    VH  0004
       PROCESSOR 06  ALTERNATE  CP    VH  0004
       PROCESSOR 07  ALTERNATE  CP    VH  0004
       PROCESSOR 08  ALTERNATE  CP    VH  0004
     NESTING LEVEL: 01  ID: 04
       PROCESSOR 19  ALTERNATE  IFL   VL  0008
       PROCESSOR 1A  PARKED     IFL   VL  0008
       PROCESSOR 1B  PARKED     IFL   VL  0008
     NESTING LEVEL: 01  ID: 05
       PROCESSOR 09  ALTERNATE  CP    VH  0003
       PROCESSOR 0A  ALTERNATE  CP    VH  0003
   NESTING LEVEL: 02  ID: 02
     NESTING LEVEL: 01  ID: 01
       PROCESSOR 0B  ALTERNATE  CP    VM  0005
       PROCESSOR 0C  ALTERNATE  CP    VL  0005
       PROCESSOR 0D  ALTERNATE  CP    VL  0005
       PROCESSOR 0E  ALTERNATE  CP    VL  0005
     NESTING LEVEL: 01  ID: 02
       PROCESSOR 0F  ALTERNATE  CP    VL  0001
       PROCESSOR 10  ALTERNATE  CP    VL  0001
       PROCESSOR 11  ALTERNATE  CP    VL  0001
       PROCESSOR 12  ALTERNATE  CP    VL  0001
     NESTING LEVEL: 01  ID: 03
       PROCESSOR 13  PARKED     CP    VL  0002
       PROCESSOR 14  PARKED     CP    VL  0002
       PROCESSOR 1C  PARKED     CP    VL  0002
     NESTING LEVEL: 01  ID: 06
       PROCESSOR 15  ALTERNATE  IFL   VH  0007
       PROCESSOR 16  ALTERNATE  IFL   VH  0007
       PROCESSOR 17  ALTERNATE  IFL   VM  0007
       PROCESSOR 18  ALTERNATE  IFL   VL  0007
 Ready;
```

How do I tell how my guest's CPUs are dispatched on logical CPUs?

You cannot see or influence how z/VM is dispatching guest CPUs on logical CPUs. The z/VM internal algorithms determine how to dispatch guests.

While running in vertical polarization mode, how can I tell z/VM how many logical CPUs to use (unpark)?

You cannot specifically tell z/VM which CPUs to keep online and in use (unparked). This is determined by z/VM internal algorithms, based on capacity, consumption, topology information, the CPUPAD setting, and the unparking model. If your system is running in vertical polarization mode, you can make adjustments to influence the unparking models' calculations as follows:

- Regarding the notion of "padding" the consumption ceiling forecast, you can specify how much additional CPU capacity to keep unparked by using the CPUPAD operand on the CP SET SRM command or the SRM configuration statement. This is a pad beyond that which the system is projected to use. This option is useful when the load on the system is known to be "bursty", that is, having periodic spikes in CPU usage. In this case, the administrator would want z/VM to provision the system's capacity with some "headroom", so that if there is a sudden spike in utilization before z/VM next adjusts the system's capacity, there will be enough unparked capacity available to handle the spike. The default value of CPUPAD is 100%, or one vertical-high CPU's worth of power.
- The SRM operand EXCESSUSE controls how aggressively z/VM tries to use the unentitled power available on the CPC. In other words, the EXCESSUSE setting controls the magnitude of the capacity floor forecast. Presented with the exact same measurement data, the four EXCESSUSE options produce four different capacity floor forecasts. The EXCESSUSE HIGH setting produces a large-magnitude forecast, but the forecast is computed with a fairly low percentage confidence; in other words, the forecast is fairly generous but it is only somewhat likely to come true. The EXCESSUSE LOW setting produces a smaller forecast, but the confidence percentage is higher, that is, the forecast is smaller in magnitude and is more reliable. The EXCESSUSE NONE setting produces a capacity floor forecast exactly equal to the LPAR's entitlement; the forecast is 100% certain to come true, but it always

predicts that no unentitled power will be available. The percent-confidence value that z/VM used in computing the capacity floor forecast is available in the PRCPUP_CALFCONF field of monitor record Domain 5 Record 16 (MRPRCPUP).

Did HiperDispatch make any changes to how the z/VM dispatcher works?

The default z/VM work distribution algorithm is called *reshuffle*. This algorithm periodically alters the home assignments of the running virtual CPUs in order to balance the number of virtual CPUs running on each logical CPU. HiperDispatch changed the reshuffle algorithm to make it more aware of partition topology and to make it more aware of the special dispatching needs of virtual MP guests.

HiperDispatch also introduced a second work distribution algorithm, called *rebalance*. This algorithm tracks guest CPU utilization and periodically alters the home assignments of virtual CPUs in order to afford reliable CPU affinity to guests with intensive CPU utilization, to keep virtual CPUs of MP guest close to one another in the topology, and to balance the CPU demands on the logical CPUs. The rebalance algorithm is suitable for use for only those workloads having specific, easily identifiable resource consumption characteristics. You should use the rebalance algorithm only if IBM assesses your system and specifically directs you to use it. You can select the rebalance algorithm by using the DSPWDMethod option on the SET SRM (System Resource Manager) command or the SRM system configuration statement.

For more information about reshuffle and rebalance, see the following on the web at IBM: VM Performance Resources (https://www.ibm.com/vm/perf/).

Are there any other changes to z/VM for HiperDispatch?

Yes, some commands, directory statements and system configuration statements have changed due to HiperDispatch support.

MAXUSERS
: The default is now the system-defined limit of 99,999 logged-on users.

INDICATE LOAD
: The AVGPROC value now represents the average value of the portion of a real CPU that each logical CPU consumed. Previously, this value was calculated using a measure of how often z/VM loaded a wait state on the processor. This technique did not properly represent partially entitled logical CPUs.

INDICATE QUEUES EXPANDED
: You no longer see the App output column, which showed affinity between logical and virtual CPUs. You can see information about where virtual CPUs were dispatched by examining monitor records in the User domain or Performance Toolkit screens.

Chapter 16. Changing the Volume Labels and System Name of a Non-SSI z/VM System

This section provides procedures that you can use to accomplish the following tasks:

- Use DDR or FlashCopy® to create a z/VM system (referred to as the *target system*) from an existing non-SSI z/VM system (referred to as the *source system*).
- Change the DASD volume labels for the system you created with DDR or FlashCopy.
- Change the system identifier, or name, of a non-SSI z/VM system.

These procedures are presented as if you were performing all of the tasks in that order. However, you can perform just the tasks that you need. For example, perhaps you only want to change the name of a system. You might need to adjust the information provided according to the tasks that you are doing.

The following assumptions apply to these procedures:

- You have a running z/VM system where you will perform these steps that is not the source system.
- The source system is not a member of an SSI cluster.
- The source system is not currently IPLed.
- The source system is defined on DASD volumes that are not attached to the running system as SYSTEM or USER volumes.
- You are not running any security manager or directory manager products on the source system.

  If you do have a security manager or directory manager on the source system, you should ensure that the products will not be started automatically when you IPL the target system, as you might need to perform additional configuration steps for those products on the target system.
- You have put all installed service into production on the source system before copying the source system to the target system.
- You have not customized the source system to restrict where it can be IPLed or to restrict the DASD devices that are sensed and brought online when the system is IPLed.

  If you have made any changes of this sort, you need to remove the restrictions before copying the source system to the target system, or you need to adjust these instructions to account for your environment restrictions. For example, if you have updated the SYSTEM_IDENTIFIER statement in the SYSTEM CONFIG file of the source system to specify an IPL location, you need to remove the IPL location from the SYSTEM CONFIG file on the source system, or you will have to IPL the target system in the specified location until you change the SYSTEM CONFIG file on the target system.

Copying the Source System to New DASD Volumes

1. Log on to a user ID with class B authority on the running system that has access to (or can access) all of the CP utility programs that are needed.
2. Attach the DASD volumes for the source system (*source volumes*) in read-only mode. Issue the following command for each volume:

   ```
   attach sourcevol * r/o
   ```
3. Select DASD volumes to be used for the target system (*target volumes*). These volumes should be at least the same size as the source volumes you will copy. Attach the target volumes in read-write mode. Issue the following command for each volume:

   ```
   attach targetvol *
   ```
4. If you are using DDR to copy the volumes (ECKD or FBA), complete this step. Otherwise, go to step "5" on page 432.

Enter the following sequence of commands for each source volume. Answer `yes` to all prompts.

```
ddr
sysprint cons
input sourcevol dasd
output targetvol dasd
copy all
```

When you have copied all the volumes, go to step "6" on page 432.

Note: If your target volume is larger than your source volume, the allocation map for the target volume might not be correct after the copy is complete. You can use the ICKDSF command to verify and correct the volume allocation if necessary.

5. If you are using FlashCopy to copy the volumes (FICON or ECKD only), complete this step.

   Enter the following command for each source volume:

   ```
   flashcopy sourcevol 0 end targetvol 0 end
   ```

   Note: FlashCopy can be used only to copy between volumes on the same DASD control unit. If a target volume is larger than the corresponding source volume, you must specify the real start and end cylinders for the target volume. For example, if source volume 1234 is a 3390 Model 3 (3339 cylinders) and target volume 2234 is a 3390 Model 9 (10017 cylinders), you would enter the following command:

   ```
   flashcopy 1234 0 end 2234 0 3338
   ```

   Also note that if your target volume is larger than your source volume, the allocation map for the target volume might not be correct after the copy is complete. You can use the ICKDSF command to verify and correct the volume allocation if necessary.

6. Rewrite the IPL record on the target RES volume to update the PDVOL parameter to specify the address of the target volume that contains the SYSTEM CONFIG file for the target system.

   - On a system that was installed on ECKD DASD, the SYSTEM CONFIG file is on the volume with default label VMCOM1.
   - On a system that was installed on SCSI volumes, the SYSTEM CONFIG file is on the volume with default label M01RES (the RES volume).

   Determine the target volume address you need and use the Stand-Alone Program Loader Creation Utility (SALIPL) to install a new copy of the Stand-Alone Program Loader (SAPL) with the updated target PDVOL address.

   - For a system on ECKD DASD, enter the following command:

```
salipl target_RES_addr (extent 1 iplparms fn=system ft=config pdnum=1 pdvol=target_PDVOL_addr
```

   - For a system using SCSI volumes, enter the following command:

```
salipl target_RES_addr (extent 1 iplparms fn=system ft=config pdnum=4 pdvol=target_PDVOL_addr
```

   Note: The SALIPL commands shown above do not include any customized IPL parameters that you might need. You should adjust the command to suit your environment. For more information about the SALIPL utility, see *z/VM: CP Commands and Utilities Reference*. For information about using SALIPL and passing IPL parameters, see *z/VM: System Operation*.

7. Detach the source volumes. Issue the following command for each volume:

   ```
   detach sourcevol
   ```

8. IPL the target system second level to make sure it will come up. Enter the following sequence of commands:

```
system clear
term conmode 3270
q cons
```

Note the console address. If the address is not 20, redefine it:

```
define conaddr 20
```

IPL the system:

```
ipl target_RES_addr clear loadparm 20
```

Check the SAPL screen to make sure the IPL parameters are correct. Add `cons=20` to the IPL parameters and press F10 to complete the load. By default, you will be running on the OPERATOR user ID.

9. Disconnect from the OPERATOR user ID and log on to the MAINT*vrm* user ID.

Changing the Labels on the Target DASD Volumes

To perform the steps in this procedure, the target system should be IPLed second level on the running system.

1. On the target system, log on to the MAINT*vrm* user ID.
2. Select and write down the labels you will use for the target volumes. The label for the target RES volume will not be changed at this time, but you can select a label now.
3. Update the USER DIRECT and SYSTEM CONFIG files to change the source volume labels (except the source RES volume label) to the corresponding target volume labels. The following steps assume that you are not using a directory manager product and the USER DIRECT file is on the PMAINT 2CC minidisk (the default location).

   a. By default, MAINT*vrm* links to the PMAINT 2CC minidisk as 2CC. Access the 2CC minidisk as file mode C:

      ```
      access 2cc c
      ```

   b. Make a backup copy of the USER DIRECT file. XEDIT the USER DIRECT file and change the source volume labels (except the source RES volume label) to the target volume labels. Save the updated USER DIRECT file.

      Note: The XEDIT CHANGE command can help you make global changes to the directory file.

   c. Use the CMS DIRMAP utility or the CP DISKMAP utility to create a directory map file on your A-disk. Review the map file to make sure that you have entries only for target volume labels (and the source RES volume label).

   d. Link and access the PMAINT CF0 minidisk in write mode. This is the default location of the SYSTEM CONFIG file.

      ```
      link to pmaint cf0 as cf0 w
      access cf0 fm
      ```

   e. Make a backup copy of the SYSTEM CONFIG file. XEDIT the SYSTEM CONFIG file and change the source volume labels (except the source RES volume label) to the target volume labels. Save the updated SYSTEM CONFIG file.

   f. Run the CPSYNTAX utility to validate the changes you made to the SYSTEM CONFIG file:

      ```
      cpsyntax system config fm
      ```

   g. Issue the DIRECTXA command to put the updated directory online:

      ```
      directxa
      ```

4. Shut down the target system:

```
shutdown
```

5. When the shutdown is complete, IPL CMS on your first level user ID.
6. While logged on to your first level system user ID, change the target volume labels (except the source RES volume label). Issue the following command for each target volume (except the target RES volume):

   ```
   cpfmtxa targetvol targetvol_label label
   ```

7. IPL the target system second level to make sure it will come up. Enter the following sequence of commands:

   ```
   system clear
   term conmode 3270
   q cons
   ```

   Note the console address. If the address is not 20, redefine it:

   ```
   define conaddr 20
   ```

   IPL the system:

   ```
   ipl target_RES_addr clear loadparm 20
   ```

   Check the SAPL screen to make sure the IPL parameters are correct. Add `cons=20` to the IPL parameters and press F10 to complete the load. You will be running on the OPERATOR user ID.

   When prompted for the type of start, enter `FORCE` instead of entering `WARM` or just pressing Enter. You should not lose any spool files. The IPL should complete normally.
8. Disconnect from the OPERATOR user ID and log on to the MAINT*vrm* user ID.

Changing the Label for the Target RES Volume

1. Select a new label for the target RES volume if you have not already done so.
2. Update the USER DIRECT and SYSTEM CONFIG files to change the source RES volume label to the target RES volume label.
   a. Access the PMAINT 2CC minidisk (already linked) as file mode C:

      ```
      access 2cc c
      ```

   b. Make a backup copy of the USER DIRECT file. XEDIT the USER DIRECT file and change all occurrences of the source RES volume label to the target RES volume label. Save the updated USER DIRECT file.
   c. Use the CMS DIRMAP utility or the CP DISKMAP utility to create a directory map file on your A-disk. Review the map file to make sure that you have entries only for target volume labels.
   d. Link and access the PMAINT CF0 minidisk in write mode:

      ```
      link to pmaint cf0 as cf0 w
      access cf0 fm
      ```

   e. Make a backup copy of the SYSTEM CONFIG file. XEDIT the SYSTEM CONFIG file and update the file with the target RES volume label. Save the updated SYSTEM CONFIG file.
   f. Run the CPSYNTAX utility to validate the changes you made to the SYSTEM CONFIG file:

      ```
      cpsyntax system config fm
      ```

3. On the **second level system**, ensure that you are logged on to the MAINT*vrm* user ID.
4. Change the volume label of the target RES volume (currently the source RES volume label) to the target RES volume label.

```
cpfmtxa 123 target_RES_label label
```

CPFMTXA will call ICKDSF to change the label for the target RES volume.

5. Issue the DIRECTXA command to put the updated directory online:

```
directxa
```

6. Shut down the target system:

```
shutdown
```

7. IPL the target system second level to make sure it will come up. Enter the following sequence of commands:

```
system clear
term conmode 3270
q cons
```

Note the console address. If the address is not 20, redefine it:

```
define conaddr 20
```

IPL the system:

```
ipl target_RES_addr clear loadparm 20
```

Check the SAPL screen to make sure the IPL parameters are correct. Add `cons=20` to the IPL parameters and press F10 to complete the load. You will be running on the OPERATOR user ID.

You should not need to enter `FORCE` this time. You should not lose any spool files. The IPL should complete normally.

8. Disconnect from the OPERATOR user ID and log on to the MAINT*vrm* user ID.

Changing the Name of the Target System

To perform the steps in this procedure, the target system should be IPLed second level on the running system.

1. On the target system, log on to the MAINT*vrm* user ID.
2. Select a name for the target system. The target system name must be 1-8 alphanumeric characters and must start with a letter. The name must not contain any blanks and must not be NOSYS or NOSSI.
3. Update the System-Level Product Inventory Table (VM SYSPINV file) with the target system name.
   a. Link to and access the PMAINT 41D minidisk in write mode:

   ```
   link to pmaint 41d as 41d w
   access 41d fm
   ```

   b. Make a backup copy of the VM SYSPINV file. XEDIT the VM SYSPINV file using the WIDTH option to avoid accidentally truncating data in the file:

   ```
   xedit vm syspinv (width 120
   ```

   Change all occurrences of the source system name (on the :SYSTEM. tag) to the target system name. Save the updated VM SYSPINV file.

   c. Do not release the PMAINT 41D minidisk. There are additional files on the PMAINT 41D minidisk that must be updated.
4. Update the SYSTEM CONFIG file with the target system name.
   a. Link to and access the PMAINT CF0 minidisk in write mode:

```
link to pmaint cf0 as cf0 w
access cf0 fm
```

   b. Make a backup copy of the SYSTEM CONFIG file. XEDIT the SYSTEM CONFIG file and change all occurrences of the source system name to the target system name. Save the updated SYSTEM CONFIG file.
   c. Run the CPSYNTAX utility to validate the changes you made to the SYSTEM CONFIG file:

```
cpsyntax system config fm
```

5. Update the VMSES/E service inventory files with the target system name.
   a. Access the MAINT*vrm* 51D minidisk.
   b. Make a backup copy of the System-Level Service Update Table (VM SYSSUF file). XEDIT the VM SYSSUF file using the WIDTH option to avoid accidentally truncating data in the file:

```
xedit vm syssuf (width 120
```

   Change all occurrences of the source system name (on the :PRODLEV. tag) to the target system name. Save the updated VM SYSSUF file.
   c. Locate and make backup copies of all the Service-Level Production Status Tables (files with a file type of SRVPROD) on the MAINT*vrm* 51D and PMAINT 41D minidisks. XEDIT each SRVPROD file using the WIDTH option to avoid truncating data in the file:

```
xedit compname srvprod (width 120
```

   Change all occurrences of the source system name (on the :STAT. tag) to the target system name. Save the updated SRVPROD file.
6. Log off the MAINT*vrm* user ID.
7. Update the recovery SFS server with the target system name.
   a. Log on to the recovery SFS server, VMSERVR. The default password is WD5JU8QP.

   To stop the server, enter:

```
STOP
```

   b. XEDIT the VMSERVR DMSPARMS file (located on the VMSERVR 191 minidisk). Change the LUNAME to match the target system name. Save the updated file.
   c. Update the CRR log minidisk with the target system name:

```
fileserv crrlog 306 307
```

   Note: 306 and 307 are the default minidisks for the recovery server defined in the VMSYSR POOLDEF file (the VDEVS assigned on the DDNAME=CRR*n* statements) on the VMSERVR 191 minidisk. If your recovery server is using minidisks other than the defaults, adjust the FILESERV command accordingly.
   d. Restart the server and disconnect:

```
profile
#cp disc
```

8. Log on to the MAINT*vrm* user ID.
9. Shut down the target system:

```
shutdown
```

10. When the shutdown is complete, IPL CMS on your first-level system user ID.
11. While logged on to your first-level system user ID, rewrite or remove the system ownership information on your spool and page volumes.

For each spool and page volume, enter the following command:

```
cpfmtxa vol_addr vol_label owner nossi target_system_name
```

You will be prompted to confirm the format command. When cylinder 0 is formatted, you need to reenter the allocation for the volume.

For a spool volume, enter:

```
spol 0-end
end
```

For a page volume, enter:

```
page 0-end
end
```

12. IPL the target system second-level to make sure it will come up. Enter the following sequence of commands:

    ```
    system clear
    term conmode 3270
    q cons
    ```

    Note the console address. If the address is not 20, redefine it:

    ```
    define conaddr 20
    ```

    IPL the system:

    ```
    ipl target_RES_addr clear loadparm 20
    ```

    Check the SAPL screen to make sure the IPL parameters are correct. Add `cons=20` to the IPL parameters and press F10 to complete the load. You will be running on the OPERATOR user ID.
13. You are now ready to start using the target system. You can shut it down if you intend to bring it up in another location, such as its own LPAR or another user ID. Otherwise, disconnect from the OPERATOR user ID and log on to the MAINT*vrm* user ID.

Additional Considerations

You might need to address the following additional items, depending on the configuration of the source system.

- SYSTEM NETID.

  If you have updated the SYSTEM NETID file on the MAINT 190 minidisk with the name of the source system, you need to change the file on the target system. Then re-save the CMS named saved system on the target system:

  1. Log on to the MAINT*vrm* user ID on the target system.
  2. Link to the MAINT 190 minidisk in multi-write mode and access the minidisk at a file mode other than S:

     ```
     link maint 190 190 mr
     access 190 t
     ```

  3. Edit the SYSTEM NETID file on the 190 minidisk to update it with the new system name, then save the file.

     ```
     xedit system netid t
     ```

  4. Issue the PUT2PROD command to re-save CMS:

```
put2prod savecms
```

5. Recycle any servers that need to start using the new saved system.

- Directory Manager.

  If you have configured a directory manager on the source system, you might need to make changes to that configuration on the target system. The directory manager should not be running on the source system or the target system while you are performing any of the renaming procedures. You will probably find it easier to update the user directory as a flat file rather than through your directory manager interface. Consult the documentation for your directory manager for information on how to do that.

- Security Manager.

  If you have configured a security manager on the source system, you might need to make changes to that configuration on the target system. The security manager should not be running on the source system or the target system while you are performing any of the renaming procedures.

  Note: If you are using RACF/VM and it was configured on your source system, there are no changes needed for RACF/VM on the target system associated with the change to the system name. RACFVM should start normally when you IPL your target system.

- Other configured products.

  If you have installed and configured other products on the source system (such as TCP/IP, Performance Toolkit, and RSCS), you might need to make changes to those configurations on the target system. The other products should not be running on the source system or the target system while you are performing any of the renaming procedures.

- Other updates for your SYSTEM CONFIG and LOGO CONFIG files.

  In addition to the changes you have already made for your new volume labels and your new system name, there are a number of other statements in the SYSTEM CONFIG file for your target system that might need to be updated before you can run your target system in its intended environment. Some of the statements you should review for correctness include:

  - ACTIVATE ISLINK
  - CU
  - DEFINE VSWITCH
  - DEVICES
  - EDEVICE
  - EMERGENCY_MESSAGE_CONSOLES
  - IODF
  - OPERATOR_CONSOLES
  - VMLAN MACPREFIX
  - VMLAN MACIDRANGE
  - Any statement that refers to a volume ID, such as CP_OWNED and USER_VOLUME_LIST

  Also review the statements in the LOGO CONFIG file.

  For more information on the configuration statements, see Chapter 6, “The System Configuration File,” on page 47 and Chapter 7, “The Logo Configuration File,” on page 337.

Chapter 17. Using PCIe Functions for z/VM Guests

This chapter describes:

- The location of industry standards for PCI functions
- System programmer tasks
- Debug aids
- Recovery from errors.

PCI Industry Standards

PCI functions are IO devices that conform to the PCI bus specification. This specification is controlled by the PCI-SIG. For more information, see PCI SIG (www.pcisig.com).

z/VM Support

In some ways, PCIe function support on z/VM is similar to channel attached devices such as DASD. This includes defining the function in the IOCDS and the VARY, ATTACH, and QUERY commands. For more information about these commands, see *z/VM: CP Commands and Utilities Reference*.

Guests with a dedicated PCI function are not eligible for Live Guest Relocation. In addition, if in an SSI environment, a guest will only be able to use PCIe functions if they are in a singleton domain or a domain where the appropriate PCIe facilities hardware is available on all members in the domain and PCI is enabled via the FEATURES ENABLE PCI statement in the system configuration file on all members in the domain. For additional details, see "Using Relocation Domains" on page 754.

A list of PCI function supported by z/VM can be found in *z/VM: General Information*.

Steps to Using a PCIe Function on VM

The following should be done on a test system first. This will allow tuning the system configuration file statements. Refer to *z/VM: CP Commands and Utilities Reference*, for more information on the listed commands.

1. Plug in the PCIe function cards.
2. Define the PCIe functions in the IO gen, either dynamically or via HCD/IOCDS. See "PCIe Functions Defined in IOCP" on page 440.
3. Enable PCI support in the system configuration file. See "FEATURES Statement" on page 158.
4. Define the IOAT subpool in the system configuration file. See "STORAGE Statement" on page 278.
5. Define storage locking warning and fail percentages in the system configuration file using the STORAGE statement.
6. IPL the VM system.
7. Verify the expected PCI functions are available with the QUERY PCIFUNCTION command.
8. Verify the IOAT subpool settings with the QUERY FRAMES command.
9. Verify the storage locking and warning settings with the QUERY FRAMES command.
10. Log on the guests that are using PCIe functions
11. Attach PCIe functions to guests with the ATTACH command. These can be included in the user directory. Note these attach commands may be restricted based on the PCIe function's UID value based on the settings of the SET IO_OPT UID command.
12. Verify the PCIe functions are attached to the guests with the QUERY VIRTUAL PCIFUNCTION command.

13. IPL the guest operating system. For more information, see the IPL command.
14. Start the applications in the guest operating system that are using the PCIe functions.
15. Watch the usage of the IOAT subpool and the locked frame percentage. This is done with the QUERY FRAMES command and PERFKIT. For more information, see *z/VM: CP Commands and Utilities Reference* and *z/VM: Performance Toolkit Reference*.
16. The IOAT usage will increase as a guest begins using a function. The usage for that function will level out and hold until the tables are reregistered. If, after putting another function into use, the IOAT subpool is 100% used, then the customer will need to increase the subpool size to enable the long term use of that number of functions with those guests.
17. Adjust the storage locking warning and fail percentages set in step "5" on page 439 based on usage. See "Protecting VM from Excessive Memory Use" on page 440 for guidelines. Tell the VM lab what the results of your testing are so that publications can be improved.

PCIe Functions Defined in IOCP

PCI functions are dedicated reconfigurable entities. This means that a PCI function can only be configured to one system at a time, but multiple systems can be specified as having potential access to the PCI function. In the IOCP one or no systems can be specified as having initial access to the PCI function and one or multiple systems can be specified as having access to the PCI function (Refer to Input/Output Configuration Program User's Guide for ICP IOCP). If a system is specified as having initial access, then that system will see that PCI function in the configured (online) state (see QUERY PCIFUNCTION command) while the other systems will not see the PCI function at all. If there are no systems specified as having initial access, then all the systems in the access list will see the PCI function in the standby (offline) state (see QUERY PCIFUNCTION command). In this case, the first system that varies the PCI function online (see VARY PCIFUNCTION command) will see it in the configured state, while all other systems will no longer be able to see the PCI function. If the first system varies off that PCI function, then all system with access to that PCI function will now see it in standby (offline) state.

Refer to *z/VM: CP Commands and Utilities Reference* for more information on these commands.

Dynamic I/O and PCI Functions

PCI function's can be dynamically defined/modified/deleted via CP's dynamic I/O commands:

- DEFINE PCIFUNCTION
- MODIFY PCIFUNCTION
- DELETE PCIFUNCTION

Refer to *z/VM: CP Commands and Utilities Reference* for information on these commands and *z/VM: I/O Configuration* for general dynamic I/O information.

You can also use zVM HCD/HCM support to dynamically define/modify/delete PCI functions. For more information, see *z/VM: I/O Configuration* and z/OS and z/VM: Hardware Configuration Manager User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/eequ100_v2r5.pdf).

Note: In order not to lose dynamic I/O changes, your IOCP should be updated with the dynamic changes that were made, see *IOCP User's Guide for ICP, IOCP, IBM System z*. The new IOCP should be made active via the SET IOCDS active command.

Protecting VM from Excessive Memory Use

PCIe functions are very different from other of z/Architecture IO devices. Data is transferred directly from the function to guest memory using Dynamic Memory Access (DMA). The application code and device driver running in the guest control how data is transferred; z/VM has no control.

The number of active PCIe functions will have to be limited to maintain acceptable system operation. This will depend on the type of function and the system load. Every installation will have to experiment to determine how best to tune the system.

When a large amount of storage is locked for PCIe data transfer, the size of the Dynamic Paging Area (DPA) is reduced, resulting in less real storage available for use by other guests. You should take this reduction in DPA into account when calculating the total system workload capacity and storage over-commitment targets. A small DPA can negatively affect overall system performance and possibly result in a system outage when insufficient memory is available for system use.

If your system is storage over-committed, it may also be useful to have a SET RESERVED value in effect for the guests using the PCIe functions, particularly during initialization of the PCIe application. The SET RESERVED command is used to specify the amount of storage a guest is to have resident. The specific guest pages contained within its reserved frame allotment is dependent on the reference pattern of the guest. If the pages to be locked by the PCIe application are not resident, they must be paged in during the locking operation. This can result in a noticeable delay. Having a reserved setting may improve the performance of the lock operation if the pages are already resident in real storage. However, because SET RESERVED restricts the amount of frames that can be stolen from a guest by CP's frame replenishment algorithm, overuse of the command can also negatively affect overall system performance. For more information, see *z/VM: CP Commands and Utilities Reference*.

Another tuning task is to determine the fail percent to specify in the system configuration file. If it is set too high, the response time for guests will be affected too much or the system could abend. If too low, guest PCIe function use will be stopped when it was not necessary. It may be appropriate to deliberately cause problems on a test system as part of this process. QUERY FRAMES or PERFKIT can be used to watch how much memory is locked. For more information, see *z/VM: CP Commands and Utilities Reference* and *z/VM: Performance Toolkit Reference*.

After the fail percent is determined, PCIe function use should be added in a controlled way, watching the amount of memory locked. This will determine the number of functions that VM can support.

The warning percent should be set low enough to allow preventive action, such as stopping some low priority work, when the percentage of locked pages is getting too high.

Recovering from Permanent Errors on PCIe Function

If a PCI function goes to permanent error state, an attempt to recover can be made. A class G user can use the RESET command. See RESET PCIFUNCTION, for more details. A class B user can use the VARY PCIFUNCTION and SET PCIFUNCTION RESETcommands. See VARY PCIFUNCTION command and SET PCIFUNCTION. If RESET, VARY, and SET PCIFUNTION cannot recover the PCI function, there is probably a hardware problem that must be corrected.

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

PCIe Debugging Aids

There are several PCI debug aids that are provided to help IBM service resolve problems. IBM instructs customers how to gather traces and interprets the results.

The SET CPTRACE command and the TRSOURCE ID command with TYPE PCI operand control what is traced. The SET PCIFUNCTION command with REPORT operand reports an error with a PCI function to the Support Element. See SET CPTRACE command, TRSOURCE TYPE PCI, and SET PCIFUNCTION in *z/VM: CP Commands and Utilities Reference*.

The ZPCIMON EXEC utility is provided as a sample program to monitor the health of NVMe devices that are connected by using a PCIe function. See "The ZPCIMON EXEC Utility" on page 704.

Chapter 18. Multiple Subchannel Set Support

z/VM provides limited support for multiple subchannel sets. In addition to supporting subchannel set zero, z/VM allows DASD in subchannel sets 1, 2 and 3 to be included in the configuration. These DASD must have corresponding devices in subchannel set zero in order to be recognized. All non-DASD devices must reside in subchannel set zero.

The DASD in the IPLed subchannel set, which is determined from where the Stand-Alone Program Loader (SAPL) loads the z/VM Control Program from, are referred to as the active configuration. These are the DASD currently in use by z/VM. The DASD in the remaining subchannel set(s) are referred to as the standby configuration. The subchannel set reflecting the active configuration will change when a HYPERSWAP occurs.

Command Affected

Most z/VM commands support only four-digit device numbers and can refer only to devices in the active configuration. The following commands and utilities accept or report five-digit device numbers and can act upon devices in either the active or standby configuration:

- DEFINE EDEVICE
- DELETE RDEVICE
- HYPERSWAP
- IOEXPLOR
- LOCATE RDEV
- QUERY DASD DETAILS
- QUERY EDEVICE
- QUERY EQID
- QUERY HYPERSWAP
- QUERY PATHS
- QUERY Real Device
- SET EDEVICE
- SET RDEVICE
- VARY PATH
- VARY Real Device
- VARY SUBCHANNEL

The following commands and utilities were also updated for multiple subchannel set support:

- CPSYNTAX
- QUERY CHPID
- QUERY MSS
- SALIPL (small documentation change only)

For information about commands, see *z/VM: CP Commands and Utilities Reference*.

Active versus Standby Configuration

The set of accessible devices currently available for use by the z/VM system and virtual machines is known as the *active* configuration. The set of accessible devices not currently available for use is known as the *standby* configuration.

If the system was IPLed from a device in subchannel set 0, all devices in that subchannel set would be part of the active configuration while devices in all other subchannel set(s) would be part of the standby configuration. If a HYPERSWAP command was used to swap some devices in subchannel set 0 with their counterparts in subchannel set 1, the swapped subchannel set 0 devices would become part of the standby configuration and the corresponding subchannel set 1 devices would become part of the active configuration.

If the system was IPLed from a device in subchannel set 2, all DASD devices from subchannel set 2 plus all other non-DASD devices from subchannel set 0 will become part of the active configuration. In this case, LOCATE RDEV 1234 would lead to the same result as LOCATE RDEV 21234, assuming 1234 is a DASD device. Because device number 1234 is in the active configuration, the 4-digit value is physically mapped by device number 21234, device number 1234 in subchannel set 2 which is the IPLed subchannel set.

The active subchannel set can be controlled by changing the device from which SAPL loads the CP nucleus to a device in a different subchannel set.

- On the Stand-Alone Program Loader Creation Utility (SALIPL) menu screen
- On the command line of the SALIPL CP utility
- On the SAPL screen

For information about running the Stand-Alone Program Loader Creation Utility and using SAPL, see *z/VM: System Operation*. For information about the SALIPL utility, see *z/VM: CP Commands and Utilities Reference*.

PAV/HyperPAV alias Support and Hyperswap

PAV and HyperPAV alias devices that correspond to a base device in the standby configuration can reside in the same subchannel set as the base device in the active configuration subchannel set.

When a HYPERSWAP process occurs, there are several actions which take place adjusting the placement of devices in the active configuration. During the swap command when swapping to target devices in the standby configuration, those target devices take the place of the source device (since they have the same 4-digit RDEV number) in the active configuration. While swapping PAV or HyperPAV base devices between subchannel sets, aliases associated with and in the same real subchannel set as the target of the swap are also brought into the active configuration. Being brought into the active configuration allows these devices to then be referenced by 4-digit RDEV numbers in various commands which only affect active configuration devices.

Adding Devices to a Subchannel Set

Initially z/VM considers any dynamic device addition to the I/O configuration to affect and to be reflected in the current active configuration. The newly defined device will be placed into the active configuration, regardless of its subchannel set, only if no device with the same 4-digit device number is already there. If there already is an active device with the same 4-digit number in the I/O configuration, then the new device is added to the stand-by configuration in an offline state.

If you are considering adding sets of devices in multiple subchannel sets with the same 4-digit device numbers, then it is recommended that two separate dynamic changes be made. First dynamically adding the devices desired to be in the active configuration and then subsequently adding the rest of the devices in a separate dynamic change.

If devices with the same 4-digit device number already exist in the active configuration and you don't want them there, then a separate dynamic I/O configuration change will be required first to remove those devices.

SSI Considerations

z/VM first introduced multiple subchannel set support in z/VM 6.3. Although that original support has been expanded upon in z/VM 7.2, it is possible to run an SSI where the VM member LPARs are at different release levels. However, there are some things to keep in mind if exploiting the multiple subchannel set support, especially when migrating from the older pre-7.2 to current support. In general, multi-release SSIs are supported when zero or a single alternate subchannel set is exploited. If a client needs to exploit two or more alternate subchannel sets, all VM members of the SSI should be at z/VM 7.2, or later. However, a staged migration is supported and highly recommended; meaning taking all SSI members to z/VM 7.2 at once is not required.

Clients that already exploit only a single alternate subchannel set have very little change, except to upgrade their xDR proxy software, if running GDPS, and be cognizant of the real subchannel set value required on z/VM commands supporting 5 digits, especially following a HYPERSWAP. For clients that require two or more alternate subchannel sets, here are the recommendations in order to perform a staged migration from a pre-7.2 z/VM release to z/VM 7.2, or later, by GDPS scenario:

- No GDPS nor need for alternate subchannel set support
  - Again, no related considerations.
- No GDPS but already exploit a pre-7.2 alternate subchannel set
  - Nothing prevents a staged approach to bring up only one or more z/VM 7.2 LPARs in an SSI but be aware that the high order 5th digit on any z/VM commands (on the z/VM 7.2 LPAR) will represent the real SS value (versus the logical value associated with pre-7.2 support).
  - Assumes no real device configuration changes take place with the initial z/VM 7.2 bring up.
  - Once real device changes take place, such as adding the second alternate subchannel set, clients should upgrade all SSI members to z/VM 7.2.
- GDPS and already exploit a pre-7.2 alternate subchannel set
  - The xDR proxy virtual machine must be upgraded to XDRFP53 after upgrading to SA MP Fixpack level 4.1.0.5. This is necessary on all z/VM LPARs in the SSI before adding a 7.2 VM member. This includes the software that will run on the 7.2 member. These software upgrades contain the GDPS "compatibility" code to support z/VM 7.2 and to allow mixed levels of VM in the SSI. These upgrades are available on IBM Fix Central (https://www.ibm.com/support/fixcentral).
  - z/VM 7.2 members can now be brought in the SSI, again make sure the xDR proxy code is upgraded, but be aware that the high-order 5th digit on any VM commands (on the z/VM 7.2 LPAR) will represent the real SS value (versus the logical value associated with pre-7.2 support).
  - Assumes NO real device configuration changes affecting the alternate subchannel set topology take place with the initial z/VM 7.2 bring up. Specifically, these would be changes that the pre-7.2 members do not support.
  - Once real device changes are necessary, such as adding a second alternate subchannel set, clients need to upgrade all SSI members to z/VM 7.2, especially as GDPS expects a consistent real device topology across all GDPS-managed z/VM LPARs.

EDEVICE Support

The SET EDEVICE command now allows for creation of 5-digit EDEVICEs. This is for standby secondary devices only and has been added for the GDPS support providing site failover. HYPERSWAP is not supported for EDEVICEs.

Restrictions

1. The DEVICES settings, defined by the DEVICES system configuration statement or the SET DEVICES command, apply to all subchannel sets. For example, if device 1234 or 01234 is NOTACCEPTED, then so are devices 11234. 21234 and 31234. If device 01234 is NOTACCEPTED and devices 11234, 21234 or 31234 are added dynamically, they will be ignored.

Part 3. User Planning and Administration

Chapter 19. Redefining Command Privilege Classes

You can extend 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.

IBM-Defined Privilege Classes

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 system accepts commands.

In general, the system programmer who creates the user directory assigns each user one or more privilege classes as part of the virtual machine definition in that directory.

Privilege classes are denoted by the letters A through Z, the numbers 1 through 6, or the word "Any". These classes, and the type of user who uses the commands belonging to each privilege class set, are summarized in Table 19 on page 449. Classes I through Z and numbers 1 through 6 are reserved so that your installation can define them to suit its needs.

Users whose password is NOLOG have no privilege class and can only receive spooled output as punched cards or printed forms. The NOLOG assignment controlled by directory statements; see “USER Directory Statement” on page 616 or “IDENTITY Directory Statement” on page 510 for more information about NOLOG users.

Table 19. IBM-Defined Privilege Classes

Class	User and Function
A	*System Operator:* The class A user controls the z/VM system. The system operator is responsible for the availability of the z/VM system and its resources. In addition, the system operator controls system accounting, broadcast messages, virtual machine performance options, and other options that affect the overall performance of z/VM. **Note:** The class A user who is automatically logged on during CP initialization is designated as the primary system operator.
B	*System Resource Operator:* The class B user controls all the real resources of the z/VM system, except those controlled by the system operator and the spooling operator.
C	*System Programmer:* The class C user updates or changes system-wide parameters of the z/VM system.
D	*Spooling Operator:* The class D user controls spool files and the system's real reader, printer, and punch equipment allocated to spooling use.
E	*System Analyst:* The class E user examines and saves system operation data in specified z/VM storage areas.
F	*Service Representative:* The class F user obtains, and examines in detail, data about input and output devices connected to the z/VM system. This privilege class is reserved for IBM use only.
G	*General User:* The class G user controls functions associated with a particular virtual machine.
Any	Commands belonging to class "Any" are available to any user, regardless of the user's privilege class. These commands are primarily those used to gain access to, or relinquish access from, the z/VM system.

Table 19. IBM-Defined Privilege Classes (continued)

Class	User and Function
H	Reserved for IBM use.
I - Z 1 - 6	These classes are reserved for redefinition by each installation for its own use (using MODIFY statements or commands).

Planning a New User Class Structure

While planning the new class structure, you should:

- Define each user's needs
- Assign commands to types of users
- Associate privilege classes with users and commands

Defining Users' Needs

The first step in restructuring command classes is to determine the different types of users and what types of functions each user needs to perform.

Consider a fictional insurance company with a variety of users and job responsibilities. The installation has decided to define a new class structure. The system administrator examines the users' needs and produces the categories as shown in Table 20 on page 450.

Table 20. A Sample Scenario: Different System Users and Their Responsibilities

Job Title	Abbreviation	Duties
System administrator	SAD	General management of the system and determining how the system will be structured and used.
Senior system programmers	SSP	Planning, generating, maintaining, extending, and controlling the use of the operating system with the aim of improving the general productivity of the installation.
Junior system programmers	JSP	Similar to senior system programmers, but with less control over the system.
System analysts	SA	Analyzing the system to determine what new applications, system programs, and devices are needed by the installation.
Primary system operators	SO	Ensuring the smooth running of the system and carrying out such duties as changing tapes and disk packs.
Database administrator	DBA	Administering resources associated with, and having access to, the main database of the system, and resources associated with spooling, printing, and archiving.
Service engineers	SE	Obtaining and examining certain data about I/O devices connected to the system. Also, controlling intensive error recording and some machine check error recording.
General users	U1, U2	Two different types of general users with different requirements were identified, the first more sophisticated than the second. (All users, even if classified above, will also be classified as one type of general user.)

Assigning Commands to Types of Users

After you have produced these categories, you need to list all the commands, DIAGNOSE codes, and system functions that each user group will need to execute.

In our insurance company example, the system administrator produces a list of commands with the user types listed across the top. (For the sake of brevity, the table presented here lists only a few commands. In reality, all CP commands, DIAGNOSE codes, and system functions should be listed, except those with an IBM-defined class of ANY.) After you have produced this list, you need to identify which commands you want each user type to be able to access. For our example, an asterisk (*) is placed in the column for each user that should have access to the command in the first column. See Table 21 on page 451.

Table 21. Example of How to Assign Commands, Diagnose Codes, and System Functions to Types of Users

Command	**IBM-Defined Class**	**New Class**	**SAD**	**SSP**	**JSP**	**SA**	**SO**	**DBA**	**SE**	**U1**	**U2**
ACNT	A						*				
ATTN	G									*	*
XAUTOLOG	A		*	*							
XAUTOLOG	B				*	*	*	*			
XAUTOLOG	G									*	
CHANGE	D						*	*			
CHANGE	G									*	*
DEDICATE	A			*							
DEFINE	A						*				
DEFINE	G									*	*
IPL	G									*	*
MESSAGE	A			*	*						
MESSAGE	B						*				
SAVESYS	E			*	*						
SPOOL	G									*	*
DIAG04	C,E		*	*	*		*				
DIAG3C	A,B,C		*	*	*	*					

Notes:

1. Do not list commands with an IBM-defined class ANY. You cannot limit access to these commands.
2. Some commands have more than one IBM-defined class. For a list of these commands, see *z/VM: CP Commands and Utilities Reference*. You need to list these separately so that you can assign them to different user groups.
3. Restricting the user class on a console command does not restrict the function of the directory statement that does the same function. For example, if there is a LINK directory statement in a user's directory, that statement takes effect at logon time, even though you have restricted the LINK command.

Associating Privilege Classes with Users and Commands

After you have associated the commands with the types of users, you can assign a different class to each type of user. Then, each command can be assigned a list of classes that correspond to the users who need access. In the example table, each asterisk is replaced by a new user class, and then these classes are collected to the “New Class” column for each command. See Table 22 on page 452.

Table 22. Example of How to Associate Privilege Classes with Commands and Users

Command	IBM-Defined Class	New Class	SAD I	SSP J	JSP K	SA L	SO M	DBA N	SE O	U1 P	U2 Q
ACNT	A	L				L					
ATTN	G	PQ								P	Q
XAUTOLOG	A	IJ	I	J							
XAUTOLOG	B	KLMN			K	L	M	N			
XAUTOLOG	G	P								P	
CHANGE	D	MN					M	N			
CHANGE	G	PQ								P	Q
DEDICATE	A	J		J							
DEFINE	A	M					M				
DEFINE	G	PQ								P	Q
IPL	G	PQ								P	Q
MESSAGE	A	JK		J	K						
MESSAGE	B	M					M				
SAVESYS	E	IJ	I	J							
SPOOL	G	PQ								P	Q
DIAG04	C,E	IJKM	I	J	K		M				
DIAG3C	A,B,C	IJKL	I	J	K	L					

An important distinction to make here is that the user classes with access to system functions and resources (classes I through O) do not have access to any commands that are useful for controlling their own virtual machines (for example, SPOOL). Only the two general user classes (P and Q) have access to these commands. Class P users have more access to more powerful commands than do class Q users. With this arrangement, the system administrator can independently control a user's access to system and virtual machine commands. To assign classes to each user, the system administrator defines at least two classes to key sophisticated users such as system programmers. An unsophisticated general user might be assigned to class Q; a system programmer would be assigned to classes J and P. In this way, the system programmer gains access to both the class J system commands and to the class P virtual machine commands.

Note also that whereas the class A and B MESSAGE commands are listed, the class ANY MESSAGE command is not. User class modification does not affect class ANY commands.

Further Considerations

While customizing the command privilege class structure of your system, you should keep in mind:

Help Files: You may also want to update the HELP files if changes to the command classes affect a class G command. See *z/VM: CMS User's Guide* for information on tailoring the HELP Facility.

Documentation Considerations: If you change the privilege class for commands, DIAGNOSE codes, or system functions, the privilege classes documented in this and other publications may no longer be correct for your installation.

How to Modify a Command or Diagnose Code

You can alter the following attributes of a CP command or a Diagnose code:

- Privilege class, or PRIVCLASSANY
- Name of the routine used to process the command or DIAGNOSE code

To redefine commands and DIAGNOSE codes during system initialization, use the MODIFY COMMAND (or MODIFY CMD) and MODIFY DIAGNOSE statements in the system configuration file. For more information, see "MODIFY COMMAND / CMD Statement" on page 188 and "MODIFY DIAGNOSE Statement" on page 192.

To redefine commands and DIAGNOSE codes after initialization, use the MODIFY COMMAND (or MODIFY CMD) and MODIFY DIAGNOSE commands. For information about these commands, see *z/VM: CP Commands and Utilities Reference*.

Modifying a Command

This section shows examples of using the MODIFY COMMAND command to change the attributes of a CP command.

If multiple versions of a command or subcommand are defined but only a single version is to be altered, then the IBMCLASS keyword is required to select the appropriate version.

Suppose that you wish to alter the IBM class A version of the MESSAGE command and change it to PRIVCLASS G. Because there are 3 versions of the MESSAGE command (IBM class 0 for the PRIVCLASSANY version, IBM class A, and IBM class B), you must supply the IBMCLASS keyword.

```
CP MODIFY COMMAND MESSAGE IBM A PRIVCLASS G
Ready;
```

MODIFY has the additional property that it will remember (if it is not presently doing so) the attributes of the command that is being altered. If it is remembering the attributes for a command, it can restore those attributes by using the RESET operand, at which time it stops remembering any prior attributes.

Suppose that MODIFY were performed 3 times for the IBM class A version of the MESSAGE command, and then a MODIFY RESET was performed. After the MODIFY RESET, the command would once again be a PRIVCLASS A command. An immediate MODIFY RESET would make no changes, because the prior MODIFY RESET would have discarded any saved information.

```
* This command changes privilege class to G
CP MODIFY COMMAND MESSAGE IBM A PRIVCLASS G
Ready;

* This command changes privilege class to 123456
CP MODIFY COMMAND MESSAGE IBM A PRIVCLASS 123456
Ready;
```

```
* This command changes privilege class to ACDEILNOPY.
* Notice that the classes are allowed to be specified in any order.
CP MODIFY COMMAND MESSAGE IBM A PRIVCLASS ENCYCLOPAEDIA
Ready;

* This command changes privilege class back to A
CP MODIFY COMMAND MESSAGE IBM A RESET
Ready;

* This command makes no change because we just performed a RESET
CP MODIFY COMMAND MESSAGE IBM A RESET
Ready;
```

The name of the routine to process the command may be changed, but all versions of the command must have the same routine name.

```
CP MODIFY COMMAND MESSAGE IBM A EPNAME HCPSRC00
HCPZPM770E MODIFY is not valid for command MESSAGE because different
           EPNAMEs would be set
Ready(00770);

CP MODIFY COMMAND MESSAGE EPNAME HCPSRC00
Ready;
```

The subcommands that make up the set of QUERY subcommands are divided into two groups and are controlled by two separate MODIFY commands. For example, QUERY DASD is a subcommand and QUERY VIRTUAL DASD is a separate subcommand. You must specify the "VIRTUAL" operand on the MODIFY command if you wish to change the QUERY VIRTUAL DASD subcommand. Because of the additional granularity that is provided by the MODIFY command, you must issue two MODIFY commands if you wish to change both the virtual and non-virtual subcommands of QUERY. For example, to change all versions of the QUERY DASD and QUERY VIRTUAL DASD subcommands so that only user privilege class 'X' can issue them, enter the following:

```
modify cmd query virtual subcmd DASD ibmclass * privclasses x
modify cmd query subcmd DASD ibmclass * privclasses x
```

The first MODIFY command changes the QUERY VIRTUAL subcommands while the second changes the QUERY subcommands.

You should also note that there are a number of commands in the system which share generic command control blocks. For example, QUERY 000E (class b version) shares the same generic command control block with QUERY L00E (class b version). The determination of the actual command being issued is not made until the command handler gets control. The generic command blocks only exist for class B and G commands. In addition, they exist on the virtual subcommand chain. To change these commands you must specify "VIRTUAL SUBCMD *" and the appropriate IBMCLASS. For example, to add a user privilege class x to the class b version, you would enter the following:

```
modify cmd query virtual subcmd * ibmclass b privclasses bx
```

Modifying a Diagnose Code

This section shows examples of using the MODIFY DIAGNOSE command to change the attributes of a DIAGNOSE code.

Because there are no multiple versions of a Diagnose code, there is no IBMCLASS keyword.

Suppose that you wish to alter the privilege class of Diagnose code X'7C' from PRIVCLASSANY to PRIVCLASS L. Because there is only one version of any Diagnose code, only the Diagnose code is specified.

```
CP MODIFY DIAGNOSE 7C PRIVCLASS L
Ready;
```

The MODIFY DIAGNOSE command has the same RESET capability that was discussed for the MODIFY COMMAND command.

The MODIFY DIAGNOSE command has the same capability to change the name of the processing routine that was discussed for the MODIFY COMMAND command. But, Diagnose code routines have an additional interface to the Diagnose code router that allows register 15 to contain a return code upon return to the Diagnose code router. If the routine that processes the Diagnose code is changed, it may be that the interface selected by the routine (whether to pass a return code or not) will be different. If so, then the CHECKR15 keyword should be used to specify which interface this routine is using.

```
   CP MODIFY DIAGNOSE 7C EPNAME HCPSRC00 CHECKR15 YES
   Ready;
```

Another Way of Changing the Privilege Class of Certain CP Functions

Privilege classes for the following CP functions can be changed without using the MODIFY statement or command:

- Logging on as the primary system operator
- Intensive error recording
- Using the read function of the CP IOCP utility
- Using the write function of the CP IOCP utility
- Specifying the default user class.

If you want to change some or all the privilege classes assigned to these CP functions, you must specify your changes by using the PRIV_CLASSES statement in the system configuration file. For more information on the PRIV_CLASSES statement, see "PRIV_CLASSES Statement" on page 223.

In our insurance company example, the classes of three system functions were changed:

- Classes for the primary system operator
- Classes authorized to perform intensive error recording
- Default classes for users who do not have a class defined in their virtual machine definitions.

Instead of including them in the MODIFY statement or command, the system administrator could have entered the PRIV_CLASSES statement as:

```
PRIV_CLASSES OPERATOR IJ HW_SERVICE L USER_DEFAULT P
```

You can also use the CP SET PRIVCLASS command to change the privilege classes for individual users, who are logged on to the system, rather than an entire class of users. This command lets you be more selective in the privileges that you grant users. For a brief description of the SET PRIVCLASS command, see "Changing the Setting for a Logged-On Virtual Machine" on page 456. For a detailed description of the SET PRIVCLASS and QUERY PRIVCLASS commands, see *z/VM: CP Commands and Utilities Reference*.

Changing the Directory

After changing the user class structure, you must update the directory, as it contains the class or classes of commands that each user can successfully execute. The directory statements in the directory that define the command privilege classes for a virtual machine are the USER and CLASS statements. The IDENTITY statement also defines the command privilege classes for a virtual machine. However, the examples do not include virtual machines defined with IDENTITY statements. (For more information about the directory and how to generate it, see Chapter 20, "Creating and Updating a User Directory," on page 459.) Use the privilege class operand of the USER statement to define the new privilege classes for the virtual machine.

If you cannot fit all the new privilege classes onto the USER statement, you can add the optional CLASS statement to the directory immediately following the USER statement. You can define up to 32 classes.

Note: Make sure you have enough free disk space before editing and making changes to the existing directory so that you can file the updated directory. See "Directory Space" on page 652 for information on how to allocate DASD space for the directory.

Defining Privilege Classes for a Virtual Machine

The privilege classes for a virtual machine can be changed by changing the definition of a logged on virtual machine or changing the definition in the user directory.

Changing the Setting for a Logged-On Virtual Machine

Authorized users can change the privilege classes definitions for logged on virtual machines. These users are authorized by the FEATURES ENABLE SET_PRIVCLASS statement in the system configuration file. To change the privilege classes, the user just issues the CP SET PRIVCLASS command.

For example, if you want the user whose virtual machine user ID is DATABASE to be able to use commands with the privilege classes C, D, H, and I, you would enter:

```
set privclass database =cdhi
```

The FEATURES ENABLE SET_PRIVCLASS statement is described in "FEATURES Statement" on page 158. For more information on the SET PRIVCLASS command, see the *z/VM: CP Commands and Utilities Reference* book.

Changing the User Directory

To change the definition of privilege classes for a virtual machine, perform the following steps:

1. Use the XEDIT command to edit the user directory. Unless you have defined a different file name, this file has a file ID of USER DIRECT.
2. Find the USER directory statement for the virtual machine whose privilege classes you wish to change.
3. This next step depends on whether you can fit all the new classes onto the USER directory statement.

   If you can fit all the new classes onto the USER directory statement:

   a. Change the privilege class operand to the classes that you want this virtual machine to have. The privilege class operand consists of EBCDIC characters (with no intervening blanks) that can be A through Z and 1 through 6. These characters define privilege classes for the virtual machine. If you do not specify a privilege class, the default is G (or whatever you have specified using the USER_DEFAULT operand of the PRIV_CLASSES statement in the system configuration file).

      For example, if you want the user whose virtual machine user ID is DATABASE to be able to use commands with the privilege classes C, D, H, and I, code the USER directory statement as follows:

      ```
      USER DATABASE pass stor mstor CDHI pri le ld cd es
      ```

   If you cannot fit all the new classes onto the USER directory statement:

   a. Change the privilege class operand to an asterisk (*).
   b. Following the USER directory statement, insert a CLASS directory statement. The format of the CLASS directory statement is described under "CLASS Directory Statement" on page 482.

      For example, if you want to assign privilege classes A through Q to a virtual machine that belongs to a user whose user ID is SYSADM, code the USER directory statement and the CLASS directory statement as follows:

      ```
      USER SYSADM pass stor mstor * pri le ld cd es
      CLASS ABCDEFGHIJKLMNOPQ
      ```

4. After you make all the desired changes to the directory, file the directory file.

5. To verify that the CMS file can be used as a directory file, enter the DIRECTXA utility with the EDIT option. (For more on the DIRECTXA utility, see "Running the User Directory Program" on page 469.) If you made a syntax error, you will receive an error message.
6. When you have verified that the directory file is correct, replace the old directory with the updated directory by entering the following:

   ```
   directxa filename
   ```

   Note: The virtual machine that enters the DIRECTXA command must have write access to the volume that is to contain the new directory. If you create a directory that is to be written on the active VM/ESA system residence volume, your virtual machine's current virtual machine definition must have write access to the volume that contains the current directory.
7. After the directory is updated, directory changes for a virtual machine currently logged on to the system do not take effect until the user logs off the system and then logs back on.

Virtual Machine Definition Example

For our example of the insurance company, the directory might include the USER and CLASS directory statements shown in Figure 15 on page 457.

```
DIRECTORY  250  3390  VMSRES
* CP DIRECTORY FOR 'OUR INSURANCE COMPANY'
*
* NOTES:
* (1)    KEY TECHNICAL USERS NEED PRIVILEGE CLASS 'P' BESIDES
*        THEIR PRIMARY FUNCTIONAL CLASS.  THIS ENABLES THEM
*        TO FULLY CONTROL A VIRTUAL MACHINE.  GENERAL USERS
*        WILL NEED ONLY CLASS 'Q'.
*
* (2)    USERID ZZZMAINT IS SET ASIDE FOR EMERGENCY USE.  THIS
*        USER HAS ACCESS TO ANY CP COMMAND REGARDLESS OF THE
*        USER CLASS STRUCTURE.  THE PASSWORD SHOULD BE
*        KNOWN BY ONLY A ***VERY*** FEW KEY PEOPLE.
*
USER ZZZMAINT SECRET 2M 8M *
  CLASS ABCDEFGHIJKLMNOPQRSTUVWXYZ123456
  (other directory statements)
*
USER SAD1 XXXXXXXX 2M 8M IP
  (other directory statements)
*
USER SSP1 XXXXXXXX 1M 2M JP
  (other directory statements)
*
USER JSP1 XXXXXXXX 1M 2M KP
  (other directory statements)
*
USER JSP2 XXXXXXXX 1M 2M KP
  (other directory statements)
*
USER SA1 XXXXXXXX 1M 2M LP
  (other directory statements)
*
USER OPERATOR XXXXXXXX 1M 2M MP
  (other directory statements)
*
   .
   .
   .
  (and so on)
```

Figure 15. Virtual Machine Definition Example Using USER and CLASS Statements

How Users Can Find Which Commands They Can Enter

To find out which commands are available, the user can enter the QUERY COMMANDS command. For example, if the IBM-defined classes are in effect for all commands, a user whose virtual machine is assigned privilege class B would receive a list of all class B commands, DIAGNOSE codes, and system functions, as well as all class ANY commands.

To find out what operands of the QUERY and SET commands are available, the user can enter QUERY COMMANDS QUERY or QUERY COMMANDS SET.

Chapter 20. Creating and Updating a User Directory

This section tells you how to:

- Create the z/VM user directory
- Run the user directory program
- Change the directory
- Check the directory for errors
- Specify a directory that contains VM/SP directory statements
- Use directory profiles
- Use each of the directory statements summarized in Table 25 on page 465.

Overview of the User Directory

The z/VM user directory specifies the configuration and operating characteristics of virtual machines. A z/VM user directory exists in two forms: a source form that consists of one or more CMS files, and an object form, created from the source, on a CP-formatted disk. Multiple object directories can be accessed by the z/VM system, but only one object directory can be active (online) for a system at any given instant.

Important: When operating in an SSI cluster environment, you should use a single source directory to create the object directory for each member in the SSI cluster. Using a single source directory results in consistent set of virtual machine configurations on each system in the SSI cluster.

The source form of the user directory consists of directory statements that define the CP-formatted volume on which the object directory is created and the configuration and operating characteristics of the virtual machines that are known to the z/VM operating system. The directory statements are divided into blocks called *entries*. An entry begins with one of the following statements: GLOBALDEFS (global definitions entry), PROFILE (profile entry), USER (user entry), IDENTITY (identity entry), or SUBCONFIG (subconfiguration entry).

A *virtual machine definition* is the group of entries that define a virtual machine. A virtual machine definition can define one of two virtual machine definition types:

Single-configuration virtual machine definition

A virtual machine definition that consists of a user entry and any included profile entry. Only one virtual machine instance can be created from a single-configuration virtual machine definition. For example, you can specify a USER1 single-configuration virtual machine and log on to a z/VM system as USER1.

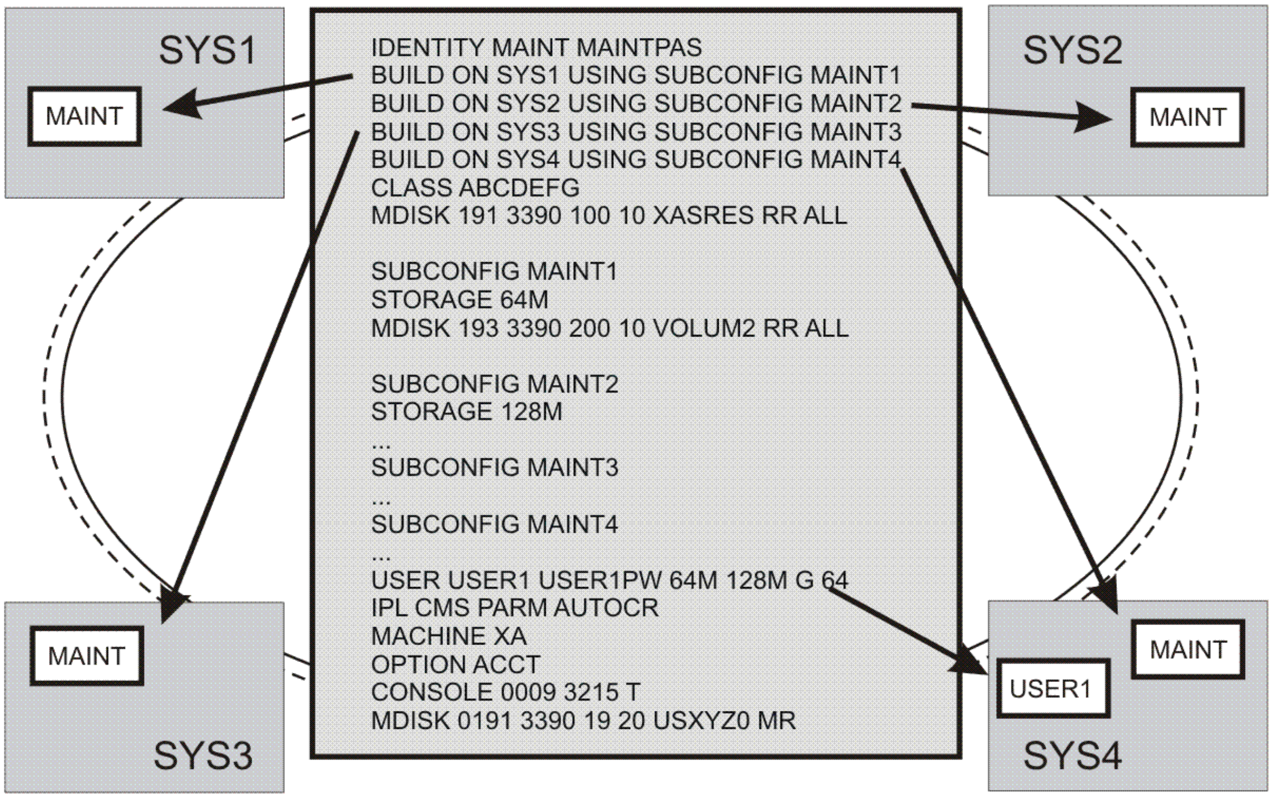
In an SSI cluster, the virtual machine can be logged on to only one SSI member at a time.

Multiconfiguration virtual machine definition

A virtual machine definition that consists of an identity entry, any included profile entry, and all associated subconfiguration entries. In an SSI-enabled source directory, this virtual machine definition allows multiple virtual machine instances to be defined, which enables the user ID to be logged on concurrently to multiple members of the SSI cluster. Each of these virtual machine instances can have a different configuration from the others. For example, you can define a MAINT multiconfiguration virtual machine and concurrently log on to all the members of an SSI cluster as MAINT.

Figure 16 on page 460 is an abbreviated view of two virtual machine definitions, one a single-configuration virtual machine and the other a multiconfiguration virtual machine. The sample source directory defines USER1 as a single-configuration virtual machine through the USER and associated directory statements, allowing USER1 to log onto one member of the SSI cluster at a time. In this case, USER1 has logged onto SYS4. The sample source directory defines MAINT as a multiconfiguration virtual machine through the IDENTITY, BUILD, SUBCONFIG, and associated directory statements, allowing

MAINT to log onto all members of the SSI cluster concurrently, using common settings from the identity entry and system-unique settings from the subconfiguration entry.



```
IDENTITY MAINT MAINTPAS
BUILD ON SYS1 USING SUBCONFIG MAINT1
BUILD ON SYS2 USING SUBCONFIG MAINT2
BUILD ON SYS3 USING SUBCONFIG MAINT3
BUILD ON SYS4 USING SUBCONFIG MAINT4
CLASS ABCDEFG
MDISK 191 3390 100 10 XASRES RR ALL

SUBCONFIG MAINT1
STORAGE 64M
MDISK 193 3390 200 10 VOLUM2 RR ALL

SUBCONFIG MAINT2
STORAGE 128M
...
SUBCONFIG MAINT3
...
SUBCONFIG MAINT4
...
USER USER1 USER1PW 64M 128M G 64
IPL CMS PARM AUTOCR
MACHINE XA
OPTION ACCT
CONSOLE 0009 3215 T
MDISK 0191 3390 19 20 USXYZ0 MR
```

Figure 16. Example single-configuration and multiconfiguration virtual machines

Source Directory Types

The source directory can be one of the following types:

Non-SSI
A source directory that does not contain the SSI option on the DIRECTORY statement or any IDENTITY, BUILD, or SUBCONFIG statements.

SSI-ready
A source directory that can contain single-configuration virtual machine definitions and multiconfiguration virtual machine definitions, but does not have the SSI option on the DIRECTORY statement. In each multiconfiguration virtual machine definition, the identity entry can contain at most one BUILD statement, on which the name of the member of the SSI cluster must be an asterisk (*). Only one virtual machine instance can be created from a multiconfiguration virtual machine definition when the object directory is created from this type of source directory.

SSI-enabled
A source directory that can contain single-configuration virtual machine definitions and multiconfiguration virtual machine definitions and that has the SSI option specified on the DIRECTORY statement. In each multiconfiguration virtual machine definition, the identity entry can have multiple BUILD statements, each of which must specify the name of a member of the SSI cluster. Virtual machine instances can be created on multiple members from a multiconfiguration virtual machine definition when the object directory is created from this type of source directory.

Table 23 on page 461 shows which directory types are compatible with specific system environments.

Table 23. Directory Types and System Environments

z/VM Release and Environment	Object Directory derived from this Source Directory Type		
	Non-SSI	SSI-ready	SSI-enabled
Prior to 6.2	Y	Y	Not supported
6.2 or later, no SSI cluster defined	Y	Y	Error
6.2 or later, in a single-member SSI cluster	Y	Y	Y
6.2 or later, in a multiple-member SSI cluster	Error	Error	Y

Note:

1. "Not supported" means this environment is not supported because in older z/VM releases, CP does not know how to use this directory format. In older z/VM releases, CP can bring the directory online but cannot support multiconfiguration virtual machines.
2. "Error:" means CP will not bring online object directories derived from this type of source directory.

Source Directory File Formats

The source directory can exist in two file formats: monolithic and cluster (not to be confused with an SSI cluster). The monolithic format consists of a single CMS sequential file that contains all of the directory statements. The cluster format consists of an index file that points to one or more definition files that contain all of the directory statements. The index file consists only of LOAD statements that point to the definition file(s). A definition file can be a separate part file that contains one of the following:

- DIRECTORY statement(s)
- A global definitions entry
- A profile entry
- A user entry
- An identity entry
- A subconfiguration entry

A definition file could also be a cluster file that contains multiple items from the above list. For details on how to create separate part files or cluster files, see "LOAD Directory Statement" on page 537.

A single cluster format source directory can be made up of an index file and a mixture of separate part definition files and cluster definition files.

Regardless of the file form used for the source directory, the user directory must be processed in the following order:

DIRECTORY statement(s)
Which consists of one or more DIRECTORY statements that define the output object directories.

Global definition entry
Which must begin with the GLOBALDEFS directory statement. The entry also includes directory statements that define global settings to be used by all virtual machine definitions.

Profile entries
Each of which begins with a PROFILE directory statement and consolidates other directory statements that are used in common by many virtual machine definitions.

User, identity, and subconfiguration entries
Each of which begins with a USER, IDENTITY, or SUBCONFIG directory statement. USER begins the entry for an single-configuration virtual machine definition. IDENTITY begins the entry for a multiconfiguration virtual machine definition and includes statements common to virtual machine

configurations in that definition. SUBCONFIG begins the entry for a set of directory statements in a multiconfiguration virtual machine definition that is specific for a member of an SSI cluster.

There are rules about the placement of directory statements within an entry. For details on these rules, refer to the individual statement descriptions in this section and to Table 24 on page 462.

How CP Brings the User Directory Online

During initialization, CP checks for a valid object directory on the system residence volume. If CP finds a valid object directory, it makes that directory the active directory. Otherwise, CP looks for the first valid object directory on the CP-owned volumes brought online during initialization. CP checks the volumes in the order of their appearance in the CP-owned list. If you defined the CP-owned list using CP_OWNED statements (see "CP_OWNED Statement" on page 73) in your system configuration file, CP checks the volumes based on the slot number, in increasing order.

If CP does not find a valid object directory, the system comes up with a default user ID of OPERATOR. You can use this virtual machine to create a valid object directory. In this case, you can bring an object directory online by:

- Entering the DIRECTXA utility to install an object directory on an appropriately formatted CP-owned volume
- Attaching a valid object directory volume whose volume ID appears in the CP-owned volume list but which was not brought online during initialization.

After CP has brought an object directory online, the volume on which it resides remains the directory volume for the duration of the IPL. Rewriting an object directory on a different volume, even one higher in the search order, has no effect. (However, an object directory written higher in the CP-owned volume search order becomes effective when the next hardware or software IPL occurs.)

When DIRECTXA is used to write an object directory on DASD, a pointer to the object directory is written in the volume label for that device. Performing any operation that modifies the volume label, allocation map, or DRCT space on DASD (e.g., CPFMTXA) might result in the object directory not being found when CP tries to bring the directory online.

Directory Statement Categories

Table 24. Directory Statement Categories

Category	Statements	Placement Requirements
Control	• DIRECTORY • GLOBALDEFS • PROFILE • USER • IDENTITY • SUBCONFIG • SYSAFFIN • LOAD • INCLUDE • BUILD • POOL	These statements control the structure of the source directory. Refer to each statement description for details on placement of these statements.
Global	• GLOBALOPTS • POSIXGROUP	These statements can appear only in the global definitions entry.

Table 24. Directory Statement Categories (continued)

Category	Statements	Placement Requirements
General (1)	• COMMAND • CRYTPO* • DATEFORMAT • IPL • LOADDEV • MAXSTORAGE • OPTION CONCEAL, CPUID, LANG, MAXCON, MAXVMCFI, MIH, NOMEASSIST, NOMDCFS, QUICKDSP, SVC76VM, TODENABLE • SCREEN • SHARE • SPOOLFILE • STORAGE • XCONFIG • XSTOR	These statements are non-authorization statements and are allowed in profile, user, identity, and subconfiguration entries (unless an exception is noted). These statements must appear before any device statements in the entry. * CRYPTO operands other than APVIRT are not allowed in profile entries.

Table 24. Directory Statement Categories (continued)

Category	Statements	Placement Requirements
General (2)	• ACCOUNT • ACIGROUP • APPCPASS • AUTOLOG • CLASS • CPU • D8ONECMD • IOPRIORITY • IUCV • LOGONBY • MACHINE • NAMESAVE • NOPDATA • OPTION ACCOUNT, APPLMON, CFVM, CFUSER, CHPID, COMSRV, CRYMEASURE, DEVINFO, DEVMAINT, DIAG88, DIAG98, D84NOPAS, IGNMAXU, LKFAC, LNKEXCL, LNKNOPAS, LNKSTBL, MAINTCCW, NETACCOUNTING, NETROUTER, RMCHINFO, SETORIG, STGEXEMPT, SVMSTAT • POSIXGLIST • POSIXINFO • POSIXOPT • STDEVOPT • XAUTOLOG	These statements are mainly authorization statements. They are allowed in profile, user, and identity entries, but not in subconfiguration entries. These statements must appear before any device statements in the entry.
Device (1)	• CONSOLE • DASDOPT • DEDICATE • LINK • NICDEF • SPECIAL • SPOOL	These statements are allowed in profile, user, identity, and subconfiguration entries. These statements must appear after all general statements in the entry.
Device (2)	• MDISK • MINIOPT	These statements are allowed in user, identity, and subconfiguration entries. These statements must appear after all general statements in the entry.

Blank lines and lines beginning with an asterisk (*) can appear. The information on each directory statement must appear in columns 1 through 71.

Some directory statements support mixed-case operands, so care must be taken when editing the directory source file to avoid accidentally altering the case of these operands. Although the example statements in this section are in mixed case, the case is not significant for most of them. They are shown in mixed case only to improve their readability.

Summary of Directory Statements

Table 25 on page 465 describes the directory statements. It shows where the statement is described, the statement category, and a brief summary of the statement's function. Although the directory statements are listed in alphabetic order, this is not the order in which they can be coded in the directory.

Table 25. Summary of User Directory Statements

Statement	Category	Function	Location
ACCOUNT	General (2)	Defines the account numbers to which a virtual machine charges its costs. It also defines a distribution code.	"ACCOUNT Directory Statement" on page 472
ACIGROUP	General (2)	Specifies the name of a group to which an individual user ID is assigned.	"ACIGROUP Directory Statement" on page 474
APPCPASS	General (2)	Allows a virtual machine to request an APPC/VM path without supplying security parameters, even when the equivalent of SECURITY (PGM) is indicated on the APPCVM CONNECT request.	"APPCPASS Directory Statement" on page 477
AUTOLOG	General (2)	A synonym for XAUTOLOG. See XAUTOLOG "XAUTOLOG Directory Statement" on page 624.	"AUTOLOG Directory Statement" on page 479
BUILD	Control	Used for the SSI environment; specifies a system name and subconfiguration entry, which in turn specifies the SSI member-specific configuration for a multiconfiguration virtual machine instance.	"BUILD Directory Statement" on page 480
CLASS	General (2)	Specifies the privilege class (or classes) assigned to an individual user.	"CLASS Directory Statement" on page 482
COMMAND	General (1)	Specifies a command to be executed after the virtual machine is logged on, and before the virtual machine is IPLed.	"COMMAND Directory Statement" on page 483
CONSOLE	Device (1)	Defines a virtual machine operator's console.	"CONSOLE Directory Statement" on page 485
CPU	General (2)	Specifies a virtual processor that is to be defined automatically when the user logs on to the system.	"CPU Directory Statement" on page 488
CRYPTO	General (1)	Authorizes the user to define virtual cryptographic facilities and provides the guest access to the crypto domains on the cryptographic cards.	"CRYPTO Directory Statement" on page 490
DASDOPT	Device (1)	An extension to the DEDICATE, LINK, and MDISK statements.	"DASDOPT Directory Statement" on page 495
DATEFORMAT	General (1)	Specifies a user's default date format for commands that provide multiple date formats.	"DATEFORMAT Directory Statement" on page 498
DEDICATE	Device (1)	Specifies that a real device is to be used solely by the virtual machine, or that a real tape device is to be shared with other users.	"DEDICATE Directory Statement" on page 500
DIRECTORY	Control	Defines the device on which the directory resides and must be the first statement in the directory.	"DIRECTORY Directory Statement" on page 503
D8ONECMD	General (2)	Defines whether a virtual machine can issue multiple CP commands with DIAGNOSE code X'08'.	"D8ONECMD Directory Statement" on page 506
GLOBALDEFS	Control	Signifies the beginning of the global entry.	"GLOBALDEFS Directory Statement" on page 508
GLOBALOPTS	Global	Used to define global settings to be used while processing virtual machine definitions.	"GLOBALOPTS Directory Statement" on page 509

Table 25. Summary of User Directory Statements (continued)

Statement	Category	Function	Location
IDENTITY	Control	Used for the single system image environment: 1. Starts a multiconfiguration virtual machine definition 2. Defines the multiconfiguration virtual machine's logon identification (user ID) and password 3. Defines the multiconfiguration virtual machine's storage size 4. Defines the multiconfiguration virtual machine's CP command privilege classes.	"IDENTITY Directory Statement" on page 510
INCLUDE	Control	Specifies the name of a profile entry to be invoked as part of the user entry.	"INCLUDE Directory Statement" on page 516
IOPRIORITY	General (2)	Defines a virtual machine's I/O priority queueing range.	"IOPRIORITY Directory Statement" on page 517
IPL	General (1)	Designates a device number or named saved system that CP automatically loads (IPLs) when the user logs on to the system.	"IPL Directory Statement" on page 519
IUCV	General (2)	Authorizes a virtual machine to create an IUCV communication path with another virtual machine.	"IUCV Directory Statement" on page 525
LINK	Device (1)	Accesses another virtual machine's minidisk.	"LINK Directory Statement" on page 533
LOAD	Control	Specifies where the directory entry is to be found. The LOAD statement is valid only in a cluster file format directory.	"LOAD Directory Statement" on page 537
LOADDEV	General (1)	Identifies the location of a program to be loaded as a result of a list-directed IPL. Parameters to be passsed to the program can also be defined.	"LOADDEV Directory Statement" on page 540
LOGONBY	General (2)	Designates up to eight user IDs that can use their own passwords to logon to and use a virtual machine.	"LOGONBY Directory Statement" on page 545
MACHINE	General (2)	Defines the virtual machine mode (ESA, XA, XC, or Z) and the number of virtual processors a virtual machine can define.	"MACHINE Directory Statement" on page 546
MAXSTORAGE	General (1)	Specifies the virtual storage size for a user.	"MAXSTORAGE Directory Statement" on page 548
MDISK	Device (2)	Defines virtual disks (minidisks): permanent minidisks, temporary minidisks, and virtual disks in storage.	"MDISK Directory Statement" on page 550
MINIOPT	Device (2)	An extension to MDISK used when defining non-full-pack minidisks.	"MINIOPT Directory Statement" on page 560
NAMESAVE	General (2)	Authorizes a virtual machine to access a restricted named saved system or saved segment and authorizes the virtual machine to obtain an exclusive writeable copy of a saved segment.	"NAMESAVE Directory Statement" on page 563
NICDEF	Device (1)	Defines virtual network adapters that are fully simulated by CP.	"NICDEF Directory Statement" on page 565
NOPDATA	General (2)	Authorizes a virtual machine to use NOP CCWs to transfer data to CP spool files.	"NOPDATA Directory Statement" on page 571
OPTION	General (1, 2)	Defines certain options available to the virtual machine.	"OPTION Directory Statement" on page 572
POOL	Control	Allows a set of virtual machines to be defined with the same configuration or characteristics.	"POOL Directory Statement" on page 581
POSIXGLIST	General (2)	Specifies all POSIX groups of which the user is a member.	"POSIXGLIST Directory Statement" on page 582
POSIXGROUP	Global	Defines a POSIX group.	"POSIXGROUP Directory Statement" on page 584
POSIXINFO	General (2)	Specifies a user's POSIX information.	"POSIXINFO Directory Statement" on page 585

Table 25. Summary of User Directory Statements (continued)

Statement	Category	Function	Location
POSIXOPT	General (2)	Specifies option settings related to a user's POSIX capabilities.	“POSIXOPT Directory Statement” on page 588
PROFILE	Control	Defines the start of a profile entry.	“PROFILE Directory Statement” on page 591
SCREEN	General (1)	Defines the extended color and extended highlighting options for the virtual machine console.	“SCREEN Directory Statement” on page 593
SHARE	General (1)	Specifies a virtual machine's scheduler share.	“SHARE Directory Statement” on page 595
SPECIAL	Device (1)	Defines virtual displays, communication lines, and channel-to-channel adapters that can or cannot be connected to real devices when the user logs on.	“SPECIAL Directory Statement” on page 597
SPOOL	Device (1)	Defines virtual unit record devices (spooling devices).	“SPOOL Directory Statement” on page 602
SPOOLFILE	General (1)	Describes virtual machine spool file characteristics.	“SPOOLFILE Directory Statement” on page 605
STDEVOPT	General (2)	Specifies the optional storage device management functions available to the virtual machine.	“STDEVOPT Directory Statement” on page 606
STORAGE	General (1)	Specifies the virtual storage size for a user.	“STORAGE Directory Statement” on page 608
SUBCONFIG	Control	Used in the single system image environment; begins the entry for an SSI member-specific configuration for a multiconfiguration virtual machine instance.	“SUBCONFIG Directory Statement” on page 610
SYSAFFIN	Control	Defines how, and to which systems of a multiple-system complex, the subsequent statements apply.	“SYSAFFIN Directory Statement” on page 613
USER	Control	1. Starts the entry for a single-configuration virtual machine definition 2. Defines the virtual machine's logon identification (user ID) and password 3. Defines the virtual machine's storage size 4. Defines the virtual machine's CP command privilege classes.	“USER Directory Statement” on page 616
VMRELOCATE	General (2)	Controls relocation capability of specified user.	“VMRELOCATE Directory Statement” on page 622
XAUTOLOG	General (2)	Designates user IDs that can enter an XAUTOLOG command for the virtual machine.	“XAUTOLOG Directory Statement” on page 624
XCONFIG	General (1)	Specifies control parameters for the extended-configuration facilities provided in the ESA/XC and z/XC virtual machine architectures: access lists and data spaces.	“XCONFIG Directory Statement” on page 626
Blank Line		Can appear anywhere in the directory control file. When the DIRECTXA utility processes the directory control file, it ignores blank lines.	
Asterisk (*)		Any line in the source directory file that begins with an asterisk (*) is a comment. When the DIRECTXA utility processes the directory control file, it ignores comment statements.	

Continued Directory Statements

Certain directory statements are permitted to continue across multiple records in the directory file. These statements continue from one record to the next when the record's last nonblank character in columns 1-71 is a comma (,). The statement is continued beginning in column 1 of the next noncomment, nonblank record. That is, since comment records and blank records are ignored by DIRECTXA, they do not terminate a continued statement.

When DIRECTXA is processing a record that ends in a continuation comma, it effectively deletes the continuation comma and concatenates the prior portion of this record with the next noncomment, nonblank record, with one intervening blank (see "Quoted String Operands" on page 468 for the syntax rules for very long strings). This process is repeated until a noncontinued, noncomment, nonblank record is encountered. The statement is then parsed according to its normal rules. Statement names and keywords cannot be split across multiple records.

Some statements that support continuation can have additional rules for continuing certain operands, such as quoted string operands, across multiple records. These rules are described in the documentation for these operands.

A continued statement can be no longer than 6144 characters. All characters, including blanks and the continuation comma count towards this limit. On the final record of a continued statement, all characters, including blanks, from column 1 up to and including the last nonblank character count towards the limit.

The following are some examples of directory statements continued across multiple records.

```
POSIXINFO IWDIR /home,
          IUPGM /myself   ,
* This is a comment among records of a continued statement.
* The POSIXINFO statement above continues on the next record
FSROOT VMBFS:FPOOL1:FSPACE1


* The following statement has 63 characters that count towards the
* 6144-character limit: 26 on the first record and 37 on the
* next.
POSIXGLIST GIDS 5402 450 ,
           GNAMES all DeptG63 DeptG30
```

Quoted String Operands

Certain operands on certain directory statements are permitted to be specified as *quoted string operands*, which consist of one or more *quoted strings*. Quoted strings are useful if an operand must be permitted to contain imbedded blanks, single quotation marks, or double quotation marks. If the statement is permitted to be continued through multiple records of the directory file, a quoted string operand, with or without imbedded blanks, can continue through multiple records.

Quoted strings and quoted string operands must obey the following rules:

1. A *quoted string operand* consists of one or more quoted strings.
2. A character string can be surrounded by single or double quotation marks. This string, along with its surrounding quotation marks, is referred to as a *quoted string*. A string which is to contain blanks, single or double quotation marks must be surrounded by quotation marks (single or double). It is permissible to include other characters inside the quotation marks.
3. To include a single quotation mark inside a quoted string surrounded by single quotation marks, two consecutive single quotation marks are required inside the string. Likewise, to include a double quotation mark inside a quoted string surrounded by double quotation marks, two consecutive double quotation marks are required inside the string.
4. To include a single quotation mark inside a quoted string surrounded by double quotation marks, the single quotation mark need not be duplicated. To include a double quotation mark inside a quoted string surrounded by single quotation marks, the double quotation mark need not be duplicated.
5. The final result string of a quoted string is formed as follows: any duplicated single or double quotation marks are reduced to one single or double quotation mark, respectively, then the surrounding quotation marks are removed.
6. The final result string of a quoted string operand is formed as follows: the result string from adjacent quoted strings are concatenated with *no intervening blank*. Note that this concatenation happens even if a continued record ends with a quoted string and the next record begins with a quoted string.

7. A quoted string cannot continue to the next record. It must begin and end on the same record, just as statement names and keywords. A quoted string operand, however, can continue through multiple records by splitting it into quoted strings and splitting the statement between quoted strings.
8. A quoted string cannot be imbedded in or concatenated with another quoted string or a nonquoted string. For example, 'AB'"C" and ABC'DEF' are not valid quoted strings.

The following are some examples of directory statements with valid quoted string operands.

```
POSIXINFO IWDIR '/home'
POSIXINFO IWDIR 'This is Larry''s',
                '"Initial Work'  ,
                'ing Directory"' ,
     IUPGM 'The latest "Init'   ,
"ial User Program"""
```

Creating the User Directory

To create a user directory, you must:

1. Create a CMS file (or edit an existing CMS file) that contains a virtual machine definition for each virtual machine that will run on z/VM.
2. Use the DIRECTXA utility to run the user directory creation program.

IBM provides a sample directory in a CMS file named USER DIRECT. You can edit the USER DIRECT file to tailor your system's user directory.

If you choose not to use the USER DIRECT file, you can create your own CMS file and code directory statements in it.

Finally, to update the directory, your virtual machine must have write access to the volume that contains the directory (by convention, the virtual machine with user ID MAINT or your chosen installation user ID usually has write access to the proper minidisk).

Running the User Directory Program

As previously mentioned, you must use the DIRECTXA utility to run the user directory creation program. The general format of the command to execute the DIRECTXA utility is:

```
directxa fn ft fm
```

fn
is the file name of your directory. The default is USER.

ft
is the file type of your directory. The default is DIRECT.

fm
is the file mode of your directory. The default is *.

For example, to bring online the new version of a directory named USER DIRECT, enter:

```
directxa
```

To bring online the new version of a directory named SPECIAL DIRECT, enter:

```
directxa special
```

To bring online the new version of a directory named SECRET DRCT, enter:

```
directxa secret drct
```

Notes:

1. By default, the DIRECTXA utility is located on the cross release utilities disk (PMAINT 551). This disk must be accessed to run the DIRECTXA utility.
2. To update the directory your installation is currently using, you must have privilege class A, B, or C (by convention, MAINT or your chosen installation user ID usually has the appropriate class).
3. The system does not check for overlapping extents in the directory. Therefore, while changing the directory source file, you should check your new virtual machine definitions to ensure that the new MDISK allocations you are defining do not overlap existing MDISK allocations. If an overlap occurs, one virtual machine user could unknowingly destroy data belonging to another.
4. On a system where different release levels of z/VM can be IPLed, DIRECTXA is required to be at the highest release level of z/VM that can run on the system. When migrating to a higher release level of z/VM, run the highest release level of DIRECTXA to update the object directory prior to IPLing the higher release level of z/VM. An object directory created using the highest release level of DIRECTXA is compatible with all supported release levels of z/VM. Running a higher release level of z/VM with an object directory created using a lower release level of DIRECTXA is not supported.

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

Changing the Directory

To change your directory, you must:

1. Edit the file that contains the source directory
2. Add, delete, or change virtual machine definitions as required
3. Bring your modified directory online by entering the command to start the DIRECTXA utility.

As an alternative method of modifying specific pieces of information in the online directory, you can run an application program that uses DIAGNOSE code X'84' on a class B virtual machine. (DirMaint, the Directory Maintenance licensed program, is one such application. Use of DIAGNOSE code X'84' allows DirMaint to make certain changes in the online directory without reprocessing the source directory.)

When used by an application such as DirMaint, DIAGNOSE X'84' improves the turnaround time needed to make the changes active and minimizes the need to totally reprocess the source directory. The DIAGNOSE code cannot, however, delete existing entries nor alter directory entries in the source directory, but can only replace existing online directory data.

For more information on this DIAGNOSE code, refer to *z/VM: CP Programming Services*.

Checking a Directory for Errors

Before you bring a directory online, if you want to check the directory for errors, use the EDIT option of the DIRECTXA utility. For example, to check a directory named MYDIRECT DIRECT for errors, enter:

```
directxa mydirect (edit
```

In response, DIRECTXA checks MYDIRECT DIRECT for errors but does not bring it online (even if there are no errors).

To check USER DIRECT for errors, enter:

```
directxa (edit
```

Creating Directory Profiles

If certain directory statements are repeated for several users, you can make use of directory profiles (*profile entries*) to save space in the directory. Using profile entries is similar to creating a user or identity entry. To create profile entries:

1. Determine which directory statements are commonly repeated for a designated set of user entries.

2. Create a profile entry that contains those directory statements on the source directory using the PROFILE statement (see "PROFILE Directory Statement" on page 591).
3. Insert an INCLUDE statement (see "INCLUDE Directory Statement" on page 516) immediately after the USER statement for those entries that will reference the profile.
4. Remove the existing directory statements in the user or identity entries that reference a profile except for those directory statements that are unique to those entries.
5. Run DIRECTXA in EDIT mode to ensure that there are no errors. Any error conditions result in appropriate messages to your console.
6. Correct any errors and run DIRECTXA again to install the newly created directory.

Creating Multiconfiguration Virtual Machine Definitions

A multiconfiguration virtual machine definition allows the virtual machine being defined to logon to multiple members of an SSI cluster concurrently. To create a multiconfiguration virtual machine definition, do the following steps:

1. Add an IDENTITY statement to the source directory file to identify the user ID of the virtual machine.
2. Add directory statements that are common to all members in the SSI cluster to the identity entry.
3. If there are directory statements that are unique to individual members in the SSI cluster, then add a BUILD statement to the identity entry for each system in the SSI cluster.
4. For each BUILD statement in the identity entry, add a corresponding SUBCONFIG statement.
5. Add directory statements that are unique to each system in the subconfiguration entry.
6. Run DIRECTXA in EDIT mode to ensure that there are no errors. Any error conditions result in appropriate messages to your console.
7. Correct any errors and run DIRECTXA again to install the newly created directory.

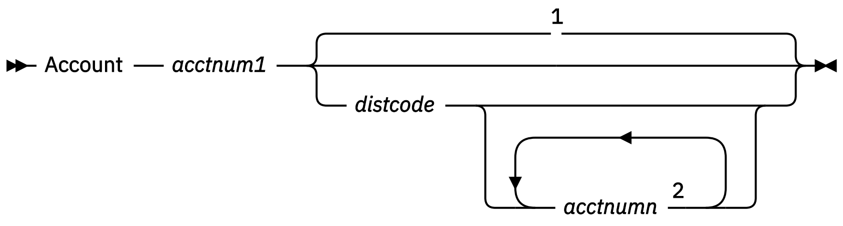
Notes:

1. If you have an identity entry with no BUILD statements, then the user ID specified on the IDENTITY statement is allowed to log on to all members in the SSI cluster concurrently. The configuration of each virtual machine instance as defined by the user directory is identical on all members in SSI cluster.
2. If you have an identity entry with at least one BUILD statement, there should be a BUILD statement for each member in the SSI cluster. Otherwise, the user ID specified on the IDENTITY statement is allowed to log on only to members that are identified on one of the BUILD statements. The user ID is treated as NOLOG on members that are not specified on any of its BUILD statements. The configuration of the virtual machine at logon time is defined by the directory statements in the identity entry, any included profile entry, and the associated subconfiguration entry. The directory statements in the subconfiguration entry allow for the virtual machine configuration to be different on each system of the SSI cluster.

Determining How Much Space the Directory Needs

The directory requires a minimum of four cylinders of CKD/ECKD space or 600 pages of FBA DASD space. (If you do not allocate enough space for your directory, the DIRECTXA program sends you a message to this effect.) For additional information on determining the space required for the directory, refer to "Directory Space" on page 652.

ACCOUNT Directory Statement



Notes:

[1] The default distribution code is the user ID from the USER or IDENTITY statement.

[2] Specify *acctnumn* up to 7 times.

Purpose

The ACCOUNT statement specifies an account number to which a virtual machine may charge its costs. It also includes the distribution code, a code that has no meaning to CP but may be used by your installation as it needs it (for example, to designate where printed output is to go).

How to Specify

The ACCOUNT statement is allowed in profile, user, and identity entries. If specified, the ACCOUNT statement must go before any device statements in an entry. (For a list of device statements, see Table 24 on page 462.) ACCOUNT statements in the profile entry are used only if no ACCOUNT statements are in the user or identity entry.

Operands

acctnum1
defines the primary account number. An account number can be 1- to 8-characters long.

distcode
defines the distribution code for printed and punched output. If you specify *acctnum1* and do not specify *distcode*, the distribution code is the user ID from the USER or IDENTITY statement. If you are defining alternative account numbers, you must specify *distcode*. See the usage notes and examples.

acctnumn
defines up to seven alternative account numbers.

Usage Notes

1. You can code multiple ACCOUNT statements in the same directory entry. The total number of account numbers you code for that entry cannot be more than eight, whether you code one ACCOUNT statement or many.
2. If you code several ACCOUNT statements, *distcode* must appear on the first statement only, immediately after the primary account number.
3. When logging on, a user has the option of specifying an alternative account number on the LOGON command. If the specified alternative account number is in the virtual machine definition, CP charges that alternate account number. If a user logs on without specifying an alternative account number on the LOGON command, CP charges the primary account number.
4. If a virtual machine has several account numbers, the user can switch account numbers using the SET ACCOUNT command. For more information, see SET ACCOUNT in *z/VM: CP Commands and Utilities Reference*.

Examples

1. To specify JOB101 as a virtual machine's account number, use the following statement in the virtual machine's definition:

   ```
   Account job101
   ```

2. To specify JOB101 as a virtual machine's account number and BIN-99Z as a virtual machine's distribution code, use the following statement in the virtual machine's definition:

   ```
   Account job101 bin-99z
   ```

3. To specify:

   - JOB101 as a virtual machine's account number,
   - BIN-99Z as a virtual machine's distribution code, and
   - JOB102, JOB103, and PLAY999 as a virtual machine's alternative account numbers

   use the following statement in the virtual machine's definition:

   ```
   Account job101 bin-99Z job102 job103 play999
   ```

ACIGROUP Directory Statement

```
►►— ACIgroup —— groupname —►◄
```

Purpose

The ACIGROUP statement is an optional statement used to specify the name of the group to which this user (user ID) is assigned. To assign Access Control Interface (ACI) group names to users, you must include the ACIGROUP statement in the virtual machine definitions of all users you are specifying as members of groups. You cannot assign more than one ACI group name to each user in the directory, nor can you specify more than one ACIGROUP statement per directory entry.

How to Specify

The ACIGROUP statement is allowed in profile, user, and identity entries. If specified, the ACCOUNT statement must go before any device statements in the entry. For a list of device statements, see Table 24 on page 462. One ACIGROUP statement is allowed in a profile entry, but is used only if no ACIGROUP statement exists in the user or identity entry.

Operands

groupname
: specifies the name of the ACI group to which this user is assigned. The variable *groupname* may consist of any 1 to 8 characters. However, the application programs that use this value (for example, IBM licensed program RACF) may place their own restrictions on the character set that can be used. Therefore, it may be necessary to review the ACI group names, specified if any applications reference them. Only one ACIGROUP statement may be specified per virtual machine definition.

Examples

To specify that user ID JONES belongs to group MUNDANE, use the following directory statements:

```
User jones ww11qq 8m 16m *
 .
 .
 .
ACIgroup mundane
```

ADJUNCT Directory Statement

```
►► ADJUNCT ── templateuser ►◄
```

Purpose

The ADJUNCT statement authorizes the user in whose directory entry it appears to create an adjunct virtual machine configuration, and identifies the user definition (elsewhere in the directory) to be used as a template for the adjunct configuration when this user issues the ADJUNCT START commmand.

How to Specify

The ADJUNCT statement is allowed in profile, user, and identity entries. If specified, the template user ID specified on the ADJUNCT statement must have a corresponding USER statement which defines the adjunct configuration.

Operands

templateuser
specifies the user definition to treat as the template for creating the adjunct configuration. Any number of users can specify the same adjunct template user. The template user must be defined with password NOLOG to prevent it from being logged on as an independent user.

Usage Notes

1. Only one ADJUNCT statement is allowed in a USER definition.

Examples

1. A template adjunct definition named CMSADJ might appear as follows in the directory:

```
USER CMSADJ NOLOG 32M 1G G
  IPL CMS PARM AUTOCR
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
  LINK MAINT 190 190 RR
  LINK MAINT 19D 19D RR
  LINK MAINT 19E 19E RR
  LINK MAINT 19F 19F RR
  LINK MAINT 1A1 1A1 RR
  LINK MAINT 120 120 RR
  LINK MAINT 19A 19A RR
  LINK * 191 191 MR
```

2. The USER entry to allow creation of an adjunct according to the above template could look as follows:

```
USER VTEST1 NOPASS 256M 1G G
  ADJUNCT CMSADJ
  IPL CMS PARM AUTOCR
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
  LINK MAINT 190 190 RR
  LINK MAINT 19D 19D RR
  LINK MAINT 19E 19E RR
  LINK MAINT 19F 19F RR
  LINK MAINT 1A1 1A1 RR2
  LINK MAINT 120 120 RR
  LINK MAINT 19A 19A RR
  MDISK 0191 3390 2463 0005 X1USR7 MRD RREAD WWRIT MMULT
  MDISK 0192 3390 2650 005  X2USR8 MR  RREAD WWRIT MMULT
```

```
     MDISK 0340 3390 2655 150  X2USR8 MR  RREAD WWRIT MMULT
     MDISK 0399 3390 2805 030  X2USR8 MR  RREAD WWRIT MMULT
```

Note: In the MRD option on MDISK 0191, the D stands for “deferred” and means the minidisk won't automatically be linked into user VTEST1's principal configuration when VTEST1 logs on. Instead, the LINK * 191 statement in the CMSADJ template definition will cause VTEST1 191 to be linked into the adjunct configuration when user VTEST1 issues ADJUNCT START.

APPCPASS Directory Statement

```
►► APPCpass ── gate_lu ── gate_known_lu ── userid ── password ─►◄
```

Purpose

The APPCPASS directory statement allows a virtual machine to request an APPC/VM path without supplying security parameters (user ID and password), even though the equivalent of SECURITY (PGM) is indicated on the APPCVM CONNECT request. Each APPCPASS statement for a virtual machine identifies a possible target of APPCVM CONNECT requests that indicate SECURITY (PGM), and provides the user ID and password needed for security authorization to complete the connection. When a CONNECT request that requires user ID and password includes only user ID, CP searches for a corresponding APPCPASS statement in the originating virtual machine's definition to obtain the password. If the CONNECT request requires user ID and password, but includes neither, CP searches for a corresponding APPCPASS directory statement to obtain both.

How to Specify

The APPCPASS statement is allowed in profile, user, and identity entries. If specified, the APPCPASS statement must go before any device statements in the entry. (For a list of device statements, see Table 24 on page 462.) APPCPASS statements found in a profile entry are added to those found in the including user or identity entry with no duplicate checking performed. A statement that appears within a user or identity entry is processed before a statement that appears within an included profile entry.

Operands

gate_lu
specifies the 8 character GATEWAY_LU_NAME that is the first part of the locally-known LU name. A shorter name must be padded on the right with blanks to a length of 8 characters.

gate_known_lu
specifies the 8 character GATEWAY_KNOWN_LU_NAME that is the second part of the locally-known LU name, which determines the target for which security parameters are being specified. When the GATEWAY_LU_NAME is *IDENT, the GATEWAY_KNOWN_LU_NAME must be 8 bytes of binary zeros (X'0000000000000000').

userid
specifies the 1- to 8-character user ID.

password
specifies the 1- to 8-character password that corresponds to the user ID value at the target identified by the locally-known LU name.

Usage Notes

1. Multiple APPCPASS statements can be specified for a single virtual machine.
2. The same *gate_lu* and *gate_known_lu* values may be on multiple APPCPASS statements for a single virtual machine. Each of these APPCPASS statements can have a different user ID value. In this manner, a virtual machine can use multiple user ID and password pairs to gain access to the target. If two or more such statements also have the same user ID value, only the first statement is used to obtain security parameters for the designated target.
3. APPCPASS statements are an alternative to specifying security parameters for APPCVM CONNECT requests from CMS applications in the CMS file that supports the Systems Application Architecture® (SAA) Common Programming Interface for Communications (CPIC).

Examples

The following APPCPASS statements in the virtual machine definition of JOEUSER's virtual machine define a list of LUs and their security parameters with which JOEUSER's virtual machine can establish communications. This is not a definitive list. Programs running in JOEUSER's virtual machine could develop and pass the required LU and security information dynamically. The APPCPASS statements simply provide default security parameters for the identified LUs for cases when they are not dynamically supplied.

```
APPCpass  Regional  SalesOff  Outlet01  pass01
APPCpass  HomeOffc  Invntory  Outlet01  pass02
APPCpass  HomeOffc  Reports   SalesMgr  passsmgr
APPCpass  HomeOffc  MailBox   MailMan   passmm
APPCpass  HomeOffc  MailBox   JoeUser   passpers
```

With the first statement, JOEUSER could establish a connection with the LU named REGIONAL SALESOFF and be known as user OUTLET01 with password PASS01. With the second, JOEUSER could establish a connection with the LU named HOMEOFFC INVNTORY and also be known as OUTLET01 but with another password of PASS02. With the third, JOEUSER could establish a connection with the LU named HOMEOFFC REPORTS and be known as SALESMGR with a password of PASSSMGR. With the fourth, JOEUSER could establish a connection with the LU named HOMEOFFC MAILBOX and be known as MAILMAN with a password of PASSMM. With the last, JOEUSER could establish a connection with the LU named HOMEOFFC MAILBOX to be known as himself, JOEUSER with a password of PASSPERS.

AUTOLOG Directory Statement

Purpose

The AUTOLOG statement is a synonym for XAUTOLOG that is available for compatibility reasons. Refer to "XAUTOLOG Directory Statement" on page 624 for the format and for information on coding the XAUTOLOG statement.

How to Specify

The AUTOLOG statement is allowed in profile, user, and identity entries. If specified, the AUTOLOG statement must go before any device statements. (For a list of device statements, see Table 24 on page 462.) Multiple AUTOLOG statements are allowed in a profile entry, but are used only if no AUTOLOG or XAUTOLOG statements are in the including user or identity entry.

BUILD Directory Statement

```
►► BUILD — ON — sysname — USING — SUBCONFIG — id ►◄
```

Purpose

The BUILD statement is used as part of a multiconfiguration virtual machine definition. The BUILD statement specifies a system name of an SSI cluster member and the SUBCONFIG ID of the subconfiguration entry that contains the system-specific directory statements for a multiconfiguration virtual machine instance.

How to Specify

BUILD statements are allowed only in identity entries. If specified, they must immediately follow an IDENTITY statement, an INCLUDE statement, or another BUILD statement. In an SSI-enabled directory, an identity entry can contain up to 32 BUILD statements. Each of these BUILD statements must have a unique system name identifying one of the members in an SSI cluster. In an SSI-ready directory, an identity entry can have at most one BUILD statement. The system name on this BUILD statement must be * .

Operands

sysname

specifies a 1- to 8-character member system that identifies the SSI member to which the SUBCONFIG definition applies. *sysname* must be an alphanumeric string or a single asterisk (*).

In an SSI-ready directory, the asterisk character (*) must be specified for the *sysname*. The asterisk indicates that the member defaults to the IPLed system. In this case, just one BUILD statement can be specified within the identity entry.

In an SSI-enabled directory, an asterisk is not allowed as the *sysname* and each BUILD statement within an identity entry must specify a unique system name.

id

defines the virtual machine's 1- to 8-character SUBCONFIG ID. Each BUILD statement within the entire directory must specify a unique SUBCONFIG ID. LOGN*xxxx*, LOGL*xxxx*, LOGV*xxxx*, LOGNSYSC, LOGNSYSG, and SYSTEM are reserved for CP use.

Usage Notes

1. The BUILD statement is used in combination with the IDENTITY and SUBCONFIG statements. For more information, see "IDENTITY Directory Statement" on page 510 and "SUBCONFIG Directory Statement" on page 610.
2. A SUBCONFIG statement with an ID matching the ID specified on the BUILD statement must exist in the directory.

Examples

1. In an SSI cluster, if you want user ID MAINT to be allowed to log on to any member in the SSI cluster by using a combination of common virtual machine specifications and member-specific virtual machine specifications, create a multiconfiguration virtual machine definition such as:

```
IDENT MAINT MAINTPAS
BUILD ON SYS1 USING SUBCONFIG MAINT1
BUILD ON SYS2 USING SUBCONFIG MAINT2
CLASS ABCDEFG

SUBCONFIG MAINT1
```

```
STORAGE 64M
MDISK 201 3390 100 10 MNTVL1 RR

SUBCONFIG MAINT2
STORAGE 128M
MDISK 202 3390 100 10 MNTVL2 RR
```

CLASS Directory Statement

```
►► CLass ── classes ►◄
```

Purpose

The CLASS statement specifies the privilege class or classes assigned to an individual user.

How to Specify

The CLASS statement is allowed in profile, user, and identity entries. If specified, the CLASS statement must go before any device statements. One CLASS statement is allowed in a profile entry if the CLASS field on each of the USER or IDENTITY statements of the including directory entries is blank or contains only an asterisk (*). The CLASS statement in a user or identity entry overrides a class statement in an included profile entry. For a list of device statements, see Table 24 on page 462.

Operands

classes

specifies up to 32 privilege classes of CP commands a user can enter. Command classes are A through Z, and 1 through 6. Each character represents a single privilege class, and may appear in any order, without duplication, and cannot be separated by blanks. You can redefine your command privilege classes to be different from the IBM-defined default classes A, B, C, D, E, F, and G. For more information, see Chapter 19, "Redefining Command Privilege Classes," on page 449.

Usage Notes

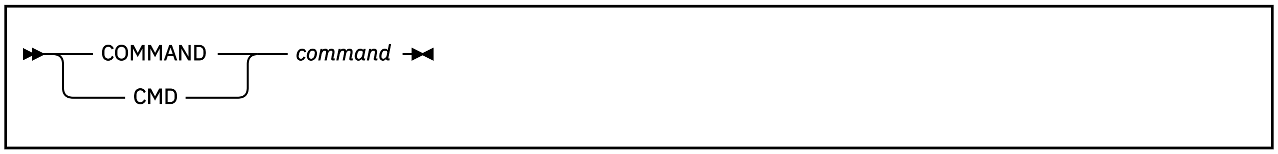
1. If you do not specify a CLASS statement and you specify the class field of the USER or IDENTITY statement as either a blank or an asterisk (*), CP uses the default class or classes specified on the PRIV_CLASSES statement in your system configuration file.
2. If the CLASS field of the USER or IDENTITY statement is not blank and is not an asterisk (*), the CLASS statement is not allowed.
3. If you change the privilege classes for any DIAGNOSE codes, be aware that the change can affect the user's ability to enter commands, because some CMS commands invoke DIAGNOSE codes (for example, DUMPVIEW and GENIMAGE).

Examples

To specify that user ID BONES can enter commands with privilege classes Z, 1, 6, A, X, C, B, 2, 5, and G, use the following directory statements:

```
User Bones ww11qq 8m 16m *
Class z16axcb25g
```

COMMAND Directory Statement



Purpose

The COMMAND statement specifies a command to be executed after the virtual machine is logged on, and before the virtual machine is IPLed. A COMMAND statement may be continued and multiple COMMAND statements are allowed.

How to Specify

COMMAND statements are allowed in profile, user, identity, and subconfiguration entries. If specified, they must go before any device statements. (For a list of device statements, see Table 24 on page 462.)

COMMAND statements in a profile entry are merged with any COMMAND statements in the user, identity, or subconfiguration entries. The result of the merging is that commands on COMMAND statements in the profile entry are executed before those in the user or identity entry. Commands from the subconfiguration entry follow those in the identity entry.

The COMMAND statement may be continued across multiple records in the source directory file. For more information about continued statement rules, see "Continued Directory Statements" on page 467.

Operands

command
specifies the command to be executed. It may not exceed 220 characters in length. (See also Usage Note "1" on page 483 below.) To substitute the virtual machine user ID into the command, specify the token &USERID.

Usage Notes

1. Besides the 220-character length restriction, please be aware that any command operands should be specified in upper case. (The one exception to this is a command operand not parsed by CP, such as the message text for a MESSAGE command, which can use mixed case.) Specify the command only (i.e. do not begin the string with CP or #CP), and do not use the '#' character to put multiple commands on one line.
2. Each command uses the command length + 1 of command buffer space. The maximum command buffer length cannot exceed 3071 characters per virtual machine. For example, the following directory statements would result in PAVUSER requiring 120 bytes of command buffer space:

   ```
   PROFILE PAVPROF
   COMMAND DEFINE HYPERPAVALIAS 5000 FOR BASE 2000
   COMMAND DEFINE HYPERPAVALIAS 5001 FOR BASE 2001
   USER PAVUSER
   INCLUDE PAVPROF
   COMMAND DEFINE HYPERPAVALIAS 3000 FOR BASE 1000
   ```

3. Any form of a command may be invoked using this mechanism, regardless of the command privilege class of the virtual machine. That is, the command is executed as if the virtual machine is authorized for all privilege classes. The syntax used for the command must take this into account.
4. An asynchronous command may be issued using this mechanism, but in this case, subsequent commands should not rely on its results. An option to execute such commands synchronously, if available, may be used to provide more deterministic behavior.

Examples

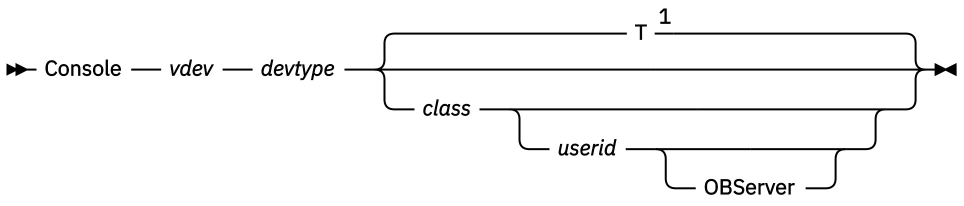
1. To vary a device online and attach it to a user, add the following directory statements in the virtual machine definition:

   ```
   COMMAND VARY ON 1234
   COMMAND ATTACH 1234 TO &USERID AS 4567
   ```

2. To display the status of a virtual device, use the following statement:

   ```
   COMMAND QUERY VIRTUAL 4567
   ```

CONSOLE Directory Statement



Notes:

1 The default is no secondary user or observer.

Purpose

The CONSOLE statement defines the virtual machine console.

This statement also allows the definition of a secondary user or observer:

- The secondary user can control this virtual machine (the primary user) when it is disconnected. The secondary user will receive console output for the primary user and can use the SEND command to send input to the primary user and initiate actions on its behalf. The secondary console function is provided by the single console image facility (SCIF).
- The observer will receive console output for this virtual machine (regardless of whether it is disconnected or connected) but cannot initiate actions on its behalf.

How to Specify

The CONSOLE statement is allowed in profile, user, identity, and subconfiguration entries. If specified, the CONSOLE statement must go after any general statements. (For a list of statements by category, see Table 24 on page 462.) Only one CONSOLE statement is allowed in each entry. If you define a CONSOLE statement in both a user entry and its included profile entry, at logon time two requests to define a console are presented to the system; the request from the user entry is presented first.

A CONSOLE statement in a subconfiguration entry must completely override one specified in the identity entry, including both the secondary user and observer parameters. Because the secondary user must not specify the user ID of the user logged on when the CONSOLE is created, a CONSOLE statement is not allowed to specify the IDENTITY user ID when the CONSOLE statement is contained in a subconfiguration or identity entry. If a CONSOLE statement is contained in the subconfiguration entry, the statement is the one created even if there is a CONSOLE statement in the identity or profile entry.

When processing the CONSOLE statement, DIRECTXA checks for a maximum of five tokens: virtual device number, device type, spooling class, a secondary user ID or observer, in that order. If you specify more than five tokens, DIRECTXA ignores the extra tokens and system processing continues as if you had not specified the extra tokens.

Operands

vdev
is the virtual device number for the virtual console.

devtype
is the device type. Valid device types are 3215 and 3270.

class
is a 1-character spooling class. With spooling classes, your installation can govern the printing of the real spooled output. The class can be any single alphanumeric character from A to Z or from 0 to 9. If you omit *class*, the default is T.

userid
is the 1- to 8-character secondary user ID whose console is to be used when the primary user disconnects. If *userid* is specified, *class* must also be specified.

OBServer
indicates that the specified user is in an observer rather than a secondary user for the virtual machine whose console is being defined.

Usage Notes

1. A virtual machine can have a maximum of one console. Once logged on, users can change the definition of their console to meet their needs. You can use the following commands to change the definition of your console:

Command	Function
DEFINE	• Changes the console address. • Creates a console if one does not already exist.
DETACH	Deletes the console.
SET OBSERVER	Changes the observer.
SET SECUSER	Changes the secondary user.
SPOOL	Changes the spooling class.
TERMINAL CONMODE	Changes the device type.

2. To define a virtual machine console with a console mode (CONMODE) of 3270, the real terminal must be a graphics device. In addition, if the real terminal is an SNA/CCS terminal, the VTAM service machine (VSM) that controls the terminal must support CONMODE 3270. VTAM service machines that use releases of ACF/VTAM earlier than version 3 release 1.1 support only CONMODE 3215. If either condition is violated, an informational message is issued during logon processing and the console mode defaults to 3215.
3. A virtual machine cannot have both a secondary user and an observer defined by any combination of the CONSOLE statement and the SET SECUSER and SET OBSERVER commands.
4. If the designated secondary user or observer is currently unable to function in that capacity (for example, not logged on, or disconnected with no connection to *MSG or *MSGALL), the setting is nevertheless accepted by CP, and the relationship is established when the specified user ID meets the necessary criteria.
5. In an SSI cluster, the secondary user or observer and the primary or observed user can be logged on to different member systems.

   If either user is a multiconfiguration virtual machine:

   - The secondary user or observer will function in that capacity only when it is local (logged on to the same member as the primary or observed user). If the primary or observed user is a multiconfiguration virtual machine, the secondary user or observer must be logged on to the same member as the instance of the primary or observed user for which the secondary user or observer setting is established:
     - If the setting is established by a CONSOLE statement in the identity entry in the primary or observed user's virtual machine definition, a potential secondary user or observer relationship exists for every member instance of the primary or observed user.
     - If the setting is established by a CONSOLE statement in a subconfiguration entry in the primary or observed user's virtual machine definition, a potential secondary user or observer relationship exists for that member instance of the primary or observed user.
   - If the secondary user is remote:

- The secondary user can issue SEND commands to the primary user. The AT *sysname* operands are required if the primary user is a multiconfiguration virtual machine instance.
- The secondary user will not receive responses to SEND commands or any other output from the primary user.

Examples

1. To define a 3215 console at address 009 that produces class T spooled output (the default value), use the following statement in the virtual machine definition:

   ```
   Console 009 3215
   ```

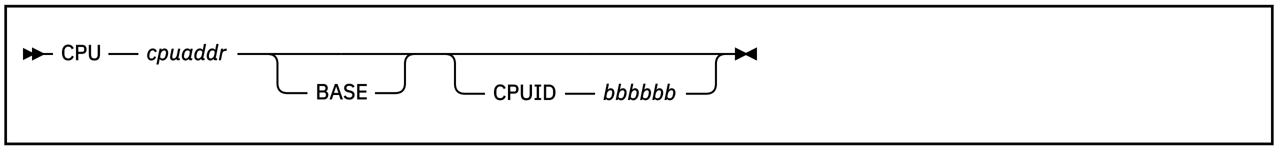
2. To define a 3270 console at address 0A9 that produces class Y spooled output, use the following statement in the virtual machine definition:

   ```
   Console 0a9 3270 y
   ```

3. To define a 3215 console at address 01F that produces class T spooled output and the user ID whose console is to be used when this user ID is disconnected is ALJONES, use the following statement in the virtual machine definition:

   ```
   Console 01f 3215 t aljones
   ```

CPU Directory Statement



Purpose

The CPU statement specifies a virtual processor that is to be defined automatically for the virtual machine at LOGON time.

How to Specify

The CPU statement is allowed in profile, user, and identity entries. If specified, the CPU statement must appear before any device statements. (For a list of device statements, see Table 24 on page 462.) The maximum number of CPU statements allowed in the virtual machine definition is controlled by the *mcpu* operand of the user's MACHINE statement, if specified.

The DEFINE CPU command provides additional parameters not supported on the CPU directory statement. The COMMAND directory statement can be used to specify the DEFINE CPU command and access these additional parameters.

Multiple CPU statements are allowed within an entry. If you specify CPU statements with the same processor address (*cpuaddr*) in a user or identity entry and in a profile entry, the statement in the user or identity entry overrides the one in the profile entry. Otherwise, the CPU statements in the user or identity entry and profile entry are additive.

If a BASE CPU is defined in both a profile and user or identity entry, the CPU statement in the user or identity entry overrides the CPU statement in the profile.

If you do not specify a CPU statement in a virtual machine definition, the address of the base virtual processor is X'00' by default.

Operands

cpuaddr
: defines a virtual processor at the specified address. The address can be any 2-digit hexadecimal number from X'00' to X'3F'.

BASE
: tells CP that this CPU statement defines the base virtual processor. You can only specify BASE on one CPU statement. If you do not specify BASE on any CPU statement, CP defines the base virtual processor as the CPU statement with the lowest virtual processor address (*cpuaddr*).

CPUID *bbbbbb*
: provides the processor identification number to be stored in bits 8 through 31 of the CPU ID that is returned in response to the store processor ID (STIDP) instruction. If the guest is relocated, these bits do not change as a result of the relocation, even if the FORCE ARCHITECTURE or FORCE DOMAIN options were used on the VMRELOCATE command.

  The variable *bbbbbb* is a 6-digit hexadecimal number. (No checking is done to ensure that this number is unique in the virtual configuration.) For base processors, this option overrides the CPUID operand on the OPTION statement. For nonbase processors, it overrides the CPU ID of the base CPU. For more information about the CPUID operand, see "OPTION Directory Statement" on page 572.

Usage Notes

1. The CPU directory statement cannot be used to define virtual CPUs of different processor types. To create these virtual CPUs, use the CP DEFINE CPU command in a COMMAND directory statement. For more information, see Specialty Engine Support in *z/VM: Running Guest Operating Systems*.
2. The protected application environment is supported only for virtual machines running a single processor. If this virtual machine is running a multiprocessor configuration, the SET CONCEAL ON command is rejected.
3. Although the CRYPTO operand is not shown in the syntax for this statement, it is accepted for compatibility, but it provides no function.
4. Dedication of a real processor to a guest is no longer supported, but the DEDICATE and NODEDICATE operands (which are not shown in the syntax for this statement) will still be accepted for compatibility reasons.

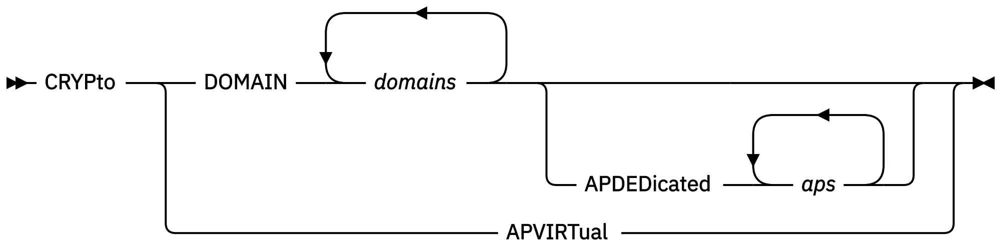
Examples

To define a base CPU at address X'3E', with a CPU ID of X'0000DE', use the following CPU statement:

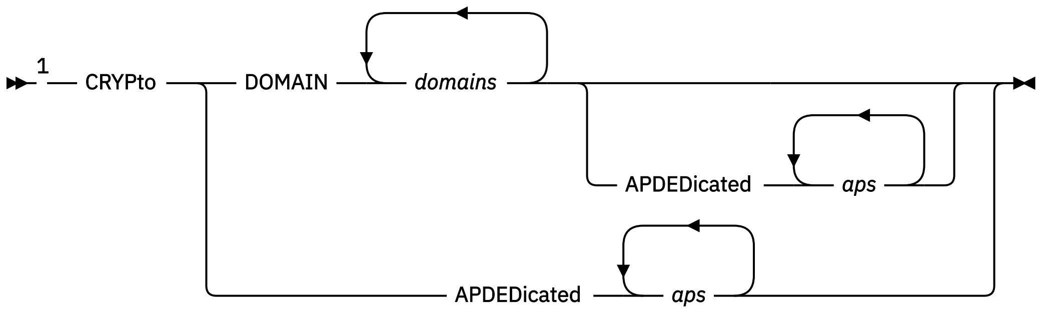
```
CPU 3E base CPUid 0000de
```

CRYPTO Directory Statement

First (or only) CRYPTO statement



Subsequent CRYPTO statements



Notes:

[1] Use subsequent CRYPTO statements only if a previous CRYPTO statement specified a DOMAIN operand. See explanations of the DOMAIN, APDEDICATED, and APVIRTUAL operands.

Purpose

The CRYPTO statement provides the virtual machine access to crypto resources.

How to Specify

The CRYPTO statement is allowed in user, identity, and subconfiguration entries. If specified, it must appear before any device statements. (For a list of device statements, see Table 24 on page 462.)

A CRYPTO APVIRTUAL statement in a profile is allowed as long as APVIRTUAL is not specified on CRYPTO statements in the user or identity entry. In this case, the APVIRTUAL specified in the profile is added to the CRYPTO definition in the user or identity entry. However, when a subconfiguration entry includes CRYPTO statements, the CRYPTO APVIRTUAL statement in the profile is ignored and does not modify the CRYPTO definition in the subconfiguration entry.

A CRYPTO statement in a subconfiguration entry completely overrides one in the identity entry.

You can specify more than one CRYPTO directory statement to assign dedicated crypto resources to the virtual machine. After an initial DOMAIN value is specified, more domains and dedicated adapters (APDEDICATED) can be specified on additional CRYPTO directory statements.

You can assign dedicated (APDEDICATED) or shared (APVIRTUAL) crypto resources but not both to a virtual machine.

If you specify a dedicated (APDEDICATED) resource, you cannot specify shared (APVIRTUAL) resources on any CRYPTO directory statements. If you specify a shared (APVIRTUAL) resource, you cannot specify dedicated (APDEDICATED) resources on any CRYPTO directory statements.

In order for a virtual machine to obtain dedicated access to crypto resources, every specified domain on a specified adapter must be available for dedicated use by this virtual machine at logon time. If one or more domains on an adapter are not available for dedicated use by this virtual machine, then none of the domains on the adapter are assigned to the virtual machine.

Operands

DOMAIN *domains*
specifies up to 256 domains that the virtual machine can use. Valid domain numbers are 0-255, specified in decimal format. The domain numbers can be specified in any order, but must not be duplicated. The DOMAIN operand can be specified on more than one CRYPTO statement.

If the DOMAIN and APVIRTUAL operands are specified, the DOMAIN operand yields no dedicated crypto resource. See usage note "5" on page 492.

APDEDicated *aps*
specifies up to 256 crypto adapters that the virtual machine can use for dedicated access. Valid adapter numbers are 0-255, specified in decimal format. The adapter numbers can be specified in any order, but must not be duplicated.

An APDEDICATED operand cannot be specified before a DOMAIN operand is specified. A DOMAIN operand can be specified on the same CRYPTO statement as the APDEDICATED statement or on a previous CRYPTO statement.

The APDEDICATED operand cannot be specified if the APVIRTUAL operand is specified.

All domains that are specified on all valid CRYPTO statements are assigned to the virtual machine for all crypto adapters that are specified on all valid CRYPTO statements.

The specified crypto adapters must be selected from the set of cryptos that are selected on the Cryptographic Online List. The Cryptographic Online List must be on the Crypto Image Profile Page for the logical partition in which z/VM is running. The specified DOMAIN values must be selected from the set of domains that are selected on the Usage Domain Index selections on the Crypto Image Profile Page for the Logical Partition. For more information about the Crypto Image Profile Page, see the *Processor Resource/Systems Manager Planning Guide*, SB10-7169.

APVIRTual
tells CP that this virtual machine can access the system's shared crypto resources. If APVIRTUAL is specified, then it must be the only crypto statement in the user's directory.

If the DOMAIN and APVIRTUAL operands are specified, the DOMAIN operand yields no dedicated crypto resource, and the APVIRTUAL operand might be processed. See usage note "5" on page 492.

Usage Notes

1. A specific crypto resource is identified with a crypto adapter number and a domain number.
2. A domain is dedicated to a user on all APs that are assigned to the user.

   For example, assume that a virtual machine uses the following CRYPTO user directory statements:

   ```
   CRYPTO DOMAIN 3 4 APDEDICATED 1
   CRYPTO DOMAIN 2
   CRYPTO APDEDICATED 9
   ```

   If all the specified domains are available on all the specified crypto adapters, then the following crypto resources are assigned to the virtual machine for dedicated use:

```
AP 1 Domain 2
AP 1 Domain 3
AP 1 Domain 4
AP 9 Domain 2
AP 9 Domain 3
AP 9 Domain 4
```

See also the examples in ATTACH in *z/VM: CP Commands and Utilities Reference*.

3. Only one virtual machine should be given dedicated access to a specific crypto resource at a time. It might be useful to have more than one virtual machine with the same crypto definition in the user directory in order to provide backup configurations. In this case, the first virtual machine who logs on receives use of the crypto resources specified. Virtual machines which could be logged on at the same time should not be defined with overlapping crypto definitions. The combination of the APDEDICATED number with the DOMAIN number should be unique across all active crypto users in the user directory.
4. It is recommended that multiple CRYPTO statements for a user be specified contiguously in the virtual machine definition.
5. When APVIRTUAL and DOMAIN operands are both specified , the following results can occur:
   - If the DOMAIN operand includes a domain number, then the user gets the APVIRTUAL capability and the DOMAIN operand is ignored. The DOMAIN operand is tolerated but ignored to provide compatibility with earlier versions of z/VM.
   - If the DOMAIN operand does not include a domain number, then an error message is issued. If the DOMAIN operand with no domain number is on a CRYPTO statement that also includes an APVIRT operand, then the APVIRT operand on that statement is not processed.
6. The crypto adapters specified must be installed on the real processor. If a specified crypto adapter is not installed on the real processor at LOGON time, a message is issued that the adapter is not available.
7. Although the CSU, KEYENTRY, SPECIAL, and MODIFY operands are not shown in the syntax for this statement, they are accepted for compatibility with previous versions. Such statements are ignored and do not update the user directory.
8. Only a z/Architecture virtual machine can use a crypto resource on a CEX adapter that is configured in EP11 coprocessor mode (CEX*P).
9. In an SSI-enabled directory, CRYPTO APDEDICATED and DOMAIN statements can be specified in a SUBCONFIG entry. When used along with BUILD statements, this allows the specified domains on the specified APs to be attached to the virtual machine depending on which member of an SSI cluster the user logs onto. The crypto resources specified for APDEDICATED statements in all SUBCONFIG entries are treated as reserved for dedication on all members of an SSI cluster. Typically, the crypto APs and domains which are available on the processor will be different on each SSI member. However, if more than one member of the SSI cluster has crypto hardware with the same AP numbers and domain numbers that are specified in a SUBCONFIG entry, these members will see these adapters as reserved for dedication or dedicated and they will not be used for crypto sharing.
10. If the crypto resources to be included in the shared pool are not specified on a CRYPTO APVIRTUAL statement in the system configuration file, CP will choose up to two shared crypto resources at CP initialization time. If CP chooses the shared crypto resources, it will not include resources that have been specified on a CRYPTO APDEDICATED directory statement. Crypto resources that are planned for dedication will not be included in the shared pool at CP initialization unless they are specified on a CRYPTO system config statement, overriding the CRYPTO APDED statements in the directory.
11. For more information on planning and managing crypto resources on a z/VM system, see Chapter 5, “Crypto Planning and Management,” on page 33.

Examples

1. To specify that the virtual machine can have access to the shared crypto resources on the system, use the following CRYPTO statement in the virtual machine's definition:

```
Crypto Apvirt
```

2. To specify that a virtual machine can use crypto domains 3 and 7 on adapters 0, 2, and 3, use the following CRYPTO statement in the virtual machine's definition:

```
Crypto Domain 3 7 Apded 0 2 3
```

3. To specify that a virtual machine can use crypto domains 2, 10, 11, 12, and 13 on adapter 4 use the following CRYPTO statement in the virtual machine's definition:

```
Crypto Domain 2 10 11 12 13 Apded 4
```

4. This example shows how CP will process conflicting crypto resource specifications.

   - When the system configuration file contains the following CRYPTO APVIRTUAL statement:

   ```
   CRYPTO APVIRT AP 2 DOMAIN 15 52
   ```

   CP assigns the following crypto resources for shared use:

   ```
   AP 2 Domain 15
   AP 2 Domain 52
   ```

   For details on assigning a crypto resource for shared use, see "CRYPTO APVIRTUAL Statement" on page 80.

   - In addition, when a user's directory entry contains the following statements:

   ```
   CRYPTO DOMAIN 53 APDED 2
   CRYPTO DOMAIN 15 APDED 0 1 3 4
   ```

   CP will attempt to assign the following crypto resources to the virtual machine for dedicated use:

   ```
   AP 0 Domain 15
   AP 0 Domain 53
   AP 1 Domain 15
   AP 1 Domain 53
   AP 2 Domain 15
   AP 2 Domain 53
   AP 3 Domain 15
   AP 3 Domain 53
   AP 4 Domain 15
   AP 4 Domain 53
   ```

   - Because adapter 2 Domain 15 was already assigned for shared use in the system configuration file, it is not reserved for dedicated use and the following message is issued at system initialization time.

   ```
   HCP1721I Crypto AP 002 Domain 015 cannot be dedicated because it is
            reserved for shared use.
   ```

   See message HCP1721I in *z/VM: CP Messages and Codes* for more information.

   - Also, because all requested domains on adapter 2 cannot be assigned for dedicated use, the virtual machine is not assigned any of the domains on adapter 2 at logon time. In this example, adapter 2, Domains 15 and 52 are assigned for shared use, so none of the requested domains on adapter 2 are assigned to this virtual machine for dedicated use. The following messages are issued at logon time:

   ```
   HCP1718I AP 2 Domain 15 is not available for dedicated use.
   HCP1718I AP 2 Domain 53 is not available for dedicated use.
   ```

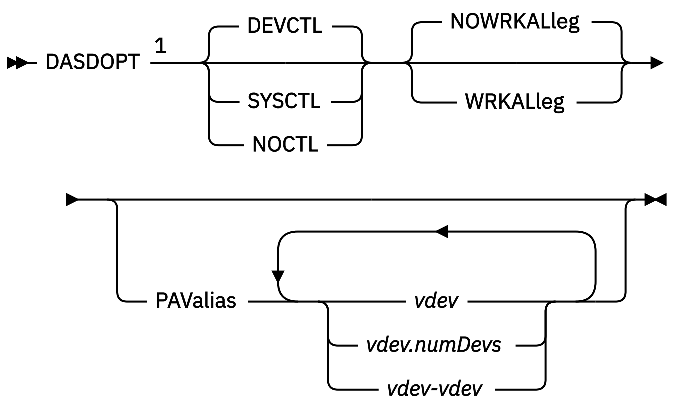
   See message HCP1718I in *z/VM: CP Messages and Codes* for more information.

   The following is the result of a Q CRYPTO DOMAIN command for that particular z/VM configuration, with the USER logged on:

```
AP 000 CEX7P Domain 015 operational  online  attached to USER
AP 000 CEX7P Domain 052 operational  online  free
AP 000 CEX7P Domain 053 operational  online  attached to USER
AP 001 CEX7A Domain 015 operational  online  attached to USER
AP 001 CEX7A Domain 052 operational  online  free
AP 001 CEX7A Domain 053 operational  online  attached to USER
AP 002 CEX7C Domain 015 operational  online  shared
AP 002 CEX7C Domain 052 operational  online  shared
AP 002 CEX7C Domain 053 operational  online  free, dedication planned
AP 003 CEX7A Domain 015 operational  online  attached to USER
AP 003 CEX7A Domain 052 operational  online  free
```

```
AP 003 CEX7A Domain 053 operational  online  attached to USER
AP 004 CEX8C Domain 015 operational  online  attached to USER
AP 004 CEX8C Domain 052 operational  online  free
AP 004 CEX8C Domain 053 operational  online  attached to USER
```

DASDOPT Directory Statement



Notes:

[1] Options can be entered in any order.

Purpose

The DASDOPT statement is an extension to the MDISK, LINK, or DEDICATE statement that immediately precedes it.

How to Specify

Multiple DASDOPT statements are allowed within a profile, user, identity, or subconfiguration entry. In a profile entry, a DASDOPT statement would immediately follow a DEDICATE and/or a LINK statement that describes a full-pack minidisk. In a user, identity, or subconfiguration entry, a DASDOPT statement would also immediately follow an MDISK statement that describes a full-pack minidisk. (For non-full-pack minidisks, see the MINIOPT statement.) Only one DASDOPT statement is allowed for each DEDICATE, LINK, and MDISK statement. DIRECTXA cannot detect every case when the MDISK preceding the DASDOPT is not a full-pack minidisk and it cannot determine if the volume represented on the LINK statement is a full-pack minidisk. In these situations, the characteristics of the DASDOPT operands are given to the disks and unexpected results can occur.

Operands

DEVCTL
: means that the device accepts CCWs that have an effect on resources and functions directly related to the device. See usage note "6" on page 496 for default values.

SYSCTL
: means that the device accepts CCWs that have a direct global effect on subsystem resources and function, not just those related to the device. See usage note "6" on page 496 for default values.

NOCTL
: means that the device does not accept any CCWs that can control subsystem resources or functions, regardless of whether they directly relate to the device.

WRKALleg
: means working allegiance is active on the minidisk. This option is allowed only if DASDOPT is immediately preceded by the MDISK statement.

NOWRKALleg
: means working allegiance is not active on the minidisk. This option is allowed only if DASDOPT is immediately preceded by the MDISK statement.

PAValias
defines one or more alias Parallel Access Volumes for the full-pack minidisk base Parallel Access Volume specified in the preceding LINK or MDISK statement.

vdev
vdev.numDevs
vdev-vdev
is the virtual device address of the alias Parallel Access Volume that you are defining. You can specify a single virtual device (*vdev*), a virtual device combined with a decimal range count (*vdev.numDevs*), a range of virtual devices (*vdev-vdev*), or any combination. The device numbers entered must be hexadecimal numbers between X'0000' and X'FFFF'.

Usage Notes

1. If you are dedicating a DASD with logical addresses (for example, a DASD connected to a paging storage director in a 3380 Storage Control Model 11 or 21), the real device number you specify must be the base address for the device, and virtual device number you specify must follow the addressing rules for base addresses.
2. If you are dedicating a device with logical addresses, both the real device number and the virtual device number you specify must be the base address.
3. When the cache status of a device is changed, the hardware presents an asynchronous state change interrupt to every *vdev* associated with that *rdev* that has DASDOPT DEVCTL specified.
4. WRKALLEG must be used when running two or more MVS guests as part of a Sysplex configuration using the cross system coupling facility (XCF) component of MVS/ESA. This option must be used for any minidisk containing the XCF couple dataset to maintain cross-system lock integrity (and thereby, data integrity) within the sysplex.
5. WRKALLEG/NOWRKALLEG is valid only when the preceding MDISK statement gives the user write access to the minidisk. If the MDISK statement specifies read-only access, CP rejects the WRKALLEG/NOWRKALLEG statement and issues an error message. Furthermore, working allegiance is simulated only when a guest with write access initiates I/O.
6. Levels of Control (DEVCTL, NOCTL, and SYSCTL):

   a. The default control level value is normally DEVCTL, unless DASDOPT follows the LINK statement for a full-pack minidisk. In that case, NOCTL is always the default.
   b. DASDOPT applies only to DASDs on a cached control unit. If you specify DASDOPT for a noncached DASD, the statement is ignored.
   c. Specifying a level of control (NOCTL, DEVCTL, or SYSCTL) for a DASD connected to a cache storage control unit authorizes CP to accept particular control CCWs.

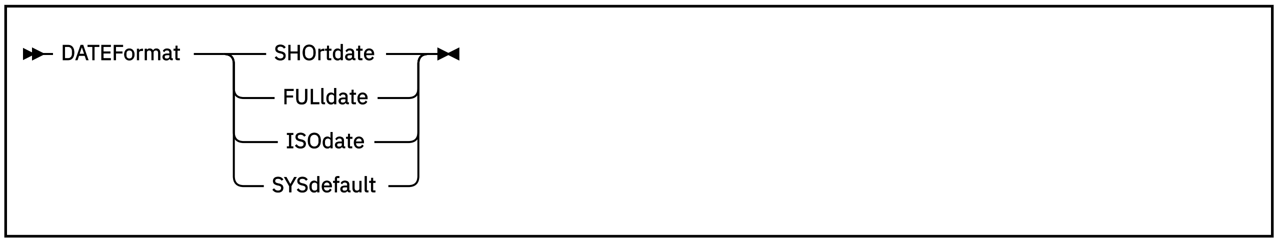
   Table 26 on page 497 shows you the additional control CCWs that CP accepts for a level of control. The DEVCTL column shows the CCWs that CP accepts in addition to those it accepts for the NOCTL level of control. The SYSCTL column shows the CCWs that CP accepts in addition to those it accepts for the DEVCTL and NOCTL levels of control.

Table 26. DASD Control Levels

Additional Control CCWs Accepted for DEVCTL	Additional Control CCWs Accepted for SYSCTL
• Set subsystem mode – Activate caching for device – Deactivate caching for device – Activate DASD fast write – Deactivate DASD fast write – Force deactivate DASD fast write • Perform Subsystem Function – Establish Duplex Pair – Terminate Duplex Pair – Suspend Duplex Pair – Direct I/O to One Device of the Duplex Pair – Set Interface ID	• Set subsystem mode – Make cache available – Make cache unavailable – Force cache unavailable – Make NVS available – Make NVS unavailable – Activate cache fast write – Deactivate cache fast write • Perform Subsystem Function – Destage Modified Tracks – Set Cache Allocation Parameters – Suspend/Resume Function

7. When using the PAValias parameter, the number of virtual alias Parallel Access Volumes that can be associated with a particular virtual base Parallel Access Volume cannot exceed the number of real alias Parallel Access Volumes that are associated with the real base Parallel Access Volume on which the virtual base is defined.
8. The PAValias keyword is only allowed on DASDOPT statements that follow LINK or MDISK statements. PAVALIAS is not allowed on a DASDOPT statement that follows a DEDICATE statement.
9. The DASDOPT statement can be continued onto additional lines. For more information, see "Continued Directory Statements" on page 467.
10. The CP LINK command contains information about the WRKALLEG restrictions when linking. For more information, see LINK in *z/VM: CP Commands and Utilities Reference*.

DATEFORMAT Directory Statement



Purpose

The DATEFORMAT statement specifies a user's default date format for commands that provide multiple date formats.

How to Specify

One DATEFORMAT statement is allowed in a profile, user, identity, or subconfiguration entry. If you specify the DATEFORMAT statement, it must precede any device statements in an entry. (For a list of device statements, see Table 24 on page 462.) A DATEFORMAT statement in a user or identity entry overrides a DATEFORMAT specification in a profile entry. A DATEFORMAT statement in the subconfiguration entry overrides one in the identity entry.

Operands

SHOrtdate
: specifies that dates in command responses be displayed in mm/dd/yy, mm/dd, or yy/mm/dd format, where mm is the month, dd is the day of the month, and yy is the 2-digit year.

FULldate
: specifies that dates in command responses be displayed in mm/dd/yyyy or yyyy/mm/dd format, where mm is the month, dd is the day of the month, and yyyy is the 4-digit year.

ISOdate
: specifies that dates in command responses be displayed in yyyy-mm-dd format, where yyyy is the 4-digit year, mm is the month, and dd is the day of the month.

SYSdefault
: specifies that the default date format for this user be set to the system-wide default. The system-wide default date format may be set in the system configuration file with the SYSTEM_DATEFORMAT statement, or with the CP SET DATEFORMAT command. For more information, see "SYSTEM_DATEFORMAT Statement" on page 286. See also SET DATEFORMAT in *z/VM: CP Commands and Utilities Reference*.

Usage Notes

1. The format of the dates, such as mm/dd/yy, mm/dd, and yy/mm/dd, are dependent upon the command or routine that displays or generates the date.
2. If you omit the DATEFORMAT statement when you code a virtual machine definition, the default format for that user will be the system-wide default (SYSdefault). The user may change their default date format with the CP SET DATEFORMAT command.
3. If the user's default date format is set to SYSDEFAULT, the system-wide default date format that is in effect at logon time will be used until the user logs off or issues the SET DATEFORMAT command. If the system-wide default date format is changed, the user must logoff and log back on or issue the SET DATEFORMAT SYSDEFAULT command to switch to the new system-wide default

Examples

1. To specify the user's default date format as SHORTDATE, use the following DATEFORMAT statement in the virtual machine definition:

   ```
   DATEFORMAT SHORTDATE
   ```

2. To specify the user's default date format as FULLDATE, use the following DATEFORMAT statement in the virtual machine definition:

   ```
   DATEFORMAT FULLDATE
   ```

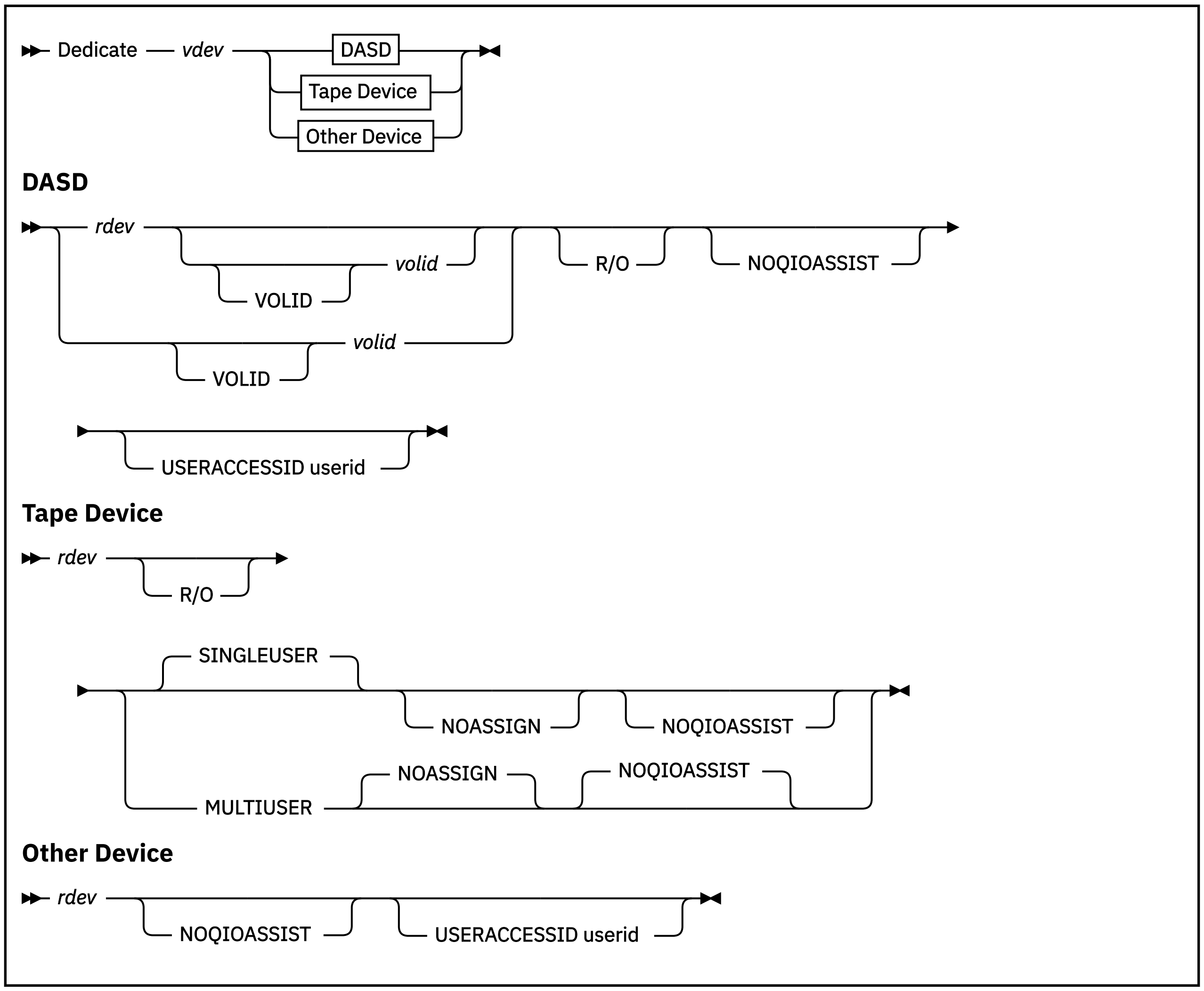
3. To specify the user's default date format as ISODATE, use the following DATEFORMAT statement in the virtual machine definition:

   ```
   DATEFORMAT ISODATE
   ```

4. To specify that the user's default date format should be the same as the system-wide default date format, use the following DATEFORMAT statement in the virtual machine definition:

   ```
   DATEFORMAT SYSDEFAULT
   ```

DEDICATE Directory Statement



Purpose

The DEDICATE statement specifies that a virtual machine has sole use of a real device or is serially sharing a real tape device with other users. The DASDOPT statement is an extension to this statement. For more information, see "DASDOPT Directory Statement" on page 495.

How to Specify

The DEDICATE statement is allowed in profile, user, identity, and subconfiguration entries. If you specify DEDICATE statements, they must follow any general statements in a directory entry. (For a list of statements by category, see Table 24 on page 462.)

Place DEDICATE statements before any SPOOL, LINK, or SPECIAL directory statements. This helps CP assign the correct subchannel number to any virtual devices to which you want to dedicate a real device. For example, you should make sure the virtual subchannel number is the same as the real subchannel number. You would do this by placing the DEDICATE statements before the LINK and SPOOL statements.

A DEDICATE request from a subconfiguration entry is presented to the system before DEDICATE requests from the identity entry. DEDICATE requests from the profile entry are presented to the system after requests from the user or identity entry. If the subchannel assignment is important for virtual devices added using DEDICATE statements, then the placement in the directory is important.

DEDICATE statements are allowed within a profile entry if no duplicate virtual device numbers are on DEDICATE statements within the profile. Any virtual device number in a profile that duplicates a virtual device number within a user, identity, or subconfiguration entry is resolved at logon time.

Operands

vdev
is the virtual device number.

rdev
rdev is the real device number.

VOLID *volid*
is the volume serial number of a disk pack mounted on a real disk storage device. The variable *volid* must be a 1- to 6-character string. If *volid* is less than five characters, VOLID is a required keyword; otherwise VOLID is optional.

R/O
specifies that the virtual device is to be in read-only mode.

SINGLEUSER
dedicates a real tape device to a single user. This is the default for a tape device.

MULTIUSER
attaches a real tape device to be serially shared with other users. See Usage Note "3" on page 501.

NOASSIGN
indicates that the ATTACH process should not issue an ASSIGN channel command for this user. This lets the ATTACH work, even if the specified real device is assigned to another processor. For more information, see ATTACH in *z/VM: CP Commands and Utilities Reference*.

NOQIOASSIST
indicates the device is not eligible for Queued-I/O Assist.

USERACCESSID userid
to allow a user (guest) the ability to give FCP LUN access to the specified userid. In z/VM, authority to LUNs in a Fabric is normally set up by entries in an Access Control Table (ACT). See Linux documentation for details on the ACT configuration tool. z/VM stores the userid into the subchannel when an FCP subchannel is attached or reset. The adapter uses the stored userid names to correlate I/O requests with the rules in the ACT and allows access to the LUNs based on those rules. The USERACCESSID option provides a proxy method to accessing a LUN. The userid specified with the option will be stored in the subchannel, rather than the invoker's userid, when an FCP subchannel is attached or reset. It is assumed that the userid specified with USERACCESSID is already in the ACT.

Usage Notes

1. You may dedicate a real device to only one virtual machine at a time. If more than one virtual machine has a DEDICATE statement for a given real device, only the first virtual machine that logs on receives control of the device. The only exception is when the MULTIUSER operand is specified, which allows serial sharing of an attached tape device.
2. The DEDICATE directory statement defaults to DEVCTL if no DASDOPT is present.
3. The MULTIUSER function is intended for guest operating systems that manage their own assignment of tape devices. It is not intended for CMS unless some external means of managing assignments or serializing access to the tape device among the sharing users is explicitly implemented. Third party assignment and multiple system assignment (Control Access CCW) are not supported.

   To share a tape device, the virtual machine definition for that user must contain a DEDICATE statement for the device that includes the MULTIUSER operand, or the user must specify the MULTIUSER option when using the CP ATTACH command to attach the device. If the first user to log on has a DEDICATE statement for the device that does not include MULTIUSER, or issues the ATTACH command for the device without specifying MULTIUSER, the device becomes dedicated to that user. The device then cannot be attached as MULTIUSER by any user until it is detached by the user to whom it is dedicated.

The MULTIUSER function is valid only for 3480, 3490, and 3590 tape devices.

4. When the ENFORCE_BY_VOLID ON statement is specified in the configuration file, only DEDICATE statements that specify the VOLID of the target device will be accepted.

Examples

1. To dedicate the real device at real device number 100 to the virtual device at virtual device number 100, use the following statement in the virtual machine definition:

   ```
     Dedicate 100 100
   ```

2. To dedicate the real disk at real device number 200 (volume serial number XA9999) to the virtual device at virtual device number 220, use the following statement in the virtual machine definition:

   ```
     Dedicate 220 200 volid xa9999
   ```

3. To dedicate in read-only mode the real disk at real device number 1300 (volume serial number XB9999) to the virtual device at virtual device number 330, use the following statement in the virtual machine definition:

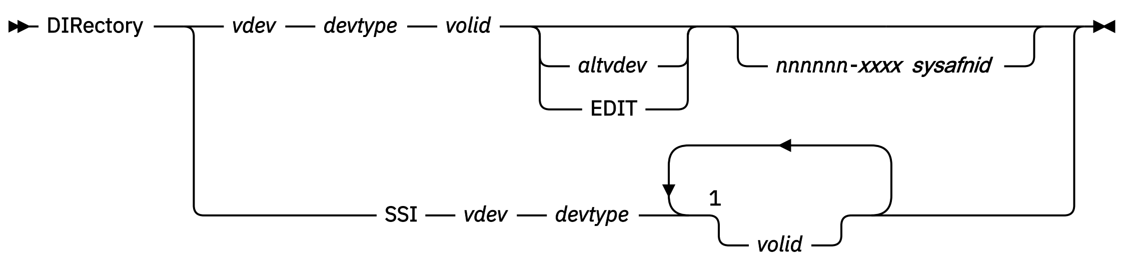
   ```
     Dedicate 330 1300 volid xb9999 r/o
   ```

4. To share the real tape device at real device number 500 and attach it to the virtual machine as virtual device number 181, use the following statement in the virtual machine definition:

   ```
   Dedicate 181 500 multiuser
   ```

5. The USERACCESSID option applies only to first-level and second-level guests. If it is attempted on a third-level or higher virtual machine, the request is translated into corresponding first-level and second-level information only.

DIRECTORY Directory Statement



Notes:

[1] With SSI, you can specify up to 8 *volid*s.

Purpose

The DIRECTORY statement defines to CP the device on which you have allocated space for the directory. This device must be a CP-owned volume.

How to Specify

The DIRECTORY statements must be the first statements in your source directory or control DIRMPART file (cluster format directory).

If this directory is to be used on systems in an SSI cluster, the SSI operand must be specified with up to 8 different volume serial numbers (*volid*s) — one for each system. If the SSI operand is specified, multiple DIRECTORY statements are allowed in order to provide enough room for eight 6-digit volume serial numbers. No other types of statements can be between these DIRECTORY statements. Each of the DIRECTORY statements has the same syntax. Any number of *volid*s (from 1 to 8) can be specified on each statement, with a combined maximum of 8 total allowed on all of the statements. DIRECTXA uses the first DIRECTORY statement for which the specified *vdev* exists and has one of the *volid*s listed on the statement.

If you are sharing a single source directory among several systems that are *not* members of an SSI cluster, a DIRECTORY statement with the processor ID (*nnnnnn*-*xxx*) and *sysafnid* is required for each logical system. This is allowed only in a non-SSI source directory.

In an SSI-ready source directory, DIRECTXA checks for a maximum of 3 tokens: virtual device number, device type, and volume serial number. If you specify more than 3 tokens, DIRECTXA produces an error message and the directory will not be updated.

In an SSI-enabled source directory, DIRECTXA checks for up to 11 tokens: the SSI operand, the virtual device number, the device type, and 1 to 8 volume serial numbers. If you specify more than 11 tokens, DIRECTXA produces an error message and the directory will not be updated.

In a non-SSI source directory, DIRECTXA checks for a maximum of 6 tokens: virtual device number, device type, volume serial number, alternate device number or the EDIT operand, processor ID, and *sysafnid*, in that order. If you specify more than 6 tokens, DIRECTXA ignores the extra tokens and system processing continues as if you had not specified the extra tokens.

Operands

vdev
: is the virtual device number of the device that is to contain the object directory. If multiple DIRECTORY statements are used, the device numbers do not have to be the same.

devtype
: is the device type. Valid device types are:

3380
3390
9336
FB-512

FB-512 is a generic value which can be used in place of specific FBA device types. A 3390 in 3380 track compatibility mode must be coded as 3380. If multiple DIRECTORY statements are used, the device types do not have to be the same.

volid
is the volume serial number of the directory volume. The variable *volid* is a 1- to 6-character alphanumeric string.

altvdev
is an alternative virtual device number, on which to write the directory if the primary virtual device number is unavailable.

The *altvdev* operand is not allowed in an SSI-ready or SSI-enabled source directory.

EDIT
(Supported for compatibility with VM/SP HPO) defines a special work volume to be used by the VM/SP HPO DIRECT command when it is entered with the EDIT operand. DIRECTXA validates the syntax of this statement but ignores its contents.

If you specify the EDIT operand on a DIRECTORY statement, that statement must:

- Be the last DIRECTORY statement in the source directory
- Immediately follow another DIRECTORY statement
- Be the only DIRECTORY statement with an EDIT operand in the source directory.

The EDIT operand is not allowed in an SSI-ready or SSI-enabled source directory.

nnnnnn-xxxx
is the processor ID of the system to which the DIRECTORY statement applies. Use the QUERY CPUID command to get the values for *nnnnnn* and *xxxx*, where *nnnnnn* is the configuration identification, and *xxxx* is the machine-type number of the real machine. For more information, see QUERY CPUID in *z/VM: CP Commands and Utilities Reference*.

No processor ID can be specified in an SSI-ready or SSI-enabled source directory.

sysafnid
is a 1- to 8-character alphanumeric string that identifies the system whose object directory is affected by the SYSAFFIN statements. A maximum of 56 system affinity IDs can be specified. For more information, see "SYSAFFIN Directory Statement" on page 613.

System affinity is not allowed in an SSI-ready or SSI-enabled source directory.

SSI
specifies that this directory is to be shared among all members in a single system image cluster. The *altdev*, EDIT, *nnnnnn-xxx*, and *sysafnid* operands are not allowed when SSI is specified.

Usage Notes

1. CP updates the active z/VM directory dynamically if your virtual machine has the proper privilege class and one of the following is true:
   - The volume serial number specified is the one on which the directory was found during initialization.
   - No directory has yet been found, and the volume serial number specified is currently owned by CP (as specified using the CP_OWNED statement in the system configuration file. For more information, see "CP_OWNED Statement" on page 73.
2. When multiple DIRECTORY statements contain the same system affinity ID, the device information (device number, device type, and volume serial number) on all of the statements must be identical.

The maximum number of 56 unique system affinity IDs can be specified on multiple DIRECTORY statements.

3. System affinity is not supported in SSI-ready or SSI-enabled source directories. If SSI is specified on the DIRECTORY statement, or if there are any IDENTITY, BUILD, or SUBCONFIG statements in the directory, no processor ID (*nnnnnn-xxxx*) or *sysafnid* can be specified.
4. One *sysafinid* per DIRECTORY statement can be specified with a maximum of 56 system affinity ids.

Examples

The following examples show the various ways of coding the DIRECTORY statement:

1. Code the following DIRECTORY statement to specify that:
   - The FBA volume labeled XA0001, at virtual device number 0123, is to contain the new directory
   - The volume at virtual device number 0223 should be used if CP encounters an error while trying to access the directory on virtual device number 0123

   ```
   Directory 0123 fb-512 xa0001 0223
   ```

2. Code the following DIRECTORY statements to specify:
   - A multiple system definition where an 8561 processor is partitioned into two logical partitions: SYSTEMA (configuration identification 002345) and SYSTEMB (configuration identification 012345)
   - A 3390 EDIT work volume with a virtual device number of 0243 and a volume serial number of DRM19F

   ```
   Directory 0123 3390 xa0001 0223 002345-8561 systema
   Directory 0223 3390 xa0002 0233 012345-8561 systemb
   Directory 0243 3390 drm19f edit
   ```

3. To specify a multiple-system environment where four 8561 processors share a single directory, use the following DIRECTORY statements:

   ```
   Directory 0123 9336 fba001 0223 02345-8561 fbasysa
   Directory 0223 9336 fba002 0223 012355-8561 fbasysb
   Directory 0323 9336 fba003 0223 022445-8561 fbasysc
   Directory 0423 9336 fba004 0223 033345-8561 fbasysd
   ```

4. The following is an example of a DIRECTORY statement for an SSI cluster with 3 member systems:

   ```
   Directory SSI 0123 3390 DIRVL1 DIRVL2 DIRVL3
   ```

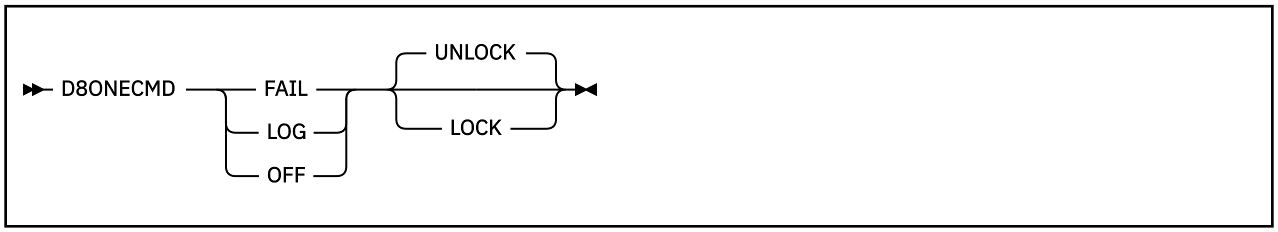
5. The following is an example of how to use multiple DIRECTORY statements for an SSI cluster with 5 member systems:

   ```
   Directory SSI 0123 3390 DIRVL1 DIRVL2 DIRVL3
   Directory SSI 0123 3390 DIRVL4 DIRVL5
   ```

   Here's another way to do this:

   ```
   Directory SSI 0123 3390 DIRVL1
   Directory SSI 0123 3390 DIRVL2
   Directory SSI 0123 3390 DIRVL3
   Directory SSI 0123 3390 DIRVL4
   Directory SSI 0123 3390 DIRVL5
   ```

D8ONECMD Directory Statement



Purpose

The D8ONECMD statement is an optional statement that defines whether a virtual machine can issue multiple CP commands using DIAGNOSE code X'08' and separating the commands with X'15's. This statement also tells CP whether to log the commands to the system operator's console.

How to Specify

The D8ONECMD statement is allowed in profile, user, and identity entries. If specified, it must come before any device statements. (For a list of device statements, see Table 24 on page 462.) You can have only one D8ONECMD statement in an entry. If you specify more than one, you receive an error message. A D8ONECMD statement in a user or identity entry overrides a D8ONECMD statement in an included profile entry.

If you do not specify a D8ONECMD statement, CP does not prevent users from issuing multiple commands with DIAGNOSE code X'08' and allows users to change their D8ONECMD setting.

Operands

OFF
: tells CP that this virtual machine can issue multiple commands with DIAGNOSE code X'08' without any logging.

FAIL
: tells CP to prevent this virtual machine from issuing multiple commands with DIAGNOSE code X'08'. If the virtual machine tries to issue more than one command with DIAGNOSE code X'08', CP will log each command to the system operator's console, process the first command, and reject any subsequent commands in the sequence.

LOG
: tells CP that this virtual machine can issue multiple commands with DIAGNOSE code X'08' and tells CP to log the multiple commands to the system operator's console.

LOCK
: tells CP to lock the D8ONECMD setting for a specific virtual machine definition. This prevents users from changing the D8ONECMD setting for their virtual machine.

UNLOCK
: tells CP not to lock the D8ONECMD setting for a specific virtual machine definition. This allows user to change the D8ONECMD setting for their virtual machine.

Usage Notes

1. If you omit the D8ONECMD for any virtual machine definition, CP uses the defaults: OFF and UNLOCK. This means everything continues as you expected it to in previous releases of VM. If you want to use the D8ONECMD statement but you are afraid to have CP start rejecting commands, we suggest you use the LOG operand. Using the LOG operand does not cause CP to reject any commands and it provides you with a list of all the commands by logging them to the system operator's console. You can then monitor any multiple command activity until you feel comfortable in switching to the FAIL operand.

2. If you are running an earlier level of CMS and D8ONECMD is set to FAIL, some CMS commands (such as SENDFILE, NETDATA, DISK LOAD, and READCARD) may not work properly. Also an earlier level of the programmable operator facility will not initialize if D8ONECMD is set to FAIL.

Examples

1. If you wanted to:
   - Prevent a user from issuing more than one command using DIAGNOSE code X'08' and X'15's,
   - Log each of the commands to the system operator's console, and
   - Prevent that user from changing the D8ONECMD setting

   use the following D8ONECMD statement:

   ```
   D8OneCmd  Fail  Lock
   ```
2. If you wanted to:
   - Allow a user to issue multiple commands using DIAGNOSE code X'08' and X'15's,
   - Record each command, and
   - Allow that user to change the D8ONECMD setting

   use the following D8ONECMD statement:

   ```
   D8OneCmd  Log  UnLock
   ```

GLOBALDEFS Directory Statement

```
►►─ GLOBALDefs ─►◄
```

Purpose

The GLOBALDEFS statement denotes the start of the global definitions entry. If no global settings are specified, this statement is optional.

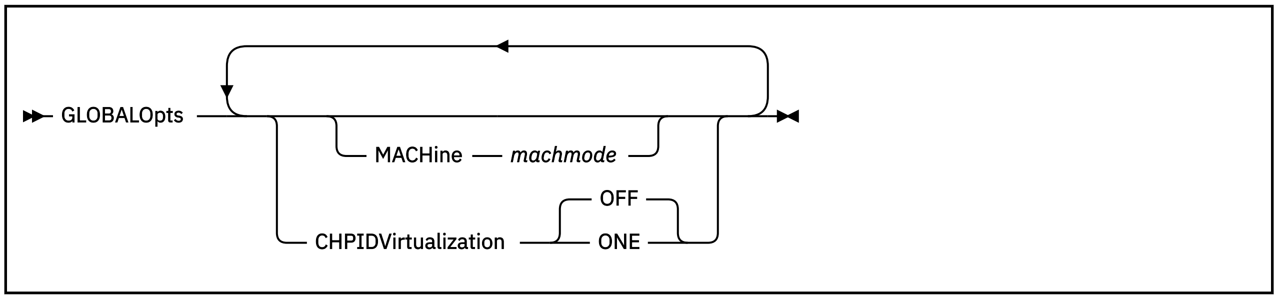
How to Specify

The GLOBALDEFS statement, if specified, must directly follow the DIRECTORY statement(s) and precede all profile, user, identity, and subconfiguration entries. At most one GLOBALDEFS statement can appear in the user directory.

Examples

For an example, see "GLOBALOPTS Directory Statement" on page 509.

GLOBALOPTS Directory Statement



Purpose

The GLOBALOPTS statement is used to define global settings that will be used during user processing.

How to Specify

The GLOBALOPTS statement, if specified, must be within the global definition entry of the source directory.

Operands

MACHine *machmode*
: specifies the virtual machine mode for any virtual machine definition that does not contain a MACHINE directory statement. The valid values that may be used for the virtual machine mode are ESA, XA, XC, and Z.

CHPIDVirtualization *setting*
: specifies the CHPIDVirtualization setting for any virtual machine definition that does not contain an OPTION directory statement with CHPIDVirtualization specified. The valid values for *setting* are ONE and OFF. If this operand is not specified, the default is OFF. GLOBALDEF CHPIDV can only be specified once. If it is specified more than once, an error message is issued.

Examples

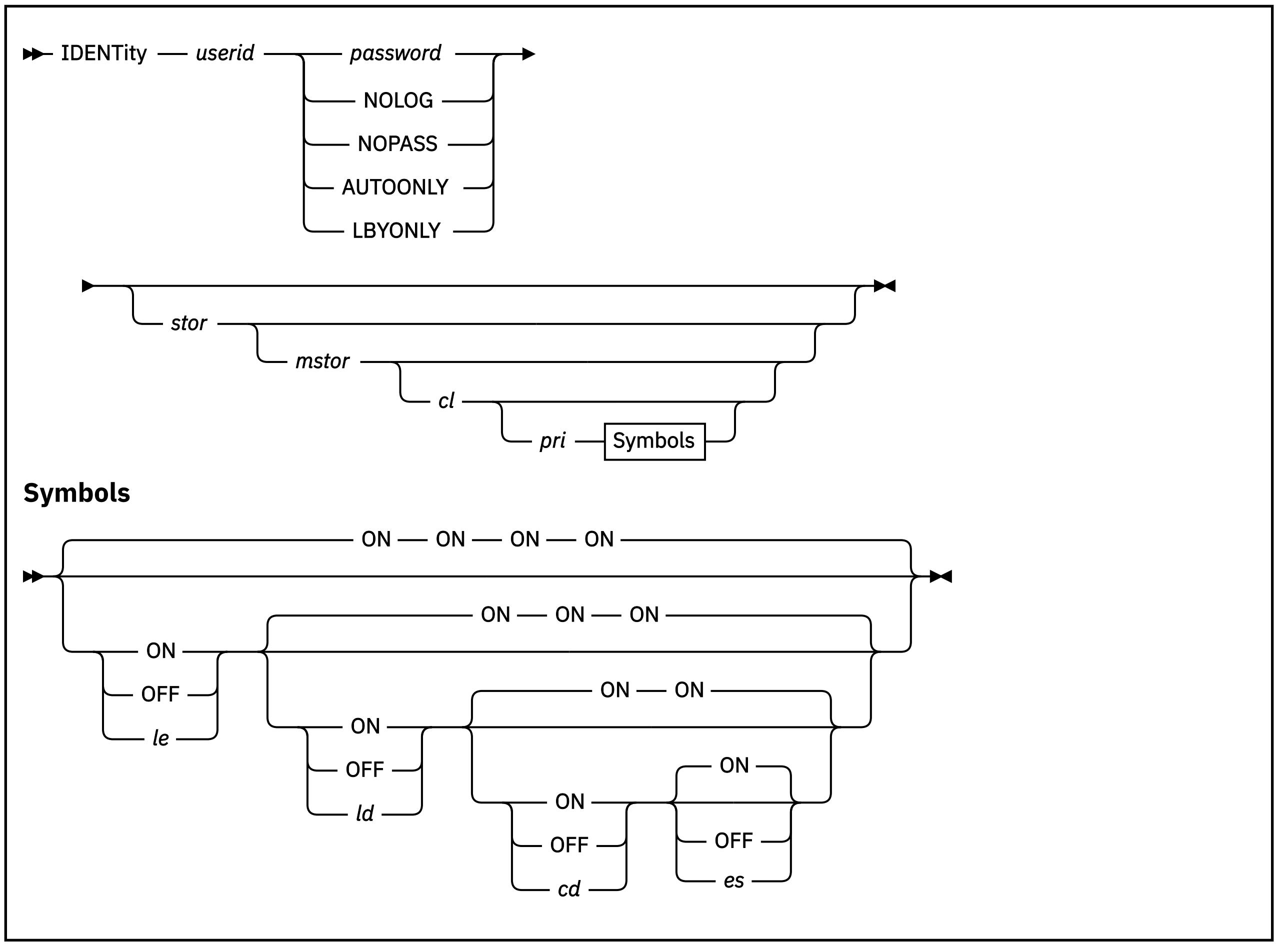
In the examples that follow, assume this multisystem complex definition:

```
Directory 07a4 3390 vmaipl  *01234-3906 vma
Directory 0b64 3390 vmbipl  104567-3906 vmb
Directory 0cF4 3390 vmcipl  *15211-3906 vmc
Directory 0664 3390 vmdipl  *00012-3906 vmd
```

The following global definition entry would define a default machine mode of XA on system vma and a default machine mode of ESA on system vmb.

```
Globaldefs
   Sysaffin vma
   Globalopts machine xa
   Sysaffin vmb
   Globalopts machine esa
```

IDENTITY Directory Statement



Purpose

The IDENTITY statement is used in combination with the BUILD and SUBCONFIG statements to create a multiconfiguration virtual machine for use in a single system image (SSI) cluster. The IDENTITY statement starts a multiconfiguration virtual machine definition, which consists of an identity entry, any included profile entry, and all associated subconfiguration entries. In an SSI-enabled source directory, this virtual machine definition allows multiple virtual machine instances to be defined, which enables the user ID to be logged on concurrently to multiple members of the SSI cluster. Each of these virtual machine instances can have a different configuration from the others.

How to Specify

If specified, an IDENTITY statement must appear after the DIRECTORY statement, global entry, and all profile entries.

When processing the IDENTITY statement, DIRECTXA checks for a maximum of ten tokens: userid, password, logon storage size, maximum storage size, privilege class, a placeholder for compatibility with VM/SP, line-end symbol, line-delete symbol, character-delete symbol, and escape character, in that order. If you specify more than ten tokens, DIRECTXA ignores the extra tokens and system processing continues as if you had not specified the extra tokens.

Operands

userid
defines the multiconfiguration virtual machine's 1- to 8-character user ID. LOGN*xxxx*, LOGL*xxxx*, LOGV*xxxx*, SYSTEM, and SYSTEMMP are reserved for CP use. The *xxxx* can be any character, number, or symbol.

The user ID cannot be the same as a user ID specified on a USER statement or an ID specified on a SUBCONFIG statement.

You should not assign user IDs that are system keywords (such as command names, command operands, or 1- to 4-digit user IDs that could be spool IDs). Assigning system keywords as user IDs can cause unpredictable results. For a list of restricted user IDs, see Restricted User IDs in *z/VM: CP Commands and Utilities Reference*.

You should not assign user IDs that contain colons (:), periods (.), semicolons (;), or slashes (/). These characters are used, or might be used in the future, as delimiters in several CP commands that also take a user ID as a parameter. Results are unpredictable when these commands are entered with a user ID containing any of these delimiter characters.

Ttranslation test tables in HCPTBL allow only the following characters in user IDs:

alphabetics
A-Z

numerics
0-9

The following, comma-delimited characters are also allowed:
@, #, $, _ (underscore), - (hyphen)

Note: It is recommended that the characters @, #, ¢, and " not be used in the *userid* because the system, by default, assigns these characters as logical line editing symbols. For more information, see Logical Line Editing Symbols in *z/VM: CP Commands and Utilities Reference*.

You can override the translation tables in HCPTBL. For more information, see "TRANSLATE_TABLE Statement" on page 304.

password
specifies a 1- to 8-character password that a user enters during the logon procedure.

Note: If DELETEUSER is specified as the password of a user when the DELTA option of the DIRECTXA utility is in effect, the user is marked for deletion in its user index entry. For more information, see DIRECTXA in *z/VM: CP Commands and Utilities Reference*.

NOLOG
specifies that a user cannot log on. You can use the NOLOG option to create virtual machines to which no one can log on. For example, you can place special system DASD areas (allocation, warm-start, and checkpoint areas) on minidisks and then assign the minidisks to virtual machines with NOLOG passwords. This helps you identify areas on a DASD.

Also, you can create minidisks for common user data and then assign the minidisks to virtual machines with NOLOG passwords. Users can then enter a CP LINK command to access those minidisks when they need the data.

NOPASS
specifies that a user does not require a password to log on. It also specifies that a virtual machine can be automatically logged on using the XAUTOLOG command without password authorization.

Even though a user ID is defined with the NOPASS operand, LOGON password authorization might be required when an external security manager is installed. For more information, see the documentation provided by your external security manager.

AUTOONLY
specifies that a user can be autologged, but not logged on at a terminal.

LBYONLY

specifies that:

- Logging on to this virtual machine with the LOGON command requires use of the BY option.
- This user ID cannot be used to log on to any virtual machine with the BY option.

Furthermore, this user ID might not be able to perform functions that require password validation, because LBYONLY is not accepted as a valid password.

CP allows an external security manager (ESM) to override these restrictions, so they are effective only when no ESM is installed or when the ESM defers to CP's processing. (See the ESM documentation for more information.)

For more information, see LOGONBY Directory Statement.

stor

specifies the virtual machine's primary address space size at logon. Specify *stor* as *nu*, where *n* is a 1- to 7-digit decimal number and *u* is the 1-character storage unit suffix (see Table 27 on page 512).

This operand can contain an asterisk (*) as a placeholder. If the operand is omitted or specified as an asterisk, this indicates that the storage size is defined by a STORAGE statement in either the identity entry or a profile entry.

Table 27. Maximum Input Values for Storage Units

Storage unit suffix (*u*)	**Maximum input value (*n*)**
K - kilobytes	9999999
M - megabytes	9999999
G - gigabytes	9999999
T - terabytes	9999999
P - petabytes	16384
E - exabytes	16

Notes:

1. A K specification is rounded up to an M value.
2. The maximum input value of 9999999 for the K, M, G, or T suffix is not a size limit but the physical limit of the operand (7 digits plus suffix). If the maximum input value for one of these suffixes does not allow you to define the amount of storage you want, you need to use a larger storage unit.
3. The maximum size you can specify is 16E or 16384P, although the actual maximum size supported might be restricted by the model of the server where the directory is used.
4. An XC virtual machine can address up to 2047M of storage in its base address space.
5. Allowing many virtual machine instances to have large storage sizes might affect real storage availability. See usage note "1" on page 514.

mstor

specifies the maximum size of the virtual machine's primary address space. Specify *mstor* as *nu*, where *n* is a 1- to 7-digit decimal number and *u* is the 1-character storage unit suffix (see Table 27 on page 512). The default is 1M (1 MB).

This operand can contain an asterisk (*) as a placeholder. If the operand is omitted or specified as an asterisk, this indicates that the maximum storage size is defined by a MAXSTORAGE statement in either the identity entry or a profile entry.

cl

specifies the privilege class or classes of CP commands a user can enter. Command classes are A through Z, and 1 through 6. You can specify up to 32 classes (if they fit). Each character represents a single privilege class and can appear in any order, without duplication, and must not be separated by blanks. The class field can also contain an asterisk (*) as a placeholder. If the field is not specified, or if it contains an asterisk (*) and there is no CLASS statement, CP uses the default class or classes assigned by the PRIV_CLASSES statement in the system configuration file. For more information, see "PRIV_CLASSES Statement" on page 223.

Note: If you have not changed your classes from the IBM-defined defaults, only classes A through G are valid. For more information on defining your own user class structure, see Chapter 19, "Redefining Command Privilege Classes," on page 449.

pri
is for VM/SP compatibility and serves no z/VM function. It must be a number from 1 to 99. If the *pri* specification is not entered, the line-end (*le*), line-delete (*ld*), character-delete (*cd*), and escape (*es*) characters default to ON.

le
ON
OFF
le is a 1-character line-end symbol or a 2-digit hexadecimal equivalent of the symbol. Input following a line-end symbol begins a new logical line. ON means that the virtual machine instance uses the system value. OFF means that CP does not recognize line-end symbols. If omitted, the default is ON.

Note: If you want to set a system-wide default for the line-end symbol, use the CHARACTER_DEFAULTS statement in the system configuration file. For more information, see "CHARACTER_DEFAULTS Statement" on page 67.

ld
ON
OFF
ld is a 1-character line-delete symbol or a 2-digit hexadecimal equivalent of the symbol. A line-delete symbol causes the previous logical line input to be ignored. ON means that the virtual machine instance uses the system value. OFF means that CP does not recognize line-delete symbols. If omitted, the default is ON.

Note: If you want to set a system-wide default for the line-delete symbol, use the CHARACTER_DEFAULTS statement in the system configuration file. For more information, see "CHARACTER_DEFAULTS Statement" on page 67.

cd
ON
OFF
cd is a 1-character delete symbol or a 2-digit hexadecimal equivalent of the symbol. A character-delete symbol causes the previous character input to be ignored. ON means the virtual machine instance uses the system value. OFF means that CP does not recognize character-delete symbols. If omitted, the default is ON.

Note: If you want to set a system-wide default for the character-delete symbol, use the CHARACTER_DEFAULTS statement in the system configuration file. For more information, see "CHARACTER_DEFAULTS Statement" on page 67.

es
ON
OFF
es is a 1-character escape symbol or a 2-digit hexadecimal equivalent of the symbol. The escape symbol causes CP to treat the following character literally, without consideration as an *le*, *ld*, or *cd* character. ON means that the virtual machine instance uses the system value. OFF means that CP does not recognize escape symbols. If omitted, the default is ON.

Note: If you want to set a system-wide default for the escape symbol, use the CHARACTER_DEFAULTS statement in the system configuration file. For more information, see "CHARACTER_DEFAULTS Statement" on page 67.

Usage Notes

1. Allowing many virtual machines to have large storage sizes might affect real storage availability. For each virtual machine, CP creates dynamic address translation (DAT) tables to reference the virtual machine storage. DAT tables include page tables, segment tables, and higher level (region) tables.

   CP keeps the page tables in page management blocks (PGMBKs). Each 8 KB PGMBK references 1 MB of virtual machine storage. PGMBKs might be pageable; as such, their impact on real storage depends on how frequently the MBs of storage they reference are used.

   Segment tables and region tables are allocated from host real storage and are not pageable:

   - To reference the page tables for a primary address space or data space up to 2 GB, 1 - 4 contiguous frames are allocated for the segment table, one frame for each 512 MB of storage.
   - For a primary address space larger than 2 GB, multiple segment tables are created, plus one or more region tables to reference the segment tables. Each region table occupies 1 - 4 contiguous frames. If needed, multiple levels of region tables are created.
2. The IDENTITY statement is used in combination with the BUILD and SUBCONFIG statements to create a multiconfiguration virtual machine definition. A multiconfiguration virtual machine is used in an SSI cluster. For more information, see “BUILD Directory Statement” on page 480 and “SUBCONFIG Directory Statement” on page 610.
3. If there are no BUILD statements in the identity entry, the user ID specified on the IDENTITY statement can log on to any member in the SSI cluster. In this case, the multiconfiguration virtual machine instance is logged on using only the settings from the identity entry.
4. If BUILD statements (even if only one) are contained in the identity entry, the user ID specified on the IDENTITY statement can log on only to members that are identified on those BUILD statements. In this case, the multiconfiguration virtual machine instance is logged on using the common settings from the identity entry and the member-specific settings from the subconfiguration entry.
5. In general, if duplicate specifications exist in an identity entry and its subconfiguration entry, the subconfiguration specification overrides the identity specification, unless noted otherwise in the statement definition. For details on override rules, see the description for each directory statement.

Examples

1. In an SSI cluster, if you want user ID MAINT to be allowed to log on to any member in the SSI cluster by using a combination of common virtual machine specifications and member-specific virtual machine specifications, create a multiconfiguration virtual machine definition such as:

   ```
   IDENT MAINT MAINTPAS
   BUILD ON SYS1 USING SUBCONFIG MAINT1
   BUILD ON SYS2 USING SUBCONFIG MAINT2
   CLASS ABCDEFG
   MDISK 191 3390 100 10 XASRES RR ALL

   SUBCONFIG MAINT1
   STORAGE 64M
   MDISK 193 3390 200 10 VOLUM2 RR ALL

   SUBCONFIG MAINT2
   STORAGE 128M
   ```

   The MAINT multiconfiguration virtual machine definition allows MAINT to log on to SYS1 or SYS2 (or both) using MAINTPAS as a password. MAINT has privilege classes A, B, C, D, E, F, and G on both SSI members. A 191 disk is defined that is global to all MAINT user IDs that log on to SYS1 or SYS2.

   On SYS1, MAINT's virtual machine instance storage size at logon is 64M. In addition to the global 191 disk, a 193 disk is defined that is local to MAINT on SYS1.

   On SYS2, MAINT's virtual machine instance storage size at logon is 128M. This MAINT virtual machine instance has access to the global 191 disk. The 193 disk defined in the MAINT1 subconfiguration is not defined for MAINT2.

2. In an SSI cluster, if you want user ID WORK to be allowed to log on to only one member in the SSI cluster by using a combination of common virtual machine specifications and member-specific virtual machine specifications, create a multiconfiguration virtual machine definition such as:

   ```
   IDENT WORK WORKPASS
   BUILD ON SYS1 USING SUBCONFIG WORK1

   SUBCONFIG WORK1
   STORAGE 64M
   ```

   The WORK multiconfiguration virtual machine definition allows WORK to log on to SYS1 only. Since no other BUILD statements are specified, WORK is not allowed to log on to any other member in the SSI cluster.
3. In an SSI cluster, if you want user ID ALLSYS to be allowed to log on to any member in the SSI cluster by using only common virtual machine specifications, create a multiconfiguration virtual machine definition such as:

   ```
   IDENT ALLSYS WORKPASS
   STORAGE 64M
   ```

   The ALLSYS multiconfiguration virtual machine definition has no BUILD statements, so ALLSYS is allowed to log on to any member in the SSI cluster.

INCLUDE Directory Statement

```
►►─ INclude ── profilename ─►◄
```

Purpose

The INCLUDE statement specifies the name of a profile entry to be invoked as part of the user or identity entry. For more information, see "USER Directory Statement" on page 616 or "IDENTITY Directory Statement" on page 510.

How to Specify

If you specify an INCLUDE statement, it must directly follow the USER or IDENTITY statement. Normally, you can specify only one INCLUDE statement in a directory entry. However, in a non-SSI multisystem complex sharing a single source directory, you could have multiple INCLUDE statements separated by system affinity records, only one of which can apply to a given system.

Operands

profilename
specifies the name assigned to the profile entry and is used to reference the profile. The variable *profilename* is a 1- to 8-character alphanumeric string. Only one profile name may be specified on an INCLUDE statement.

Usage Notes

1. DIRECTXA tries to locate the profile name coded on an INCLUDE statement by searching the list of PROFILE entries already processed and being held in free storage buffers. The PROFILE is then processed (included) into the virtual machine definition and analyzed with respect to the rest of the user's definition data.
2. In a non-SSI multisystem complex sharing a single source directory, it is possible to tailor the user's PROFILE definition by system. This is accomplished by selective inclusion of the PROFILE using the internal form of the SYSAFFIN statement.

Examples

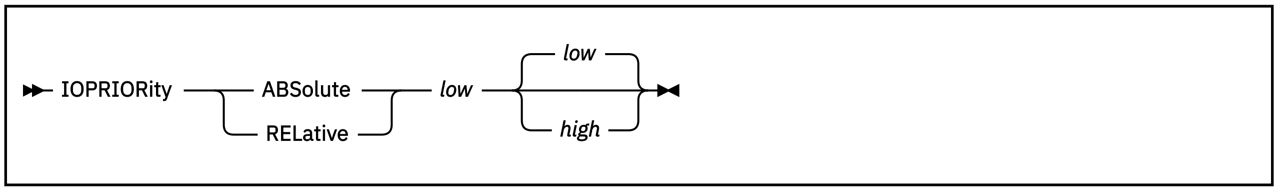
1. To specify PROFILE1 as a profile entry to be invoked as part of the USER statement, use the following statements:

```
User example ...
Include profile1
 .
 .
 .
```

2. To tailor a user's profile selection by system in a non-SSI multisystem complex, use the following statements:

```
User example ...
Sysaffin vma
Include d123vma
Sysaffin vmb
Include d123vmb
 .
 .
 .
```

IOPRIORITY Directory Statement



Purpose

The IOPRIORITY statement defines a virtual machine's I/O priority queueing range.

How to Specify

One IOPRIORITY statement is allowed in a user, identity, or profile entry. An IOPRIORITY statement in a user or identity entry overrides an IOPRIORITY statement in a profile entry.

If you specify the IOPRIORITY statement, it must precede any device statements that you specify in a profile, user, or identity entry. (For a list of device statements, see Table 24 on page 462.)

Operands

ABSolute
indicates the type of I/O priority range. If the LPAR I/O Priority Facility is enabled, the I/O priority range must fit onto the range available to CP. If the range specified falls outside of the range available to CP, an informational message is displayed during LOGON processing, and the requested range is clipped to fall within the range available to CP.

If LPAR I/O Priority Facility is not enabled or not installed, CP simulates an I/O priority range from 0 to 255.

RELative
indicates the type of I/O priority range. The maximum I/O priority range is 0 to 255. If LPAR I/O Priority Facility is installed and enabled, the maximum I/O priority range maps proportionately onto the range available to CP, with the highest I/O priority mapping directly to the highest value available to CP. The requested RELATIVE I/O priority range is used to calculate a percentage of the maximum I/O priority range, and the user's effective I/O priority range is the corresponding percentage of the I/O priority range available to CP.

If LPAR I/O Priority Facility is not enabled or not installed, CP simulates an I/O priority range from 0 to 255. Thus, the maximum I/O priority range maps directly onto the range simulated by CP.

low
is the low value of the I/O priority range. It must be a number from 0 to 255.

high
is the high value of the I/O priority range. It must be a number from 0 to 255 and greater than or equal to the low value. If not specified, the high value is equal to the low value.

Usage Notes

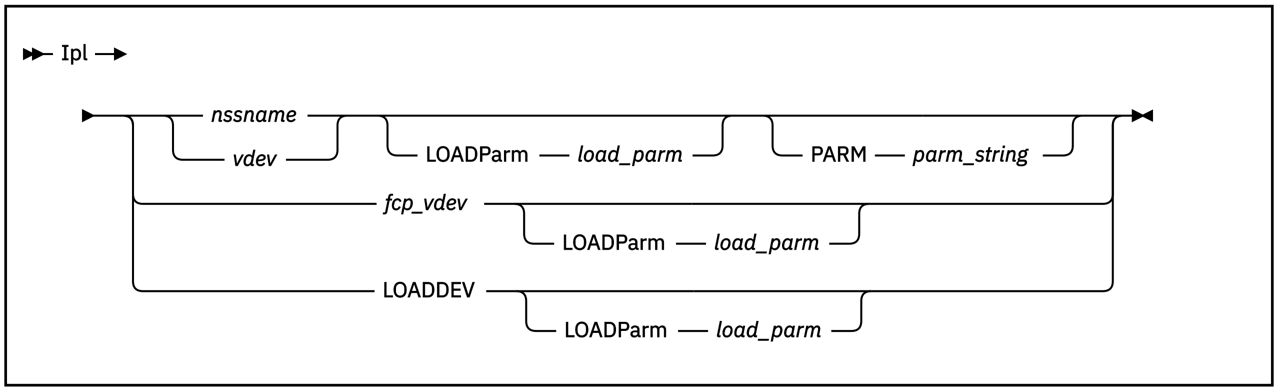
1. If an IOPRIORITY statement is not found, LOGON processing requests RELATIVE I/O priority range with a low value of 0 and a high value of 0.
2. For more information on IOPRIORITY ranges, see *SET IOPRIORITY*

Examples

1. To assign a RELATIVE range of 8 values, use the following IOPRIORITY statement in a virtual machine definition:

```
iopriorr rel 0 7
```

IPL Directory Statement



Purpose

The IPL statement provides information to load an operating system program when a user logs on a virtual machine. IPL statements can specify four methods to load the operating system program:

- Load a named saved system.
- Use an ECKD device that is specified by the IPL statement. Optionally specify ECKD device parameters.
- Use a SCSI device that is specified by the IPL statement's *fcp_vdev* operand and use required information that is specified in LOADDEV directory statements. A list-directed IPL is initiated.
- Use required IPL information, including the virtual device number, that is specified in LOADDEV directory statements. The device can be an ECKD device or a SCSI device. A list-directed IPL is initiated.

The IPL statement can specify optional IPL information for any of the four methods.

How to Specify

The IPL statement is allowed in profile, user, identity, and subconfiguration entries. An IPL statement in a user or identity entry overrides one in a profile entry. An IPL statement in a subconfiguration entry overrides one in an identity entry.

If you specify the IPL statement, it must precede any device statements in a directory entry. (For a list of device statements, see Table 24 on page 462 in “Directory Statement Categories” on page 462.)

Operands

nssname
The *nssname* operand specifies the 1- to 8-character alphanumeric name of a named saved system that CP automatically loads when the user logs on. The system was saved by using the **SAVESYS** command.

vdev
The *vdev* operand specifies the virtual device number of the device that CP automatically IPLs when the user logs on. The value must be a hexadecimal number in the range X'0000' - X'FFFF'.

Note: DASD that is defined as being unsupported cannot be IPLed by CP. Unsupported DASD can be used only by a virtual machine that is IPLed from a supported device or from a named saved system. See Direct access storage devices (DASD) in *z/VM: General Information*.

fcp_vdev
The *fcp_vdev* operand initiates a list-directed IPL operation from an FCP-attached (SCSI) device when the user logs on. The *fcp_vdev* operand specifies the virtual device number of the attached FCP device.

The *fcp_vdev* operand indicates a list-directed IPL, in which some required IPL information is specified by LOADDEV directory statements. For more information, see usage notes "9" on page 521 and "10" on page 523.

If a LOADDEV directory statement identifies the virtual device number of the device to IPL (specified by the DEVICE operand), you can omit the IPL directory statement's *fcp_vdev* operand and instead use the IPL directory statement's LOADDEV operand. If both the IPL directory statement and the LOADDEV directory statement specify a virtual device number, they must both specify the same virtual device number.

The *fcp_vdev* operand is not allowed if a LOADDEV directory statement specifies the SECURE parameter.

LOADDEV

The LOADDEV operand initiates a list-directed IPL operation from an FCP-attached (SCSI) device or an ECKD device when the user logs on. All parameters that are required for IPL, including the device number, must be set by using LOADDEV directory statements. See "LOADDEV Directory Statement" on page 540.

A secure IPL at logon is possible only when a LOADDEV directory statement specifies the SECURE parameter and the **IPL** command specifies the LOADDEV operand.

PARM *parm_string*

specifies a parameter string to pass to the user's virtual machine in general-purpose registers at the completion of the IPL. When you specify PARM, all characters that follow the statement are considered part of the parameter string. Therefore, you must code the PARM option last on the IPL statement. The **IPL** command allows for 64 bytes of parameters. However, the string on the directory statement is limited to the number of characters that can be specified in the first 71 positions of the statement. The string begins with the first nonblank character in the statement. (This behavior differs from the **IPL** command, where trailing blanks are included.)

If you specify an IPL statement that uses a virtual device number, the parameter string is inserted into the virtual machine registers, 4 bytes per register, starting with register 0. One byte of binary 0's is inserted after the string. If PARM is specified without a parameter string (only blanks), byte 0 of register 0 contains binary 0's. If you omit PARM, the virtual machine's registers remain unchanged.

If you specify an IPL statement by using a named saved system that was defined with the PARMREGS=*m*-*n* operand of the CP **DEFSYS** command, the parameter string is inserted into the virtual machine registers *m* through *n* that are first initialized to binary 0's. For more information, see DEFSYS in *z/VM: CP Commands and Utilities Reference*. If you enter a string larger than can fit in the designated registers, an error message is issued during LOGON and the **IPL** command is rejected. If you omit PARM or specify it with no parameter string, the virtual machine registers contain all 0's.

If you specify an IPL statement by using a named saved system that was defined with the PARMREG=NONE operand on the CP **DEFSYS** command, any parameter string causes an error message to be issued and the command to be rejected.

If you specify an IPL statement using a named saved system that was defined without the PARMREGS=*m*-*n* operand on the CP **DEFSYS** command, the parameter string is inserted into virtual machine registers 0 through 15. The registers are not initialized first to binary 0's. If you omit PARM or specify it with no parameter string, the virtual machine's registers remain unchanged.

LOADParm *load_parm*

The LOADPARM operand specifies a load parameter of 1-8 characters. The parameter value can be retrieved by the guest operating system during the IPL sequence. The use of the parameter is determined by the guest operating system, such as indicating the nucleus to be loaded during the IPL sequence.

Use single quotation marks to preserve leading and embedded blanks. Two consecutive interior single quotation marks is formatted as a single quotation mark. For example, the operand `'BETSY''S'` yields a parameter of BETSY'S.

The parameter is preserved until a system-reset-clear operation is completed on the virtual machine, which resets the parameter to 8 EBCDIC blanks.

Usage Notes

1. Although the CP **IPL** command allows up to 64 characters on the PARM option, the directory restricts each statement to a single card image. This restriction limits the number of characters that you can enter on the IPL statement in the directory.
2. If a user IPLs a named saved system that was created with the VMGROUP option on the CP **DEFSYS** command, that user's virtual machine becomes a member of the virtual machine group that is known by the NSS name. Members of a virtual machine group can connect to the signal system service to provide signalling within the group. Signaling can include awareness messages (signal-in and signal-out) about members joining the group or leaving it.
3. A user whose virtual machine is a member of a virtual machine group and has the appropriate privilege class can authorize trace data recording to a system data file for the group.
4. For more information on the **IPL** command (including information about the LOADPARM and PARM options), see IPL in *z/VM: CP Commands and Utilities Reference*.
5. For more information on the LOADDEV parameters that can be set by using the LOADDEV directory statement, see "LOADDEV Directory Statement" on page 540
6. The IPL action might not be completed if the user does not have certain privilege classes. For example, if IPL CMS is coded in a nonclass G virtual machine definition, the IPL fails because CMS issues class G CP commands that use DIAGNOSE code X'08'.
7. Specifying the load parameter in single quotation marks gives you leading blanks, embedded blanks, or single quotation marks. Remember that any single quotation mark that is a part of the load parameter must be doubled. For example, use 'BETSY''S' to specify BETSY'S as a load parameter.
8. You can specify both the LOADPARM and PARM options when IPLing by either *vdev* or *nssname*. The PARM option must be the last option on the IPL statement.
9. A list-directed IPL is an IPL that is initiated when an **IPL** command or IPL directory statement specifies one of the following operands:

   - *fcp_vdev*
   - LOADDEV
   - DUMPDEV (DUMPDEV is not a valid operand for IPL directory statements)

   In these cases, the **IPL** command or IPL directory statement operands do not provide all required information for the IPL. List-directed IPL parameters (LOADDEV and DUMPDEV parameters) provide the IPL information that the **IPL** command or IPL directory statement operands don't provide.

   Most list-directed IPL parameters specify information that cannot be entered as operands on the **IPL** command or IPL directory statement. (The one exception is the DEVICE parameter, which value can be specified by the *fcp_vdev* operand of the **IPL** command or IPL directory statement.)

   Table 28 on page 521 lists parameters that are used in list-directed IPLs.

Table 28. LOADDEV and DUMPDEV parameters

Parameter	Description	Device type	Default value
DEVICE	The number of the virtual device that is IPLed	SCSI and ECKD	There is no default value
PORTNAME	The fibre channel port name of the device that is IPLed	SCSI only	0[1]
LUN	The logical unit number of the device that is IPLed	SCSI only	0[1]
BR_LBA	The logical-block address of the boot record	SCSI only	0
BOOTPROG	The number of the boot program	SCSI and ECKD	0

Table 28. LOADDEV and DUMPDEV parameters (continued)			
Parameter	**Description**	**Device type**	**Default value**
SECURE/ NOSECURE	Specifies whether the IPL is a secure IPL	SCSI and ECKD	NOSECURE
ALTERNATE	Specifies the device number and fibre-channel port name of an alternate path to the IPL device.	SCSI only	There is no default value
SCPDATA	Information that is passed to the program that is loaded during IPL	SCSI and ECKD	There is no default value
BOOTREC	The boot record location on the device that is IPLed	ECKD only	LABEL

Note:

1. The default value 0 is not a valid device port name and not a valid device LUN. An IPL attempt that uses the default value for the PORTNAME or LUN parameter will fail.

Table 29 on page 522 shows which LOADDEV and DUMPDEV parameters have default values that must be set (changed from the default) before each type of list-directed IPL can be started. Other parameters might need to be set if the default values are not appropriate for your environment.

Table 29. LOADDEV and DUMPDEV parameters requirements by type of list-directed IPL				
IPL type	**DEVICE parameter**	**PORTNAME parameter**	**LUN parameter**	**SECURE parameter**
IPL *fcp_vdev*	The DEVICE parameter is not set or matches *fcp_vdev* on the **IPL** command	Required	Required	Not allowed
IPL *fcp_vdev* DUMP				
IPL LOADDEV from SCSI device	Required	Required	Required	Required if the SECUREIPLREQ option is specified in the OPTION directory statement
IPL LOADDEV from ECKD device	Required	Not applicable	Not applicable	
IPL DUMPDEV from SCSI device	Required	Required	Required	Required if the previous IPL used the SECURE parameter or if the SECUREIPLREQ option is specified in the OPTION directory statement
IPL DUMPDEV from ECKD device	Required	Not applicable	Not applicable	

There are two groups of parameters for list-directed IPLs:

LOADDEV parameters
LOADDEV parameters provide IPL specifications when the IPL does not include a dump.

DUMPDEV parameters
DUMPDEV parameters provide IPL specifications when the IPL does include a dump.

In each group, a value can be set for each parameter that is listed in Table 28 on page 521. The values of the LOADDEV and DUMPDEV parameters are independent of each other and do not affect each other. The settings for LOADDEV parameters are displayed by the **QUERY LOADDEV** command. The settings for DUMPDEV parameters are displayed by the **QUERY DUMPDEV** command.

For more information, see QUERY LOADDEV and QUERY DUMPDEV in *z/VM: CP Commands and Utilities Reference*.

10. Load parameters for a list-directed IPL at logon are set by using the LOADDEV directory statement.

    The LOADDEV directory statement can be modified by a directory management tool such as IBM Directory Maintenance (DirMaint). To set load parameters by using DirMaint, use the **DIRM LOADDEV** command. The LOADDEV user directory statement can be modified also by the Image_IPL_Characteristics_Define_DM API function.

    For more information, see "LOADDEV Directory Statement" on page 540, **DIRM** LOADDEV in *z/VM: Directory Maintenance Facility Commands Reference*, and Image_IPL_Characteristics_Define_DM API function in *z/VM: Systems Management Application Programming*.

11. A secure IPL is required if a user's OPTION directory statement includes the SECUREIPLREQUIRED option.
12. To initiate a secure IPL at logon, the following conditions must be satisfied:
    - The IPL directory statement uses the LOADDEV operand.
    - A LOADDEV directory statement specifies the virtual device number of the device to IPL.
    - A LOADDEV directory statement specifies the SECURE operand.
    - LOADDEV directory statements specify appropriate values for other load parameters for your environment.
    - The program that is loaded must be signed. The appropriate certificate must be loaded in the HMC and assigned to the LPAR on which the IPL occurs.
13. Up to three alternate paths can be defined as part of a SCSI LOADDEV or DUMPDEV specification. For a SCSI list-directed IPL that is invoked with either the LOADDEV or DUMPDEV operand, if IPL is not successful for the primary device path, then the alternate device paths are automatically considered:
    - If a device is not available, then the next alternate path is considered.
    - Alternate device paths are considered in turn until IPL is successful or all device paths are considered and IPL is not successful.

    Alternate paths are not attempted for list-directed IPLs that are invoked by specifying the virtual device number (**IPL** *fcp_vdev*). For more information, see SET LOADDEV and SET DUMPDEV.

Examples

1. To specify that CP is to automatically IPL virtual device number 490 for a virtual machine, use the following IPL statement in the virtual machine definition:

   ```
   Ipl 490
   ```

2. To specify that CP is to automatically IPL a named saved system called CMS for a virtual machine, use the following IPL statement in the virtual machine definition:

   ```
   Ipl cms
   ```

3. To specify that CP is to automatically IPL virtual device number 490 for a virtual machine and pass parameters XYZ to that virtual machine, use the following IPL statement in the virtual machine definition:

   ```
   Ipl 490 Parm xyz
   ```

4. To specify that CP is to automatically IPL virtual device number 490 for a virtual machine and pass parameters XYZ to the virtual machine and load parm data of BETSY'S, code the following statement in the virtual machine definition:

   ```
   Ipl 490 LOADParm 'betsy''s' Parm xyz
   ```

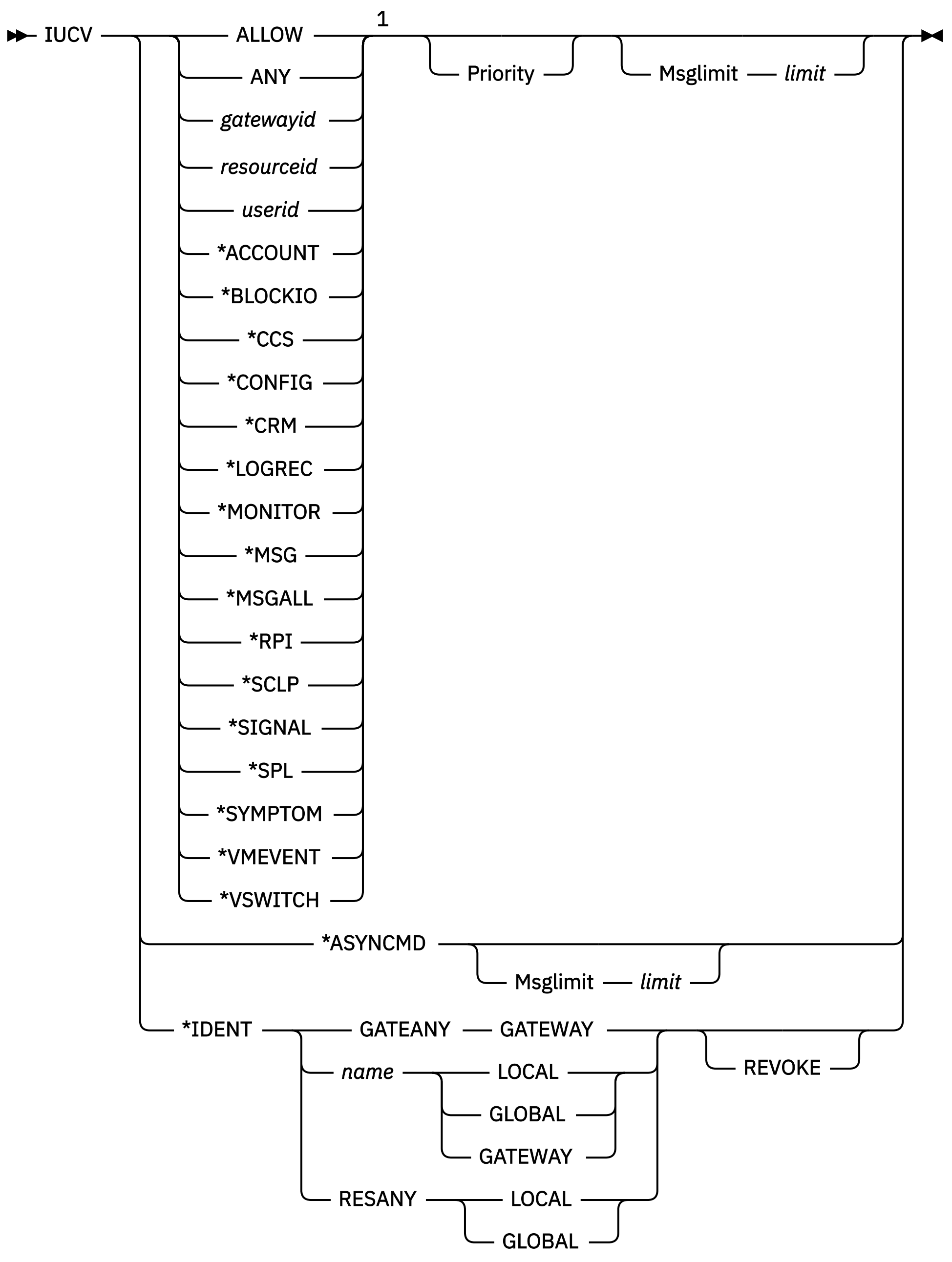
5. To specify that CP is to automatically initiate a secure IPL of FCP device number 1234 by using the parameters that are specified on LOADDEV statements, use the following user directory statements in the virtual machine definition. Because SCSI is the default device type, the SCSI keyword is not required.

```
LOADDEV DEVICE 1234
LOADDEV PORTNAME 1111111122222222
LOADDEV LUN 3333333344444444
LOADDEV BOOTPROG 1
LOADDEV BR_LBA 10
LOADDEV SECURE
Ipl LOADDEV
```

6. To specify that CP is to automatically initiate a secure IPL of ECKD device number 5678 by using the parameters that are specified in LOADDEV statements, use the following user directory statements in the virtual machine definition. The ECKD keyword must be specified on every statement.

```
LOADDEV ECKD DEVICE 5678
LOADDEV ECKD BOOTREC LABEL
LOADDEV ECKD BOOTPROG AUTO
LOADDEV ECKD SECURE
Ipl LOADDEV
```

IUCV Directory Statement



Notes:

[1] The PRIORITY and MSGLIMIT operands can be specified in any order.

Purpose

The IUCV statement authorizes a virtual machine to create an inter-user communications vehicle (IUCV) communication path with another virtual machine or CP system service, or to create APPC/VM communication paths. (To communicate with itself, a virtual machine does not need an IUCV statement in its virtual machine definition.) For CP system services which do not require authorization, this statement can be used to define the message limit.

You can pass information between virtual machines with IUCV. IUCV allows virtual machines to send and receive any amount of data to and from other virtual machines. To use IUCV, you must specify the IUCV statement for the virtual machines that use IUCV. For more information on IUCV, see *z/VM: Connectivity*.

How to Specify

The IUCV statement is allowed in profile, user, and identity entries. If you specify the IUCV statement, it must precede any device statements in an entry. (For a list of device statements, see Table 24 on page 462.)

You can specify multiple IUCV statements within each entry. Any IUCV statements within a profile entry are added to those in the including user entry with no duplicate checking performed.

Operands

ALLOW
specifies that any other virtual machine can establish a communication path with this virtual machine. No further authorization is required in the virtual machine that initiates the communication.

ANY
is a general authorization indicating that a communications path can be established with any other virtual machine, local resource, system resource, global resource, or gateway residing on this VM/ESA system. This option does *not* indicate that a communication path can be established with a CP system service.

gatewayid
is a 1-character to 8-character gateway name used to connect to the resources in the SNA network rather than to a specified virtual machine. The first byte of the gateway name must be alphanumeric. (IBM reserves the names beginning with non-alphanumeric characters for its own use.)

Be sure that the gateway name you specify is not the same as a user ID or resource name on the system, ALLOW, ANY, or SYSTEM.

resourceid
is a 1-character to 8-character resource identifier used to connect to a resource manager rather than to a specified virtual machine. The first byte of the name must be alphanumeric. (IBM reserves names beginning with the remaining characters for its own use.)

Be sure you do *not* specify the same name as a user ID or gateway on the system; or as ALLOW, ANY, or SYSTEM.

Specifying IUCV *resourceid* does not give authority to connect by user ID to the virtual machine that owns the specified resource. At the same time, specifying IUCV *userid* does not give authority to connect by *resourceid* to the specified virtual machine.

When you explicitly authorize each virtual machine (with IUCV *name* or IUCV ANY), you should also give explicit directory authorization to the TSAF virtual machine residing on the same system as the name (with IUCV *name* or IUCV ANY).

userid
is the z/VM user ID of the virtual machine to which an IUCV communication path is authorized.

***ACCOUNT**
is the Accounting system service, which handles sending accounting records to authorized virtual machines. For further information, see Account System Service (*ACCOUNT) in *z/VM: CP Programming Services*.

***ASYNCMD**
is the Asynchronous CP Command Response system service, which handles capturing CP command responses as a result of issuing the FOR command. For further information, see FOR command in *z/VM: CP Commands and Utilities Reference* and Asynchronous CP Command Response system service (*ASYNCMD) in *z/VM: CP Programming Services*.

***BLOCKIO**
: is the DASD Block I/O system service, which provides a virtual machine with device-independent, asynchronous access to its CMS-formatted virtual disk devices. For further information, see DASD Block I/O System Service (*BLOCKIO) in *z/VM: CP Programming Services*.

***CCS**
: is the Console Communications Services system service. The VTAM service machine (VSM) must be authorized for this system service to use IUCV communication with SNA/CCS. For information on VSM and SNA/CCS, see Chapter 9, "Planning for SNA Console Communication Services (SNA/CCS)," on page 367. This function is reserved for IBM use.

***CONFIG**
: is the system service used by a virtual machine running the Enterprise Systems Connection Manager (ESCM) licensed program. The function is reserved for IBM use.

***CRM**
: is the Collection Resource Management system service. This function is reserved for IBM use.

***IDENT**
: allows the virtual machine to connect to the Identify System Service to identify a resource or gateway LU. For further information, see Identify System Service (*IDENT) in *z/VM: CP Programming Services*.

***LOGREC**
: is the Error Logging system service, which handles sending error log records to authorized virtual machines. For further information, see Error Logging System Service (*LOGREC) in *z/VM: CP Programming Services*.

***MONITOR**
: is the Monitor system service, which notifies connected virtual machines when records are created by the z/VM monitor. For further information see *z/VM: Performance*.

***MSG**
: is the Message system service, which allows a virtual machine read messages and responses that come from CP, rather than display them on the terminal. For further information, see Message System Service (*MSG) in *z/VM: CP Programming Services*.

***MSGALL**
: is the Message All system service, which acts as an alternative to the Message system service, where it will collect all terminal messages and responses. For further information, see Message All System Service (*MSGALL) in *z/VM: CP Programming Services*.

***RPI**
: is the Access Verification system service, which handles IUCV communications between the CP access control interface (ACI) and an external security manager (ESM) service virtual machine. For further information, see the Access Verification System Service (*RPI) in *z/VM: CP Programming Services*.

***SCLP**
: is the SCLP system service, which provides an IUCV communication interface between CP and virtual machines for IBM Hardware Management Console (HMC) event management. For further information, see SCLP System Service (*SCLP) in *z/VM: CP Programming Services*.

***SIGNAL**
: is the Signal system service, which allows a virtual machines in a virtual machine group to signal each other. For further information, see Signal System Service (*SIGNAL) in *z/VM: CP Programming Services*.

***SPL**
: is the Spool system service, which provides a communication interface using IUCV between CP and virtual machines for print services, reader services, and reader notification. For further information, see Spool System Service (*SPL) in *z/VM: CP Programming Services*.

***SYMPTOM**
: is the Symptom system service, which handles sending symptom records to authorized virtual machines. For further information, see Symptom System Service (*SYMPTOM) in *z/VM: CP Programming Services*.

***VMEVENT**
: is the VM Event system service, which provides an IUCV communication interface between CP and virtual machines for event notification. For further information, see VM Event System Service (*VMEVENT) in *z/VM: CP Programming Services*.

***VSWITCH**
: is the virtual switch system service, which allows communication between CP and the TCP/IP for z/VM stack in order to provide device status and other information. This function is reserved for IBM use.

Priority
: indicates that a communication path with the specified virtual machine can handle priority IUCV communications and IUCV communications that have no priority. If you do not specify PRIORITY, paths authorized by this entry cannot handle priority messages. PRIORITY does not apply to APPC/VM paths. For any communication path routed through ISFC, PRIORITY status is determined by the IUCV directory statement for the initial invoker of the IUCV CONNECT.

  Note: PRIORITY is ignored for the *MSG, *MSGALL, and *ASYNCMD system services because the virtual machine(s) will only receive messages.

Msglimit *limit*
: defines the maximum number of outstanding messages allowed on any path authorized by this entry. If you omit MSGLIMIT or if the IUCV CONNECT and ACCEPT functions specify a lower message limit, the message limit is taken from the CONNECT or ACCEPT parameter list. If you omit MSGLIMIT and the IUCV CONNECT and ACCEPT functions do not specify a message limit, the default value is 10. The maximum allowed value for MSGLIMIT is 65,535. MSGLIMIT does not apply to APPC/VM paths.

  For further information on message limits, see *z/VM: CP Programming Services*.

GATEANY
: allows the virtual machine to identify any gateway LU name.

  Note: Be careful when you assign names and when you give authorization for GATEANY. A virtual machine that has authority for GATEANY can identify a gateway LU name as *gateany*. Also, this virtual machine would be authorized to identify any other gateway LU name.

name
: is a 1- to 8-character resource or gateway LU name that the virtual machine is authorized to identify. The first byte of the name should be alphanumeric. (IBM reserves names beginning with the remaining characters for its own use.)

  Be sure that the name you specify is not the same as a user ID on the system. Also, do not specify the name as any of the following: ALLOW, ANY, or SYSTEM.

  Specify LOCAL or GLOBAL for the next operand if the name corresponds to a resource managed by the virtual machine. Specify GATEWAY as the next parameter if the name corresponds to a gateway LU provided by the virtual machine.

RESANY
: allows the virtual machine to identify any resource name.

  Note: Be careful when you assign names and when you give authorization for RESANY. A virtual machine that has authority for RESANY can identify a resource name as *resany*. Also, this virtual machine would be authorized to identify any other resource name.

LOCAL
: authorizes the virtual machine to identify the resource as a local resource known only to the local system. If you specify LOCAL with RESANY, the virtual machine can identify any resource as a local resource.

GLOBAL
: authorizes the virtual machine to identify the resource as a global resource known to all systems in the collection. This operand allows the virtual machine to identify the resource as local also. If you specify GLOBAL with RESANY, the virtual machine can identify any number of local or global resources.

GATEWAY
authorizes the virtual machine to identify the gateway LU. If you specify GATEANY as the system gateway name, the virtual machine can identify any gateway LU in the collection.

REVOKE
authorizes the virtual machine to revoke the specified resource or gateway LU name without owning it. A resource manager can always revoke ownership of resource that it owns by severing its session with the *IDENT system service. A virtual machine that can revoke resources or gateway LUs can also identify them.

If you specify REVOKE with:

- LOCAL, the virtual machine can revoke and identify the specified resource only on this z/VM system.
- GLOBAL, the virtual machine can revoke and identify:
  - The specified global resource
  - A local or system resource on this z/VM system that has the same name as the global resource.

  A virtual machine cannot revoke a global and local resource at the same time. The virtual machine must specify which resource to revoke when the connection is made to *IDENT.
- GATEWAY, the virtual machine can revoke and identify the gateway LU anywhere within the collection.
- GATEANY, the virtual machine can revoke and identify any gateway LU in the collection. Since gateway LUs are always known throughout the collection, there is no need for the LOCAL or GLOBAL authority keywords.
- RESANY and LOCAL, the virtual machine can revoke and identify any local resource.
- RESANY and GLOBAL, the virtual machine can revoke and identify any resource, local or global.

Because the TSAF virtual machines do not keep track of the local resources, a virtual machine cannot revoke a local resource on another system.

Usage Notes

1. When a virtual machine invokes the IUCV CONNECT function, IUCV searches virtual machine definitions in the following order:
   a. The invoker's IUCV statements for the target's user ID
   b. The invoker's IUCV statements for an ANY virtual machine definitions
   c. The target's IUCV statements for an ALLOW virtual machine definitions.

   Connections invoked from CP system code do not need directory authorization. Priority status and message limit are taken from the CONNECT parameter list.
2. For information on how to specify the maximum number of IUCV connections a virtual machine can make, see "OPTION Directory Statement" on page 572.
3. For information on how to code IUCV programs, see *z/VM: CP Programming Services*.
4. Authorization using an IUCV statement is required in order to communicate with certain system services. Table 30 on page 529 indicates which system services require authorization.

Table 30. System services requiring authorization

System Service	Authorization Required
*ACCOUNT (Accounting)	Yes
*ASYNCMD (Asynchronous CP Command Response)	No
*BLOCKIO (DASD Block I/O)	No
*CCS (Console Communication Services)	Yes
*CONFIG (Enterprise Systems Connection Manager)	Yes

Table 30. System services requiring authorization (continued)

System Service	Authorization Required
*CRM (Collection Resource Management)	Yes
*IDENT (Identify)	Yes
*LOGREC (Error Recording)	Yes
*MONITOR (MONITOR)	Yes
*MSG (Message)	No
*MSGALL (Message All)	No
*RPI (Access Verification)	Yes
*SCLP	Yes
*SIGNAL (Signal)	No
*SPL (Spool)	Yes
*SYMPTOM (Symptom)	Yes
*VMEVENT	Yes
*VSWITCH (Virtual Switch)	Yes

Notes:

a. Authorization using an IUCV *CCS statement is required for a virtual machine running the Enterprise Systems Connection Manager (ESCM) licensed program and when communicating using SNA/CCS.

b. Authorization using an IUCV *MONITOR statement is required for a virtual machine to receive information concerning the location of monitor records in a saved segment for monitor. Without *MONITOR on the IUCV statement, a user cannot issue IUCV CONNECT to *MONITOR.

   After a virtual machine is connected, it receives notifications from IUCV SEND commands whenever data is in the monitor saved segment. The IPARML points to the guest real addresses in the saved segment where the monitor control area lies. The control area points to each data area.

   More than one virtual machine can have an IUCV *MONITOR statement. If so, every virtual machine with an IUCV *MONITOR statement can connect concurrently to *MONITOR.

c. Before communications can be established with the *RPI system service, an external security manager must be installed on the system.

5. When the virtual machine definition associated with a resource manager user ID contains *IDENT authority to identify or revoke any resources or gateway LUs, this information is saved in storage and checked in place of a subsequent directory search. This reduces the number of directory searches required when a resource manager virtual machine identifies or revokes resources (thereby improving performance). This means that if *IDENT authority is updated, it is not recognized until the resource manager stops using IUCV support and restarts.
6. If a virtual machine must identify more than one resource, you can specify more than one IUCV statement for *IDENT in the virtual machine definition. To check whether a virtual machine is authorized to identify or revoke the resource or gateway LU named in the IUCV CONNECT request presented to *IDENT, the Identify System Service searches the virtual machine definition for IUCV statements in the following order:

   a. The first *IDENT entry that has either the resource name RESANY or the gateway LU name GATEANY (depending on the type of name). If *IDENT finds a match, it checks whether the LOCAL or GLOBAL (for resources) and REVOKE operands have enough authority to do the requested function.

b. If it does not find a match, or if the other operands are not enough authority, *IDENT searches for the first *IDENT entry that has the same resource name or gateway LU name as specified on the CONNECT request.

c. If *IDENT does not find a match, or if it finds a match but authorization for the LOCAL or GLOBAL (for resources) and REVOKE operands does not correspond to that specified in the CONNECT parameter list, *IDENT severs the requested connection, and the resource or gateway LU is not identified.

Examples

1. To indicate that a virtual machine can establish an IUCV communication path to SOURCE, use the following statement in the virtual machine definition:

   ```
   Iucv source
   ```

2. To indicate that a virtual machine can establish both of the following:

   a. An IUCV communication path to SOURCE

   b. A priority IUCV communication path with a message limit of 100 to TARGET

   use the following statements in the virtual machine definition:

   ```
   Iucv source
   Iucv target Priority MsgLimit 100
   ```

3. To indicate that a virtual machine can establish:

   a. A priority IUCV communication path with a message limit of 100 to TARGET

   b. IUCV communication paths with any other virtual machine

   use the following statements in the virtual machine definition:

   ```
   Iucv target Priority MsgLimit 100
   Iucv Any
   ```

4. To indicate that any virtual machine can establish a path to the virtual machine whose virtual machine definition you are creating, use the following statement:

   ```
   Iucv Allow
   ```

5. The following examples show how to use multiple IUCV statements for *IDENT.

   a. A resource manager, RESMGR1, has the following IUCV statements:

      ```
      Iucv *Ident ResAny Global
      Iucv *Ident residx Local Revoke
      ```

      The first IUCV statement authorizes RESMGR1 to identify any resource (RESANY) as a local or global resource (GLOBAL). This includes the local resource, RESIDX. The second IUCV statement authorizes RESMGR1 to revoke (REVOKE) the resource RESIDX, if it is defined on the local system as a local resource (LOCAL).

   b. A resource manager, RESMGR2, has the following IUCV statements:

      ```
      Iucv *Ident Gateany Gateway
      Iucv *Ident gatewayx Gateway Revoke
      ```

      The first IUCV statement authorizes RESMGR2 to identify any gateway LU. This includes the gateway LU, GATEWAYX. The second IUCV statement authorizes RESMGR2 to revoke the gateway LU GATEWAYX, but no other gateway LUs.

      Because gateway LUs are always known throughout the collection, no keyword is needed to authorize a user to identify a gateway LU that can be accessed only by users on the local system.

c. A resource manager, RESMGR3, has the following IUCV statements:

```
   Iucv *Ident residx Global
   Iucv *Ident residy Global
   Iucv *Ident ResAny Local Revoke
```

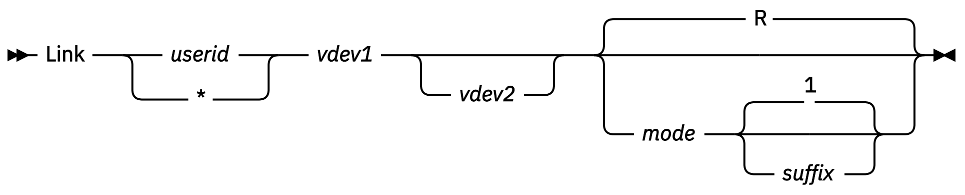
RESMGR3 can identify the resources, RESIDX and RESIDY, as local and global resources. RESMGR3 is not authorized to revoke any global resources. Because of the last IUCV statement, RESMGR3 can identify any resource as a local resource. Also, RESMGR3 can revoke any resource known on the local system as a local resource.

d. A resource manager, RESMGR4, has the following IUCV statements:

```
   Iucv *Ident residx Local
   Iucv *Ident residy Global
   Iucv *Ident residx Global Revoke
```

RESMGR4 can identify the resource RESIDX as a local resource and the resource RESIDY as either a local or global resource. RESMGR4 cannot revoke any resources, because *IDENT searches for the first entry that matches the resource name specified on the CONNECT. If RESMGR4 tries to connect to *IDENT to identify or revoke the GLOBAL resource, RESIDX, *IDENT severs the connection. In this case, if you want RESMGR4 to identify and revoke the global resource, RESIDX, you must delete the first IUCV statement.

LINK Directory Statement



Notes:

[1] The default is a blank.

Purpose

The LINK statement gets access to another user's minidisk. The target minidisk must be defined by an MDISK statement in the target virtual machine definition. For more information, see "MDISK Directory Statement" on page 550. The DASDOPT statement is an extension to the LINK statement. If no DASDOPT statement follows it, the LINK directory statement defaults to NOCTL (which behaves like the LINK command). For more information, see "MDISK Directory Statement" on page 550.

How to Specify

The LINK statement is allowed in profile, user, identity, and subconfiguration entries. If you specify the LINK statement, it must follow any general statements you specify in a directory entry. (For a list of statements by category, see Table 24 on page 462.)

Multiple LINK statements are allowed within an entry. Any LINK statements in a subconfiguration entry are processed first at logon time, followed by LINK statements in a user or identity entry, followed by LINK statements in a profile entry. LINK statements with duplicate virtual device numbers within an entry are errors. No user/identity-to-profile or identity-to-subconfiguration duplicate virtual device number checking is performed.

When processing the LINK statement, DIRECTXA checks for a maximum of five tokens: target user ID, virtual device number (or address) of the target minidisk, virtual device number (or address) for the linking virtual machine, access mode, and a suffix of S or E. If you specify more than five tokens, DIRECTXA ignores the extra tokens and system processing continues as if you had not specified the extra tokens.

Operands

userid
is the virtual machine user ID of the target minidisk owner.

*****
indicates that the user being processed owns the minidisk. When the LINK statement appears in an adjunct template definition, this asterisk refers to the directory entry for the user creating the adjunct.

vdev1
is the virtual device number or virtual address of the target minidisk.

vdev2
is the virtual device number or virtual address that the linking virtual machine uses for the target minidisk. If you omit *vdev2* , the definition of *vdev1* is used.

mode
is the access mode for the minidisk. The first letter in the access mode is the primary access mode (read-only, write, or multiple-write); the second letter (optional) is the alternate access (read-only or write). A suffix letter, S or E, may be added to any of the one or two letter modes to provide this user

specific authorization for use of stable or exclusive link modes when linking the specified minidisk using the LINK command, the CMS VMLINK command, or DIAGNOSE code X'E4'.

The access modes are:

R
Read-only access. Read access is established, unless another user holds a write or an exclusive mode (ER, EW) access to the disk. If you omit the mode, R is the default.

RR
Read-only access. Read access is established, unless another user holds an exclusive mode (ER, EW) access.

W
Write access. Write access is established, unless another user holds access (any mode) to the disk.

WR
Write access. Write access is established unless another user holds access (any mode) to the disk. If write access is denied, read access is established unless another user holds an exclusive (ER, EW) mode access for the disk.

M
Multiple-write access. Write access is established unless another user holds a write, a stable (SR, SW, SM) or an exclusive (ER, EW) mode access to the disk.

MR
Multiple-write access. Write access is established unless another user has a write, stable mode (SR, SW, SM) or exclusive mode (ER, EW) access to the disk. In the case of a previous write or stable access, read-only access is established. In the case of an exclusive mode access existing, read access is also denied.

MW
Multiple-write access. Write access is established in all cases except when another user holds either a stable (SR, SW, SM) or an exclusive mode (ER, EW) access to the disk.

suffix
The optional suffix letters authorize virtual reserve/release and the use of the stable and exclusive (data integrity) access modes of the LINK command and DIAGNOSE code X'E4'. The specification on the LINK statement allows the user to enter a LINK command with the specified type of access mode, stable or exclusive, against the specified user's minidisk. For more information, see LINK in *z/VM: CP Commands and Utilities Reference*.

The suffix letters can be combined as follows (note that CP requires they be specified in this order):

1. S, E, or null (S and E are mutually exclusive), plus
2. D or null.

Therefore, the only valid combinations are: S, E, D, SD, or ED. This suffix is in turn concatenated with the mode, with no intervening blanks. For example, RS, RRE, and MRSD are all valid.

S
Authorizes the virtual machine to use the LINK command stable access modes (SR, SW, SM) against the specified user's minidisk.

E
Authorizes the virtual machine to use the LINK command exclusive and stable access modes (ER, EW, SR, SW, SM) against the specified user's minidisk.

D
Tells CP that the device should not be defined when the virtual machine initially logs on or is autologged, but to defer doing so until an explicit LINK command is issued for that device.

Usage Notes

1. It is the responsibility of the operating system running in each virtual machine to keep data from being destroyed or altered on shared disks.
2. CMS allows many virtual machines to have read access to the same minidisk; however, CMS does not supervise virtual machines that have write access to the same minidisk. If two or more CMS virtual machines have write access to the same disk, all data on the disk may be destroyed.

   Also note that CP does not prevent a virtual machine with an MW access to another virtual machine's minidisk from formatting that minidisk.
3. The use of the stable (SR, SW, and SM) and exclusive (ER and EW) link modes can be authorized in two different ways:
   - Globally, by using the OPTION statement values, LNKSTABL and LNKEXCLU. (For more information, see "OPTION Directory Statement" on page 572).
   - Specifically, by using the mode suffix letters (S or E) on the LINK or MDISK statements. (For more information, see "MDISK Directory Statement" on page 550).
4. Do not add LINK statements to the z/VM directory shared file system file pool minidisks. Shared file system file pool server machines link to the file pool minidisks when needed.
5. Use the LINK statement to give a user direct access to another user's minidisk, or indirect access by linking to a third user's LINK or MDISK statement. You can have up to 50 of these *indirect* links (LINK indirections). For example:

   ```
   USER0
     MDISK 393 3390 1 10 USRVOL ALL ALL ALL
   USER1
     LINK USER0 393 393 RR
   USER2
     LINK USER1 393 393 RR
   USER3
     LINK USER2 393 393 RR
   ```

   Since USER1 has had to first access the 393 minidisk through a direct link to USER0, USER2's link to the 393 minidisk is considered an indirect link. USER3's link to the 393 minidisk through USER2 is thus also an indirect link. Up to 50 of these indirect links are allowed before the following message is issued:

   HCP109E
   userid vdev not linked; excessive LINK indirections

   Note: This limit of 50 indirect links includes the original link. (In the example, USER1 to USER0.)
6. If the target minidisk is a virtual disk in storage that is defined in the specified virtual machine definition but currently does not exist, the virtual disk in storage is created to satisfy the request, as long its creation does not exceed the system limit on the storage available for virtual disks in storage. Note that a new virtual disk in storage must be formatted before it can be used.
7. If a minidisk is defined as virtual device number 192 in the linking virtual machine, the following special rules apply when that virtual machine IPLs CMS:
   - If 192 is an unformatted virtual disk in storage, CMS formats it and accessed it as file mode D.
   - If 192 is a CP-formatted virtual disk in storage, CMS reformats it for CMS use and accesses it as file mode D.
   - If 192 is a CMS-formatted virtual disk in storage or permanent minidisk that is accessed as a file mode other than D, CMS reaccesses it as file mode D.
   - If 192 is an unformatted or CP-formatted permanent minidisk, CMS does not automatically format, reformat, access, or reaccess it.

   When CMS accesses a 192 minidisk as file mode D, any minidisk or SFS directory already accessed as D is released.

8. If the directory definition of a permanent minidisk is changed while users are linked to it, existing links are unchanged, but any new links are made using the new definition of the minidisk. Unpredictable results may occur.
9. If the directory definition of a virtual disk in storage is changed while users are linked to it, existing links are unchanged, and any new links are made to the existing virtual disk in storage. The existing virtual disk in storage is used until the last user detaches it or logs off, at which time the virtual disk in storage is destroyed. After that, any new links use the new definition of the virtual disk in storage.
10. Linking a minidisk where the MDISK statement has been defined with the V mode suffix or WRKALLEG has been specified on the MINIOPT or DASDOPT extension of the MDISK has restrictions For more information, see LINK in *z/VM: CP Commands and Utilities Reference*.

Examples

1. To link in read-only mode LINKMAN's 191 disk to a virtual machine as a 192 disk, use the following LINK statement in the virtual machine definition:

   ```
   Link linkman 191 192 rr
   ```

2. To link in write mode LINKMAN's 291 disk to a virtual machine as a 292 disk, use the following LINK statement in the virtual machine definition:

   ```
   Link linkman 291 292 wr
   ```

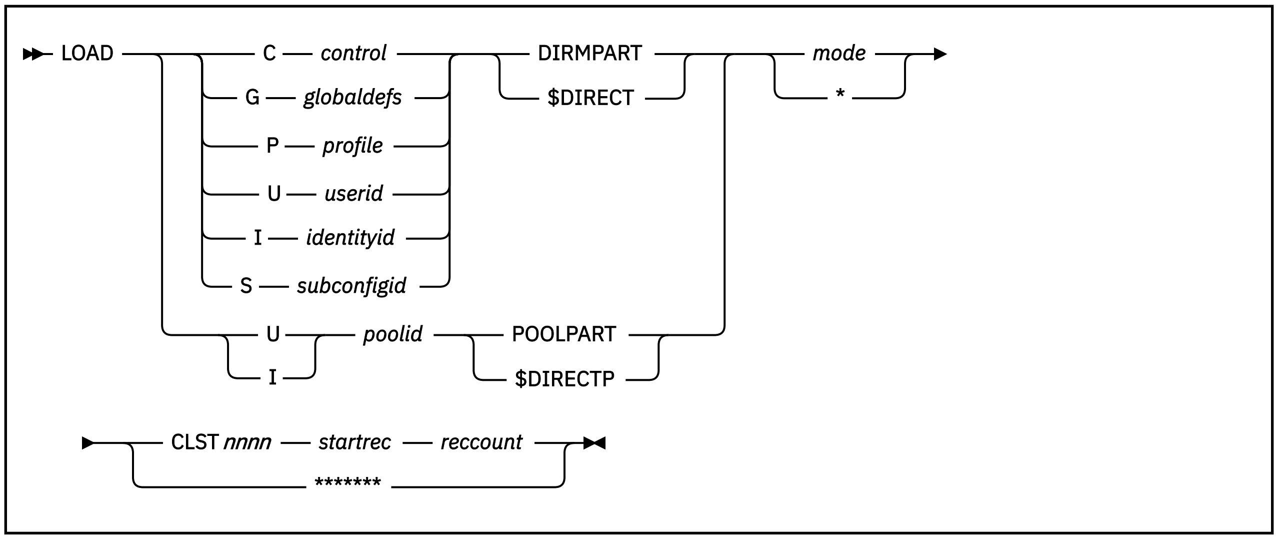
   WR indicates that write access is given only if no one else is linked to the disk; if someone else is linked to the disk, read access is given.

3. To authorize a specific user the ability to enter the LINK command with one of the stable access modes, use the following LINK statement in the virtual machine definition:

   ```
   Link linkman 191 291 rrs
   ```

   RRS ensures the virtual machine a default logon access of read to the specified disk and permits this virtual machine to issue the LINK command with one of the stable modes at a later time. If the virtual machine issues the LINK command specifying the issuer's own directory and the stable read access mode of SR, password-level authorization will not be necessary. However, if a higher level of access (such as SW or SM) is requested, password-level authorization would be required.

LOAD Directory Statement



Purpose

The LOAD statement tells DIRECTXA where the specified directory definition entry is located in a separate file or in a cluster file.

How to Specify

Use this statement only if the entire directory is being maintained in cluster-file format. If it is, this is the only record type found in the USER DIRECT file.

When processing the LOAD statement, DIRECTXA checks for a maximum of five tokens when a separate part file is specified or seven tokens when a cluster file is specified. If you specify more tokens, DIRECTXA ignores the extra tokens and system processing continues as if you had not specified the extra tokens.

Operands

C
G
P
U
I
S
specifies the type of definition to be loaded; C for CONTROL, G for Global definitions, P for PROFILE, U for USER, I for IDENTITY, and S for SUBCONFIG.

control
specifies the file name of the file if the control definition (consisting only of DIRECTORY statements) is in a separate file. Otherwise, this operand is ignored.

globaldefs
specifies the file name of the file containing the global definitions. If the global definitions are not in a separate file, this operand is ignored.

profile
specifies the name of the profile. If the profile definition is in a separate part file, *profile* is also the file name of the file.

userid
specifies the user ID. If the user definition is in a separate part file, *userid* is also the file name of the file.

identityid
specifies the IDENTITY user ID. If the IDENTITY definition is in a separate part file, *identityid* is also the file name of the file. For more information, see "IDENTITY Directory Statement" on page 510.

subconfigid
specifies the SUBCONFIG ID. If the SUBCONFIG definition is in a separate part file, *subconfigid* is also the file name of the file. For more information, see "SUBCONFIG Directory Statement" on page 610.

poolid
specifies the pool ID (the 1- to 3-character user ID used to define the pool). If the pool definition is in a separate part file, *poolid* is also the file name of the file.

DIRMPART
$DIRECT
specifies the file type of the file if the directory definition is in a separate part file; otherwise, this operand is ignored.

POOLPART
$DIRECTP
specifies that the directory definition is for a pool of users. If the directory definition is in a separate part file, the file type of the file must be DIRMPART or $DIRECT (DIRMPART if POOLPART is specified; $DIRECT if $DIRECTP is specified).

mode
*****
specifies the alphabetic portion of the file mode of the separate part file or of the cluster file that contains the directory definition. If the mode is an asterisk (*), CP uses the CMS search order to locate the files.

CLST*nnnn*
****
specifies whether the directory definition entry is located in a separate part file or in a cluster file. If this operand is specified as *******, the directory is located in a separate part file. Otherwise, the directory definition is located in the cluster file identified by CLST*nnnn* CLUSTER *mode*, where *nnnn* is a decimal number from 0000 to 9999.

startrec
specifies the starting record number of the directory definition in the specified cluster file. The number must be greater than 0 and in decimal format.

reccount
specifies the record count of the specified directory definition in the cluster file. The number must be greater than 0 and in decimal format.

Usage Notes

1. The following restrictions apply to cluster format and content:
   - A USER DIRECT file maintained in cluster file format should contain only one type C definition, followed by zero or one type G definitions, followed by zero or more type P definitions, followed by zero or more type U, I, and S definitions. U, I, and S definitions can be intermixed.
   - Each definition in a cluster format source directory should contain only the named definition: the type C should contain only the DIRECTORY statements, the type G should contain only global definition statements, each type P should contain only the named PROFILE definition, each type U should contain only the named USER definition, each type I should contain only the named IDENTITY definition, and each type S should contain only the named SUBCONFIG definition.
   - The first directory statement of each definition should match the definition type. Thus, the first directory statement in a type C definition should be a DIRECTORY statement, the first directory statement in a type G definition should be a GLOBALDEFS statement, the first directory statement in a type P definition should be a PROFILE statement, the first directory statement in a type U definition should be a USER statement, the first directory statement in a type I definition should be an IDENTITY statement, and the first directory statement in a type S definition should be a SUBCONFIG

statement. Note also that comments and blank lines cannot be the first line in a definition. (This restriction is due to the relationship between DIRECTXA and DIRMAINT.)

2. DIRMPART and $DIRECT are the only file types recognized for separate part files in a cluster file installation. The file types DIRMPART and $DIRECT are interchangeable. However, DIRMPART is preferred. $DIRECT is a temporary file type used by DIRMAINT.
3. When you define a pool of users, you must specify POOLPART (or $DIRECTP). If the definition is in a separate part file, the file type of the separate part file must be DIRMPART (or $DIRECT).
4. When you use the z/VM Directory Maintenance Facility (DIRMAINT) to maintain the source directory, you should leave the creation and maintenance of the LOAD statements strictly to DIRMAINT.
5. To migrate a manually created and maintained cluster format source directory to DIRMAINT, it must first be converted to the sequential format.

Examples

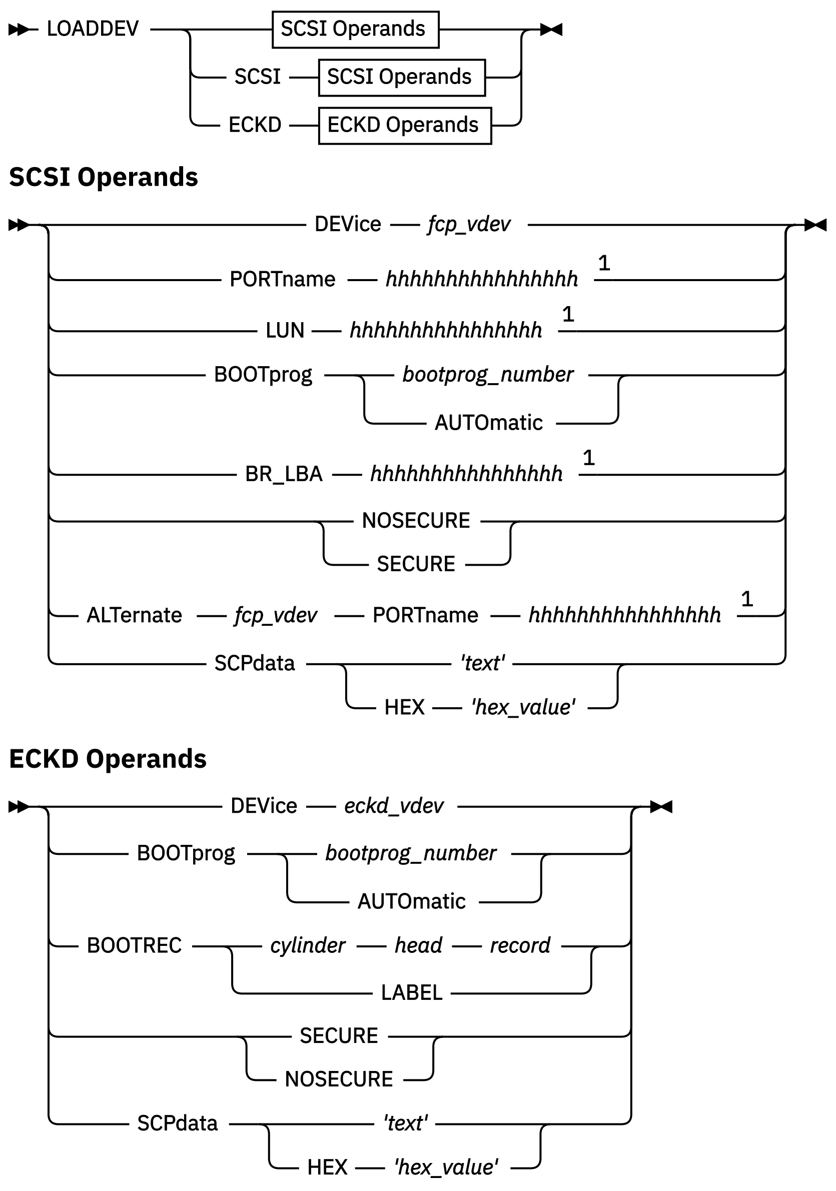
To define a directory in cluster file format with:

- Three POSIXGROUP definitions, GRP0001, GRP0002 and GRP0003, in a separate global definition part file
- Two profiles, DEPT0000 and DEPT0001, where one is in a separate part file and the other is in a cluster file
- Three users, USER0001, USER0002, and USER0003, where two are in a cluster file and the third is in a separate part file
- An IDENTITY and its SUBCONFIGs, for which the IDENTITY and one of the SUBCONFIGs are in separate files but the other SUBCONFIG statement is in the cluster file
- Two pools of users, USR00001 through USR00025, and USR00050 through USR00075, where one is a separate part file and the other is in a cluster file

you might use the following LOAD statements:

```
Load  C  $dirctl$  dirmpart  e  *******
Load  G  glbdefs   dirmpart  e  *******
Load  P  dept0000  dirmpart  e  *******
Load  P  dept0001  dirmpart  e  clst0000 1 20
Load  S  tcpip001  dirmpart  e  *******
Load  U  user0001  dirmpart  e  clst0000 21 5
Load  I  tcpip     dirmpart  e  *******
Load  U  user0002  dirmpart  e  clst0000 26 5
Load  U  user0003  dirmpart  e  *******
Load  S  tcpip002  dirmpart  e  clst0000 97 4
Load  U  usr       poolpart  e  *******
Load  U  usr       poolpart  e  clst0000 31 2
```

LOADDEV Directory Statement



Notes:

[1] The letter '*h*' in the syntax diagram represents a hex character in the range 0 - F. The number of letters indicates the maximum number of hex characters that can be entered (16). Leading zeros are not required.

Purpose

Use the LOADDEV directory statement to set LOADDEV parameters for a guest IPL from SCSI or ECKD disk. The parameters can identify the virtual device number and specify whether the IPL is a secure IPL. The parameters can identify the location of an operating system program and data to pass to the program.

The parameters that are set by using the LOADDEV directory statement are called LOADDEV parameters and are used only for list-directed IPLs, which occur at logon in the following cases:

- The IPL directory statement specifies the *fcp_vdev* operand.
- The IPL directory statement specifies the LOADDEV operand.

See usage note “1” on page 543.

The parameters persist after the virtual machine logs on and can be used by an **IPL** command when the virtual machine is running.

How to Specify

LOADDEV statements are allowed in profile, user, identity, and subconfiguration entries. Each LOADDEV statement can specify the device type (SCSI or ECKD) followed by only one operand and the associated operand value.

If you specify the LOADDEV statement, the LOADDEV statement must precede any device statements that you specify in an entry. (For a list of device statements, see Table 24 on page 462.) Multiple LOADDEV statements are allowed within an entry. Each operand can be specified only once within an entry.

Operands that are unique to ECKD devices (BOOTREC) cannot be specified in the same stanza as operands that are unique to SCSI devices (PORTNAME, LUN, BR_LBA).

Individual operand values on LOADDEV statements in the subconfiguration entry override the corresponding operand values in the identity entry. Values for operands that are specified in the identity entry but not in the subconfiguration entry retain their value. The result is that the LOADDEV is a merging of all settings. Operand values in the subconfiguration entry override the corresponding values in the user or identity entry. Operand values in the user or identity entry override the corresponding values in the profile entry. If conflicting operands are detected during merging, a conflict error results.

Operands

SCSI

The SCSI keyword indicates that the LOADDEV parameters are for IPL of a SCSI device. When the SCSI keyword is specified, only SCSI operands are acceptable on this statement and on other LOADDEV statements for the same user.

SCSI is the default. If the statement does not specify SCSI or ECKD, SCSI is assumed.

ECKD

The ECKD keyword indicates that the LOADDEV parameters are for IPL of an ECKD device. When the ECKD keyword is specified, only ECKD operands are acceptable on this statement and on other LOADDEV statements for the same user.

Because SCSI is the default, you must specify ECKD on all statements that set parameters for an ECKD device.

DEVice *fcp_vdev*

The DEVICE value (*fcp_vdev*) specifies the virtual device number of the FCP device to IPL.

The combination of device number and port name must be unique from those specified in any ALTERNATE path definitions.

If the IPL directory statement specifies the LOADDEV operand, then the LOADDEV DEVICE parameter must be set. A secure IPL at logon from a SCSI device can be initiated only when the IPL directory statement specifies the LOADDEV operand.

If the IPL directory statement specifies the *fcp_vdev* operand, then the LOADDEV DEVICE parameter must match that value or must not be set.

Restrictions and requirements for using the DEVICE parameter are listed in usage note “1” on page 543.

There is no default value for the DEVICE parameter.

DEVice *eckd_vdev*

The DEVICE value (*eckd_vdev*) specifies the virtual device number of the ECKD device to IPL.

If the IPL directory statement specifies the LOADDEV operand, then the LOADDEV DEVICE parameter must be set. A secure IPL at logon from an ECKD device can be initiated only when the IPL directory statement specifies the LOADDEV operand.

Restrictions and requirements for using the DEVICE parameter are listed in usage note "1" on page 543.

There is no default value for the DEVICE parameter.

PORTname *hhhhhhhhhhhhhhhh*
The PORTNAME value (*hhhhhhhhhhhhhhhh*) specifies the 8-byte fibre channel port name of the SCSI device that is IPLed. The value must be a hexadecimal value in the range 0 - FFFFFFFFFFFFFFFF. The value cannot include interior blank space.

The combination of device number and port name must be unique from those specified in any ALTERNATE path definitions.

If you do not specify the PORTNAME operand, the PORTNAME parameter defaults to 0.

LUN *hhhhhhhhhhhhhhhh*
The LUN value (*hhhhhhhhhhhhhhhh*) specifies the 8-byte logical unit number of the SCSI device that is IPLed. The value must be a hexadecimal value in the range 0 - FFFFFFFFFFFFFFFF. The value cannot include interior blank space.

If you do not specify the LUN operand, the LUN parameter defaults to 0.

BOOTprog *bootprog_number*
The *bootprog_number* value specifies the program that is loaded from the SCSI or ECKD device that is IPLed. The value must be a decimal value in the range 0 - 30.

If you do not specify the BOOTPROG operand, the BOOTPROG parameter defaults to 0.

BOOTprog AUTOmatic
The BOOTPROG AUTOMATIC operand loads the first operating system program (not a dump program) that is listed in the program table of the device that is IPLed.

BR_LBA *hhhhhhhhhhhhhhhh*
The BR_LBA value (*hhhhhhhhhhhhhhhh*) specifies the logical-block address of the boot record of the SCSI device that is IPLed. The value must be a hexadecimal value in the range 0 - FFFFFFFFFFFFFFFF. The value cannot include interior blank space.

If you do not specify the BR_LBA operand, the BR_LBA parameter defaults to 0.

BOOTREC *cylinder head record*
The BOOTREC values (*cylinder head record*) specify the boot record location on the ECKD device that is IPLed. Use hexadecimal values and separate the values by blank space. Leading zeros are not required. The following ranges are allowed when you specify the boot record location:

- Cylinder: 0 - FFFFFFF
- Head (or track): 0 - F
- Record: 1 - FF

BOOTREC LABEL
The BOOTREC LABEL operands specify that the boot record is the record that is identified in the volume label of the ECKD device that is IPLed.

If you do not specify the BOOTREC operand, the value defaults to LABEL.

SECURE
NOSECURE
The SECURE operand specifies that the IPL is a secure IPL.

The NOSECURE keyword specifies that the IPL is not a secure IPL.

If you do not set the SECURE or NOSECURE parameter, the value defaults to NOSECURE.

The SECURE parameter is not valid unless the DEVICE parameter (virtual device number) is set.

If a user's OPTION directory statement includes the SECUREIPLREQUIRED option, the SECURE operand must be set.

ALTERNATE *fcp_vdev* PORTNAME *hhhhhhhhhhhhhhhh*
The ALTERNATE keyword specifies an alternate virtual device number and fibre channel port name to be IPLed if the initial device IPL fails. Up to three alternate paths can be defined. The combination of alternate device number and port name must be unique among all ALTERNATE definitions and different from the device number and port name combination that is specified by the DEVICE and PORTNAME keywords.

SCPDATA *'text'*
SCPDATA HEX *'hex_value'*
The SCPDATA keyword designates data to be passed to the program that is loaded for a list-directed IPL. If the program does not require any data, then the SCPDATA parameter is optional. The data must be enclosed in quotation marks.

The HEX keyword indicates that the data (*hex_value*) is UTF-8 encoded hexadecimal format (characters 0-F). The hexadecimal data must be an even number of digits. If continuation lines are used, the quotation-delimited hexadecimal string on each line of the statement must contain an even number of hexadecimal digits. If the HEX keyword is not specified, the data is assumed to be EBCDIC text (code page 924).

Up to 4096 (4K) characters of data (text or hexadecimal) can be entered. The actual number of input characters might be less than 4096 depending on the translation to UTF-8. Because two hex characters are required to represent each UTF-8 data byte, the maximum number of UTF-8 data bytes that can be defined by using the HEX option is 2048 (1/2 of the 4K character input limit).

There is no default value for SCPDATA.

Usage Notes

1. LOADDEV parameters are used for list-directed IPLs. Table 31 on page 543 shows which LOADDEV parameters have default values that must be set (changed from the default) before each type of list-directed IPL can be started. Other parameters might need to be set if the default values are not appropriate for your environment.

Table 31. LOADDEV parameters requirements by type of list-directed IPL without dump

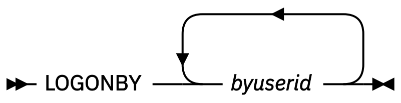
IPL type	DEVICE parameter	PORTNAME parameter	LUN parameter	SECURE parameter
IPL *fcp_vdev*	The DEVICE parameter is not set or matches *fcp_vdev* on the IPL command or IPL directory statement	Required	Required	The SECURE parameter is not allowed when the *fcp_vdev* operand is used
IPL LOADDEV from SCSI device	Required	Required	Required	Required if the SECUREIPLREQ option is specified in the OPTION directory statement
IPL LOADDEV from ECKD device	Required	Not applicable	Not applicable	

For more information, see IPL.

2. To prepare for a secure IPL, the program that is loaded must be signed. The appropriate certificate must be loaded in the HMC and assigned to any LPAR on which the IPL might occur.
3. The default value (0) for the PORTNAME and LUN parameters is not a valid device port name and not a valid device LUN. An IPL attempt that uses the default value for the PORTNAME or LUN parameter will fail.

4. The **SET LOADDEV** and **SET DUMPDEV** commands can change the LOADDEV and DUMPDEV parameters for the virtual machine temporarily. The temporary parameters can be used by an **IPL** command when the virtual machine is running. The temporary parameters do not persist after the virtual machine is logged off. See the following topics:
   - SET LOADDEV in *z/VM: CP Commands and Utilities Reference*
   - SET DUMPDEV in *z/VM: CP Commands and Utilities Reference*
5. You can change the LOADDEV directory statement and thus change parameters for a list-directed IPL by using the **DIRM LOADDEV** command and the Image_IPL_Characteristics_Define_DM API function. See the following topics:
   - **DIRM** LOADDEV in *z/VM: Directory Maintenance Facility Commands Reference*
   - Image_IPL_Characteristics_Define_DM in *z/VM: Systems Management Application Programming*
6. For information about the IPL directory statement and the **IPL** command, see the following topics:
   - "IPL Directory Statement" on page 519
   - IPL in *z/VM: CP Commands and Utilities Reference*
7. If the IPL of the initial device via the **IPL LOADDEV** command is not successful and alternate paths are defined, then the alternate device paths are automatically considered:
   - If a device is not available, then the next alternate path is considered.
   - Alternate device paths are considered in turn until IPL is successful or all device paths are considered and IPL is not successful.

LOGONBY Directory Statement



Purpose

The LOGONBY statement designates up to eight user IDs that can use their own passwords to log on to and use the virtual machine.

How to Specify

The LOGONBY statement is allowed in profile, user, and identity entries. The LOGONBY statement must precede any device statements in an entry. (For a list of device statements, see Table 24 on page 462.) An entry can contain several LOGONBY statements, and more than one LOGONBY user ID can be specified on each statement, but the maximum total of LOGONBY user IDs that can be specified in a user, identity, or profile entry is eight.

Operands

byuserid
is a 1- to 8-character user ID that is to log on to this virtual machine with the BY operand. Up to eight user IDs can be specified.

Usage Notes

1. When an External Security Manager (ESM) such as RACF is installed, the authorization provided by the LOGONBY statement may be overridden. Also, RACF support has been added to perform authorization checks for attempts to log on to shared user IDs. Refer to documentation provided by your ESM for further details.
2. When a user logs on with the BY operand, that user's ID is the by-user ID for the virtual machine.
3. The password of the by-user ID is used for LOGON authorization checking. For more information about the BY operand of the CP LOGON command, see LOGON in *z/VM: CP Commands and Utilities Reference*.
4. The current by-user ID for a virtual machine can be queried by using the CP QUERY BYUSER command, DIAGNOSE X'260' subcode 4, or DIAGNOSE X'26C' subcode 4.

Examples

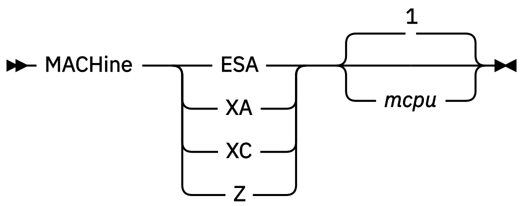
1. To authorize user ID LARRY to log on to a virtual machine with the BY operand, put this LOGONBY statement in the virtual machine definition:

   ```
   LOGONBY larry
   ```

2. To authorize user IDs OPER1, OPER2, OPER3, and JONES to log on to a virtual machine with the BY operand, put this LOGONBY statement in the virtual machine definition:

   ```
   LOGONBY oper1 oper2 oper3 jones
   ```

MACHINE Directory Statement



Notes:

[1] The default value is either 1 or the number of CPU statements for this user, whichever is greater.

Purpose

The MACHINE statement specifies the virtual machine architecture.

How to Specify

The MACHINE statement is allowed in profile, user, and identity entries. Only one MACHINE statement is allowed in each entry. If you specify the MACHINE statement, it must precede any device statements you specify in an entry. (For a list of device statements, see Table 24 on page 462.) A MACHINE statement in a user or identity entry overrides a machine statement in an included profile entry.

When processing the MACHINE statement, DIRECTXA checks for a maximum of two tokens: architecture type and maximum virtual processors, in that order. If you specify more than two tokens, DIRECTXA ignores the extra tokens and system processing continues as if you had not specified the extra tokens.

Operands

ESA

designates an ESA virtual machine, which simulates ESA/390 architecture.

A guest operating system might have the capability to issue an instruction to switch the virtual machine to z/Architecture mode.

See usage note "5" on page 547.

XA

designates an XA virtual machine, which is functionally equivalent to an ESA virtual machine.

XC

designates an XC virtual machine, which simulates ESA/XC architecture.

A guest operating system might have the capability to issue an instruction to switch the virtual machine to z/XC architecture mode.

Z

designates a Z virtual machine, which simulates a z/Architecture-only environment.

mcpu

defines the maximum number of virtual processors the user can define. The number must be between 1 and 64 (decimal). The default value is either 1 or the number of CPU statements for this user, whichever is greater.

Usage Notes

1. If you omit the MACHINE statement when you code a virtual machine definition:

- The user's virtual machine mode is defined by the mode specified by the GLOBALOPTS directory statement. If no mode is specified in the GLOBALOPTS directory statement, the default mode will be ESA.
- Additionally, if no CPU statements are in the virtual machine definition, the virtual machine has no virtual multiprocessor capabilities. However, if CPU statements are in the virtual machine definition, the number of these statements determines the allowable number of virtual processors.

2. For information on globally changing the default architecture, see "GLOBALOPTS Directory Statement" on page 509.
3. For information on defining multiple virtual processors for a virtual machine, see z/VM: Running Guest Operating Systems. For more information on defining multiple virtual processors for a virtual machine, see *z/VM: CP Commands and Utilities Reference*.
4. For compatibility, z/VM continues to accept the designation XA and continues to report XA as the virtual machine mode in response to the QUERY SET and INDICATE USER commands when the virtual machine has been defined using the XA designation. However, whether the virtual machine is defined as XA or ESA makes no difference when running on z/VM. A virtual machine defined as XA has the capabilities of an ESA virtual machine and is considered by CP to be an ESA virtual machine.
5. An ESA (or XA) virtual machine will be put into either full ESA/390 mode or ESA/390 compatibility mode, depending on the level of ESA/390 capability available in the machine where the virtual machine is logged on or in the virtual machine's relocation domain. ESA/390 compatibility mode allows a subset of ESA/390 functionality sufficient for CMS and GCS, but lacks advanced functions like dynamic address translation (DAT). For information on ESA/390 compatibility mode, see *z/Architecture Principles of Operation*.

Examples

1. To specify that a virtual machine can simulate ESA/390 architecture, use the following MACHINE statement in the virtual machine definition:

   ```
   Machine esa
   ```

2. To specify that a virtual machine can simulate ESA/390 architecture and define a maximum of 12 virtual processors, use the following MACHINE statement in the virtual machine definition:

   ```
   Machine esa 12
   ```

3. To specify that a virtual machine can simulate the ESA/XC architecture, use the following MACHINE statement in the virtual machine definition:

   ```
   Machine xc
   ```

MAXSTORAGE Directory Statement

```
▶▶─ MAXSTORAGE ── size ─▶◀
```

Purpose

The MAXSTORAGE statement specifies the maximum virtual storage size for a user.

How to Specify

The MAXSTORAGE statement is allowed in profile, user, identity, and subconfiguration entries. If specified, it must appear before any device statements in the entry. (For a list of device statements, see Table 24 on page 462.) A MAXSTORAGE statement is allowed in a profile if the USER or IDENTITY statement maximum storage size field of each of the including directory entries is either omitted or specified as an asterisk (*). A MAXSTORAGE statement in a user or identity entry overrides one in a profile entry.

If MAXSTORAGE or STORAGE statements are specified in the subconfiguration entry, then those statements override any in the identity entry. If either a STORAGE or MAXSTORAGE statement is specified, but not the other, within a subconfiguration entry, then the default is applied to the other in the subconfiguration entry. The combination of the specified value and the default then override whatever settings were specified in the identity entry.

Operands

size
is the maximum storage size of the virtual machine. Specify *size* as *nu*, where *n* is a 1- to 7-digit decimal number and *u* is the 1-character storage unit suffix (see Table 32 on page 548).

Table 32. Maximum Input Values for Storage Units

Storage unit suffix (*u*)	Maximum input value (*n*)
K - kilobytes	9999999
M - megabytes	9999999
G - gigabytes	9999999
T - terabytes	9999999
P - petabytes	16384
E - exabytes	16

Note:

1. A K specification is rounded up to a MB value.
2. The maximum input value of 9999999 for the K, M, G, or T suffix is not a size limit but the physical limit of the operand (7 digits plus suffix). If the maximum input value for one of these suffixes does not allow you to define the amount of storage you want, you need to use a larger storage unit.
3. The maximum size you can specify is 16E or 16384P, although the actual maximum size supported may be restricted by the model of the server where the directory is used.
4. An XC virtual machine can address up to 2047 MB of storage in its base address space.
5. Allowing many virtual machines to have large storage sizes might affect real storage availability. See usage note "1" on page 549.

Usage Notes

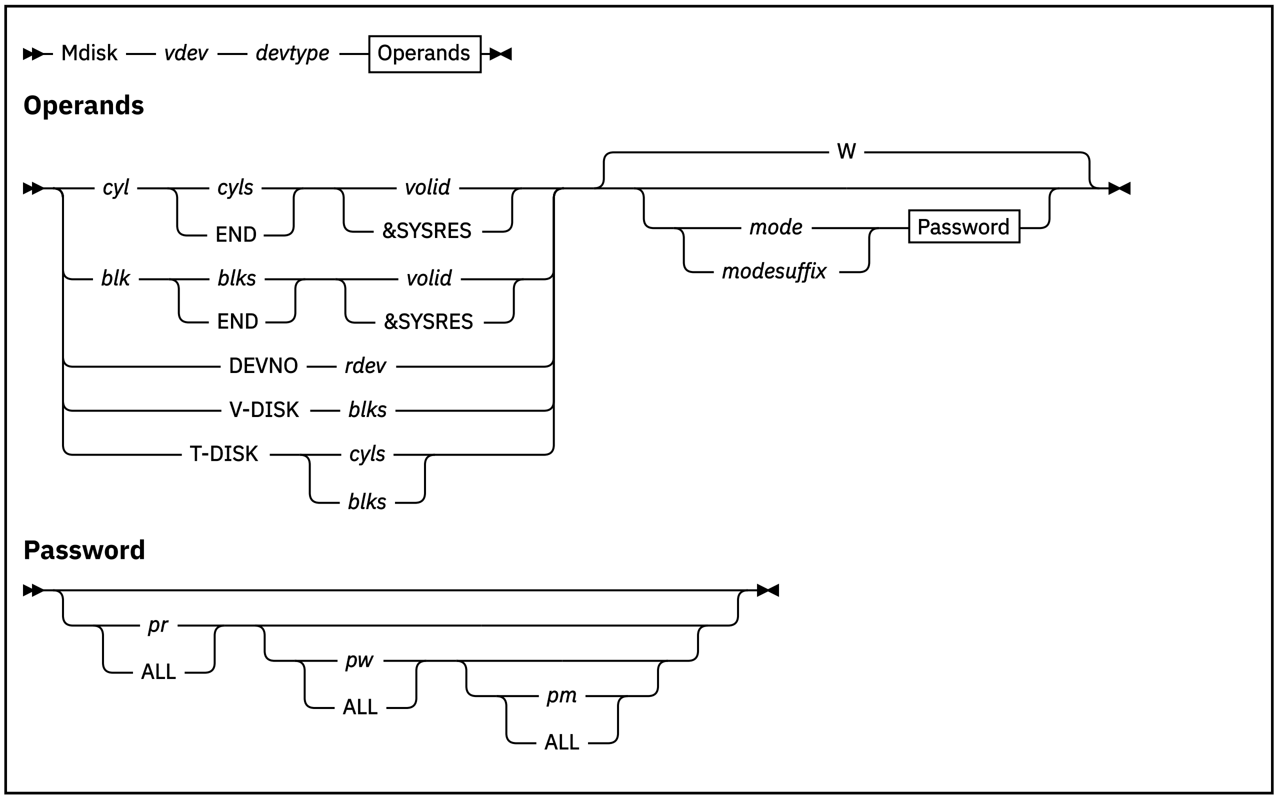
1. Allowing many virtual machines to have large storage sizes might affect real storage availability. For each virtual machine, CP creates dynamic address translation (DAT) tables to reference the virtual machine storage. DAT tables include page tables, segment tables, and higher level (region) tables.

   CP keeps the page tables in page management blocks (PGMBKs). Each 8 KB PGMBK references 1 MB of virtual machine storage. PGMBKs might be pageable; as such, their impact on real storage depends on how frequently the MBs of storage they reference are used.

   Segment tables and region tables are allocated from host real storage and are not pageable:

   - To reference the page tables for a primary address space or data space up to 2 GB, 1 - 4 contiguous frames are allocated for the segment table, one frame for each 512 MB of storage.
   - For a primary address space larger than 2 GB, multiple segment tables are created, plus one or more region tables to reference the segment tables. Each region table occupies 1 - 4 contiguous frames. If needed, multiple levels of region tables are created.

MDISK Directory Statement



Purpose

The MDISK statement defines minidisks (virtual disks) for virtual machines:

- Permanent minidisks
- Temporary minidisks
- Virtual disks in storage

The DASDOPT and MINIOPT statements are extensions to the MDISK statement. The MDISK statement defaults to DEVCTL if no DASDOPT statement follows. For more information, see "DASDOPT Directory Statement" on page 495 and "MINIOPT Directory Statement" on page 560. You can use one DASD to provide several minidisks or you can use one DASD to provide one minidisk (a *full-pack minidisk*).

Attention: CP and the directory program do not completely prevent you from defining minidisks that overlap. Overlap defeats the integrity of link access modes and RESERVE/RELEASE. If you define such overlap, you assume responsibility for data integrity.

How to Specify

MDISK statements are allowed within user, identity, and subconfiguration entries. If you specify the MDISK statement, it must follow any general statements that you specify in an entry. (For a list of general statements, see Table 24 on page 462.)

In an SSI cluster, the scope of an MDISK definition can be global (accessible from all members of the cluster) or local (accessible from only one member of the cluster):

- An MDISK definition for a permanent minidisk in a user or identity entry is global. The minidisk can be shared with users on any member of the cluster.
- An MDISK definition for a permanent minidisk in a subconfiguration entry is local. The minidisk can be shared only with users on the member to which the subconfiguration entry applies.

- An MDISK definition for a virtual disk in storage can be included only in a user or subconfiguration entry and is local. The minidisk can be shared only with users on the member where the minidisk is created.
- An MDISK definition for a temporary minidisk is local. The minidisk is created on the member where the user logs on.

MDISK passwords for MDISKs in the subconfiguration entry are encrypted with the user ID from the IDENTITY statement.

DIRECTXA does not check for extra tokens that might be specified at the end of the MDISK statement. If you specify additional tokens, DIRECTXA ignores the extra tokens and system processing continues without regard for the extra tokens.

Operands

vdev
is the virtual device number of the minidisk.

devtype
is the device type of the minidisk.

- For a permanent minidisk or temporary disk, this value is the device type of the real device on which the minidisk resides. Valid device types are:

  3380
  3390
  9336
  FB-512

  FB-512 is a generic specification for FBA DASD and can be used in lieu of a specific FBA device type (9336). A 3390 in 3380 track-compatibility mode must be coded as a 3380.
- For a virtual disk in storage, which is not mapped to a real DASD, this value is the device type of the device that is simulated in host storage. A virtual disk in storage simulates an FBA minidisk; therefore, the device type must be FB-512.

cyl
blk
is the starting cylinder (ECKD) or block number (FBA) on the real DASD you specify on the *volid* operand. Generally, ECKD user minidisks must begin with cylinder 1 or higher and FBA user minidisks must begin with block 32 or higher. If the SALIPL utility was used to previously install a copy of the Stand-Alone Program Loader (SAPL), then FBA DASD user minidisks must begin with block 208 or higher.

DEVNO
specifies a full-pack minidisk.

V-DISK
indicates that the minidisk is a virtual disk in storage. A virtual disk in storage is temporary but shareable. It is created when the first user links to it and destroyed when the last user detaches it or logs off.

The V-DISK operand is not allowed on an MDISK statement in an identity entry.

T-DISK
indicates that the minidisk is temporary. The minidisk is created from a preselected pool of temporary disk space when the virtual machine logs on. The minidisk is returned to the pool when the virtual machine logs off or the virtual device is detached. Temporary minidisks cannot have passwords and cannot be shared.

cyls
blks
is the number of cylinders (ECKD) or blocks (FBA) allocated to the minidisk. Table 33 on page 553 shows the number of cylinders or blocks available in each format for each device type on which you can put minidisks.

END
: specifies that the minidisk is defined with the remaining available cylinders or blocks. This is useful when a full-pack minidisk is defined. For more information, see usage note "4" on page 554.

rdev
: is the real device number.

volid
: is the volume serial number of the real DASD volume on which the minidisk resides.

&SYSRES
: indicates that the minidisk resides on whichever real DASD is the VM system residence volume. This volume is established when VM is IPLed but can change from one IPL to another.

Use the &SYSRES option of the DIRECTXA command to assign a value to represent the volume serial number of the system residence volume for minidisks that are defined by using the &SYSRES operand on the MDISK statement. The value that is used on the &SYSRES option of the DIRECTXA command is substituted for &SYSRES on the MDISK statement and used regardless of the actual system residence volume.

This value is used in responses from the VMUDQ LISTMDSK function. If the &SYSRES option is omitted on the DIRECTXA command, a value of +VMRES is used.

Note: If the SSI option is added to the DIRECTORY statement, the following are not allowed on the MDISK statement:

- &SYSRES
- +VMRES
- *volid*s that match the value that is supplied by the &SYSRES option of the DIRECTXA command.

mode
: is the access mode for the minidisk. The first letter in the access mode is the primary access mode (read-only, write, or multiple-write); the second letter (optional) is the alternate access (read-only or write).

The access modes are:

R
: Read-only access. Read access is established, unless another user holds a write or an exclusive mode (ER, EW) access to the disk.

RR
: Read-only access. Read access is established, unless another user holds an exclusive mode (ER, EW) access.

W
: Write access. Write access is established, unless another user holds access (any mode) to the disk. If you do not specify an access mode on the MDISK statement, the default is W.

WR
: Write access. Write access is established unless another user holds access (any mode) to the disk. If write access is denied, read access is established unless another user holds an exclusive (ER, EW) mode access for the disk.

M
: Multiple-write access. Write access is established unless another user holds a write, a stable (SR, SW, SM) or an exclusive (ER, EW) mode access to the disk.

MR
: Multiple-write access. Write access is established unless another user has a write, stable (SR, SW, SM) or exclusive mode (ER, EW) access to the disk. If another user has write or stable (SR, SW, SM) access, read-only access is established. If an exclusive mode access exists, access is denied.

MW
: Multiple-write access. Write access is established in all cases except when another user holds either a stable (SR, SW, SM) or an exclusive mode (ER, EW) access to the disk.

modesuffix

The optional mode suffix letters authorize virtual reserve/release and the use of the stable and exclusive (data integrity) access modes of the CP LINK command and DIAGNOSE code X'E4'. The specification here on the MDISK statement allows the user to enter a LINK command with the specified type of access mode, stable, or exclusive. For more information, see LINK in *z/VM: CP Commands and Utilities Reference*.

The suffix letters can be combined as follows (note that CP requires they be specified in this order):

1. V or null, plus
2. S, E, or null (S and E are mutually exclusive), plus
3. D or null

Therefore, the only valid combinations are: V, S, E, D, VS, VE, VD, VSD, VED, SD, or ED. This suffix is in turn concatenated with the mode, with no intervening blanks. For example, RV, RRS, WVE, and MVSD are all valid.

V

Tells CP to use its virtual reserve/release support in the I/O operations for the minidisk. For example, MWV means the minidisk functions with write linkage by using CP's virtual reserve/release.

S

Authorizes the virtual machine in whose virtual machine definition this statement appears the ability to use the LINK command stable access modes (SR, SW, SM) against the specified user's minidisk. For more information, see LINK in *z/VM: CP Commands and Utilities Reference*.

E

Authorizes the virtual machine in whose virtual machine definition this statement appears the ability to use the LINK command exclusive and stable access modes (ER, EW, SR, SW, SM) against the specified user's minidisk. For more information, see LINK in *z/VM: CP Commands and Utilities Reference*.

D

Tells CP that the device must not be defined when the virtual machine initially logs on or is auto logged. The definition is deferred until an explicit LINK command is issued for that device.

pr
ALL

is the password that allows sharing the minidisk (by using the CP LINK command) in read mode. The variable *pr* is a 1- to 8-character string. If you specify ALL, a user can link in read mode to this minidisk without using a password.

pw
ALL

is the password that allows sharing the minidisk (by using the CP LINK command) in write mode. If you specify ALL, a user can link in write mode to this minidisk without using a password.

pm
ALL

is the password that allows sharing the minidisk (by using the CP LINK command) in multiple-write mode. The variable *pm* is a 1- to 8-character string. If you specify ALL, a user can link in multiple-write mode to this minidisk without using a password.

Table 33. Maximum Minidisk Sizes

Device Type and Model	Maximum Size (Cylinders/Blocks)	CMS/VSAM Size Limits
3390-1 (See note 1)	1113 cylinders	1113 cylinders
3390-2 (See note 1)	2226 cylinders	2226 cylinders
3390-3 (See note 1)	3339 cylinders	3339 cylinders

Table 33. Maximum Minidisk Sizes (continued)

Device Type and Model	Maximum Size (Cylinders/Blocks)	CMS/VSAM Size Limits
3390-9 (See note 2)	65520 cylinders	10017 cylinders (See notes 3,4,6)
3390-A	1182006 cylinders	65520 cylinders
9336-020	2147483640 blocks	381 GB (See notes 5,6)
FB-512 (virtual disk in storage)	4194296 blocks	4194296 blocks

Notes:

1. These capacities apply to 3390 mode and 3380 track compatibility mode.
2. Minidisks that are used with VSAM are limited to 64 KB (65,536) tracks (4369 3390 cylinders). This is a limitation of the VSE/VSAM licensed program. This limit applies whether the minidisk is used by a VSE guest or by CMS/VSAM.
3. Value can be up to the maximum number of cylinders on the DASD (GCS is limited to 32,767 cylinders; CMS limits depend on the 3390-9 capacity model: 32,767 for a Mod 27 and 65,520 for a Mod 54.)
4. z/VM supports a SCSI disk, which is emulated as a 9336-020, up to a capacity of 268,435,455 4096-byte pages (1 terabyte minus one page). However, directory, paging, and spooling allocations must reside within the first 16,777,215 pages (64 GB minus one page) of a CP-formatted disk.
5. The CMS file system requires that file status and control information resides in the virtual address lower storage region, at an address that is less than 16 MB. Hence, the size of CMS minidisks has a practical limitation. As a minidisk increases in size, or more files reside on the disk, the amount of virtual storage that is required for CMS file system status and control increases. The current ECKD DASD limitation for CMS is 65520 cylinders for a 3390 disk on an IBM TotalStorage DASD subsystem, or about 45 GB of data. The maximum supported size for FBA SCSI disks that are used by CMS or GCS is 381 GB. A practical limit for CMS minidisks is about 22 GB. If larger disks are defined, they must contain few files. Larger disks with more files might cause a problem for the CMS file system to obtain enough virtual storage in the lower storage region to format or access those disks. For more information, see CMS ACCESS in *z/VM: CMS Commands and Utilities Reference*.

Usage Notes

1. You must format a temporary minidisk or virtual disk in storage before you use it. If you want to format the disk for CP use, use the Device Support Facility (ICKDSF) program; it is the recommended method of CP volume maintenance. For more information about ICKDSF procedures, see the *Device Support Facilities User's Guide and Reference*, GC35-0033. If you want to format the disk for CMS use, use the CMS FORMAT command. For more information, see CMS FORMATdms in *z/VM: CMS Commands and Utilities Reference*. To format minidisk space for other operating systems, use the commands of the operating system that controls the minidisk.
2. For security, temporary disk space that contains sensitive data can be reformatted before the disk is detached from the virtual machine. If the system is configured to clear all temporary disk space, this clearing is done by the system. You can determine whether temporary disk clearing is enabled by entering the QUERY TDISKCLR command. Temporary disk clearing removes any risk of security exposure.
3. If for some reason two or more volumes have the same label, CP defines the minidisk on the volume that is attached to the system. (You cannot attach two volumes with the same label to the system at the same time.)
4. To define a full-pack minidisk, you must specify that the minidisk is to contain all of a DASD's primary cylinders or blocks. Specify '0' to 'END' on the MDISK statement.

   Note:

   - A minidisk that begins with cylinder 1 or block 1 is not a full-pack minidisk.

- Define by using the keyword 'END' so your MDISK statement is not dependent on the size and type of DASD.
- The 'DEVNO' keyword can also be used to define a full-pack minidisk.

You can specify any of the DASD's alternate cylinders or blocks, but those cylinders or blocks, and all others, are accessible to anyone with a full-pack minidisk.

5. Full-pack minidisks that overlap other minidisks are ignored during link access mode checking. However, you must not define any other type of overlap. CP checks overlap to ensure the integrity of link access modes. This is not a feature to exploit.

   If you do overlap other minidisks, you might experience the following consequences:

   - You cannot link to those minidisks and receive messages HCP104E, HCP105E, and HCP106E.
   - You can link only in read-only mode to those minidisks and receive messages HCP101E, HCP102E, and HCP103E.
6. CP volumes are supported in either 3380 track compatibility mode or 3390 mode of operation on a volume basis except for the 3390 Model 9, which does not support 3380 track compatibility mode.
7. Read Special home address and Write Special home address CCWs are only supported for dedicated and full-pack minidisks for guests with the OPTION statement value MAINTCCW or DEVMAINT.
8. The use of the stable (SR, SW, and SM) and exclusive (ER and EW) link modes can be authorized in two different ways:

   - Globally, by using the OPTION statement values LNKSTABL and LNKEXCLU
   - Specifically, by using the mode suffix letter on the MDISK or LINK statements
9. The DEVNO operand allows the user to define a full-pack minidisk by specifying the real device number of the minidisk. This operand must be used when you define full-pack minidisks where the guest controls the duplexing of the device.
10. The DEVNO operand must be used when you define multiple minidisks with identical volume serial numbers.
11. Refer to the appropriate storage control manuals for more dual copy information.
12. For information on the use of the MDISK statement for the CMS shared file system, refer to *z/VM: CMS File Pool Planning, Administration, and Operation* .
13. If an external security manager (ESM) is installed, you might not be authorized to use the MDISK statement for all minidisks and all access modes. The ESM might downgrade certain requests for write access to read access. For additional information, contact your security administrator.
14. Virtual disks in storage and FBA temporary minidisks are created in 8-block pages. Therefore, the size of the virtual disk in storage or FBA temporary minidisk that is created might be rounded up to the nearest page.
15. If you define a minidisk as virtual device number 192, the following special rules apply when you IPL CMS:

    - If 192 is an unformatted temporary minidisk or virtual disk in storage, CMS formats it and accessed it as file mode D.
    - If 192 is a CP-formatted temporary minidisk or virtual disk in storage, CMS reformats it for CMS use and accesses it as file mode D.
    - If 192 is a CMS-formatted temporary minidisk, virtual disk in storage, or permanent minidisk that is accessed as a file mode other than D, CMS reaccesses it as file mode D.
    - If 192 is an unformatted or CP-formatted permanent minidisk, CMS does not automatically format, reformat, access, or reaccess it.

    When CMS accesses a 192 minidisk as file mode D, any minidisk or SFS directory that is already accessed as D is released.
16. If you change the definition of a permanent minidisk while users are linked to it, existing links are unchanged. However, any new links are made by using the new definition of the minidisk. Unpredictable results might occur.

17. If you change the definition of a virtual disk in storage while users are linked to it, existing links are unchanged. Any new links are made to the existing virtual disk in storage. The existing virtual disk in storage is used until the last user detaches it or logs off, at which time the virtual disk in storage is destroyed. After that, any new links use the new definition of the virtual disk in storage.
18. Some programs might not support the use of &SYSRES, and require the use of a synonym. The synonym is the value that is specified on the DIRECTXA command with the &SYSRES operand. If the &SYSRES option is omitted on the DIRECTXA command, a synonym of +VMRES is used. In other words, either &SYSRES or its synonym can be used to define a minidisk on the system residence volume.
19. A minidisk, including a temporary disk, is not eligible for minidisk cache if it is defined with greater than 32767 cylinders. This restriction does not apply to FBA devices. The restriction applies only to ECKD devices.
20. For more information about the V mode suffix restrictions when linking, see LINK in *z/VM: CP Commands and Utilities Reference*.
21. In an SSI cluster, if a virtual disk in storage is defined in a user entry, the minidisk can be created on any member of the cluster. The minidisk is created when the owner logs on or when the first user links to it. After creation, the minidisk can be shared only by other users on that member until the last user detaches it or logs off.
22. If a temporary minidisk is defined in an identity entry, a separate temporary minidisk is created for each logon instance of the virtual machine.
23. Minidisk cache is not supported for a minidisk that is defined on an extended address volume (EAV) DASD such as the 3390-0E. The restriction is because of the large number of cylinders that are supported by that device.
24. The *cyl* operand of the MDISK statement and the *cyls* option of the TDISK operand allow values up to 1,182,006 cylinders.

Examples

1. Define a minidisk that has the following features:
   - Has the virtual device number 191
   - Takes up five cylinders, from cylinder 100, of the 3390 DASD with volume serial MDDASD
   - Is accessible only in read-only mode
   - Cannot be linked to in any mode (no passwords are specified)

   Use the following MDISK statement in the virtual machine definition:

   ```
   Mdisk 191 3390 100 5 mddasd rr
   ```
2. Define a minidisk that has the following features:
   - Has the virtual device number 291
   - Takes up 10 cylinders, from cylinder 105, of the 3390 DASD with volume serial MDDASD
   - Is accessible only in read or write mode
   - Can be linked to by anyone in read mode (but no other mode)

   Use the following MDISK statement in the virtual machine definition:

   ```
   Mdisk 291 3390 105 10 mddasd mr all
   ```
3. Define a minidisk that has the following features:
   - Has the virtual device number 198
   - Takes up 6000 blocks, from block 12,000, of a supported FBA DASD with volume serial FBDASD
   - Is accessible only in multiple-write (MW) modes

- Can be shared by using CP's virtual reserve/release
- Has a read password of 12WE45

Use the following MDISK statement in the virtual machine definition:

```
Mdisk 198 9336 12000 6000 fbdasd mwv 12we45
```

4. Define a minidisk that has the following features:
   - Has the virtual device number 198
   - Takes up 6000 blocks, from block 12,000, of the 9336 DASD with volume serial FBDASD
   - Has the default access mode of Write when the virtual machine is logged on, but specifically authorizes the user to access it with a stable access mode by using the LINK command later.
   - Can be shared by using CP's virtual reserve/release
   - Has a read password of 12WE45

   Use the following MDISK statement in the virtual machine definition:

   ```
   Mdisk 198 fb-512 12000 6000 fbdasd mvs 12we45
   ```
5. To define five cylinders of temporary 3390 DASD space at virtual device number 391, use the following MDISK statement in the virtual machine definition:

   ```
   Mdisk 391 3390 t-disk 5
   ```
6. To define 4000 blocks of temporary FBA DASD space at virtual device 392, use the following MDISK statement in the virtual machine definition:

   ```
   Mdisk 392 fb-512 t-disk 4000
   ```
7. To define a 3380 DASD full-pack minidisk at virtual device number 199, use one of the following MDISK statements in the virtual machine definition:

   ```
   Mdisk 199 3380 000 885 mddasd mr
   Mdisk 199 3380 000 end mddasd mr
   ```
8. Define a pair of full-pack minidisks that are candidates for duplexing and have the following features:
   - Have virtual device numbers 198 and 199
   - Have real device numbers 200 and 201
   - Are accessible in read or write mode
   - Can be linked to by anyone in read mode

   Use the following MDISK statements in the virtual machine definition:

   ```
   mdisk 198 3390 devno 200 mr all
   mdisk 199 3390 devno 201 mr all
   ```
9. Define a minidisk that has the following features:
   - Has virtual device number 401
   - Consists of 8000 blocks
   - Is accessible in read or write mode
   - Can be shared by using CP's virtual reserve/release

   Use the following MDISK statement in the virtual machine definition:

   ```
   mdisk 401 fb-512 v-disk 8000 mwv
   ```
10. To define a minidisk that uses a synonym of VM-RES instead of &SYSRES, use the following MDISK statement:

```
   MDISK 123 3390 0 END VM-RES RR
```

And use the following command to place the directory online:

```
DIRECTXA USER DIRECT (&SYSRES VM-RES
```

Minidisk Restrictions

The following restrictions exist for minidisks:

1. In the following cases, z/VM modifies the cylinder data in user storage at the completion of the channel program:

   Read Home Address (with the skip bit off)
   Read Record Zero (with the skip bit off)
   Sense (with the skip bit off)
   Read Track (with the skip bit off)
   Read Device Characteristics.

   This is necessary because the addresses must be converted for minidisks. Therefore, the data buffer area cannot be dynamically modified during the I/O operation in these cases.

   Note: For the Read Record Zero case, the restriction does not apply to devices that do not provide alternate track recovery.
2. On a minidisk, if a CCW string uses multitrack-search on I/O operations, subsequent operations to that disk must have preceding seeks or continue to use multitrack operations. Dedicated disks are not restricted.
3. If the user's channel program for a count-key-data minidisk does not complete a seek operation, CP inserts a positioning seek operation into the user's channel program to prevent accidental accessing. Thus, certain channel programs might generate a condition code (CC) of 0 on an SIO instead of the expected CC of 1, which is reflected to a virtual machine. The final status is reflected to the virtual machine as an interruption. The final states for some channel programs that are initiated through an SIOF or SSCH might not indicate deferred CC 1.
4. A DASD channel program might give results on dedicated DASD that differ from results on minidisks that have nonzero relocation factors if the following conditions are true:

   - The channel program includes multiple-track operations.
   - The channel program depends on a Search ID High or Search ID Equal or High to terminate the program.

   The record 0 count fields on these devices must contain the real cylinder number of the track on which they reside. Therefore, a Search ID High, for example, based on a low virtual cylinder number, might terminate prematurely if a real record 0 is encountered.

   Minidisks with nonzero relocation factors are not usable under OS and OS/VS systems. The restriction is because the locate catalog management function employs a Search ID Equal or High CCW to find the end of the VTOC.
5. The IBCDASDI program cannot assign alternate tracks.

   Notes:

   a. Device-Support Facilities (ICKDSF) might assign alternate tracks
   b. Alternate track assignment is permitted for full-pack minidisks only.
6. If the DASD channel programs include a Write Record Zero CCW or a Write Home Address CCW, the results depend on whether the device is dedicated or not dedicated.

   - For a dedicated DASD, a Write Record Zero or a Write Home Address CCW is allowed. However, the user must be aware that the track descriptor record might not be valid from one DASD to another.
   - For a DASD that is not dedicated, a Write Record Zero or a Write Home Address CCW is accepted only if the device is a full-pack minidisk. CP rejects the command if the device that issues a Write

Record Zero or a Write Home Address CCW on a nondedicated DASD is not defined as a full-pack minidisk.

7. During DASD I/O, the real Search ID uses the relocated search argument instead of the argument that was read dynamically if the following conditions are true:

   - The record field of a Search ID Argument is zero when a virtual SIO, SIOF, or SSCH is issued
   - The Search ID Argument is dynamically read by the channel program before the Search ID CCW is executed.

   This problem can be avoided if you do not set the record field of a Search ID Argument to binary zero in the following cases:

   - The search argument is read dynamically.
   - The search ID on record 0 is not intended.

8. The following CCWs are restricted to virtual machines with MAINTCCW authority. You specify this authority by coding the MAINTCCW operand on the OPTION statement in the virtual machine definition.

   - Diagnostic Write Home Address
   - Diagnostic Read Home Address
   - Write Record Zero
   - Diagnostic Load
   - Diagnostic Sense/Reset Allegiance: CP treats the command as a reset allegiance CCW and accepts it if it is the first CCW in the channel program.
   - Diagnostic Write
   - Diagnostic Sense/Read
   - Diagnostic Control

9. Diagnostic Read Home Address and Diagnostic Write Home Address commands are supported only for dedicated and full-pack 3380 and 3390 minidisks.

   All Diagnostic CCWs are restricted to users with the OPTION statement value MAINTCCW or DEVMAINT.

10. Refer to *OS/VS Device Support Facilities* for procedures to format all supported DASDs for use in an OS/VS operating system that runs in a virtual machine.

11. The 3390s that are used for CP volumes are supported in either 3380 track compatibility mode or 3390 mode of operation on a volume basis. The exception is the 3390 Model 9, which does not support 3380 track compatibility mode. The ability to change operating modes is restricted to guests with the OPTION statement value MAINTCCW or DEVMAINT. The use of ICKDSF is the recommended method to change modes.

12. A user must not have an MDISK statement with the DEVNO operand and an MDISK statement with the volume serial number operand for the same volume.

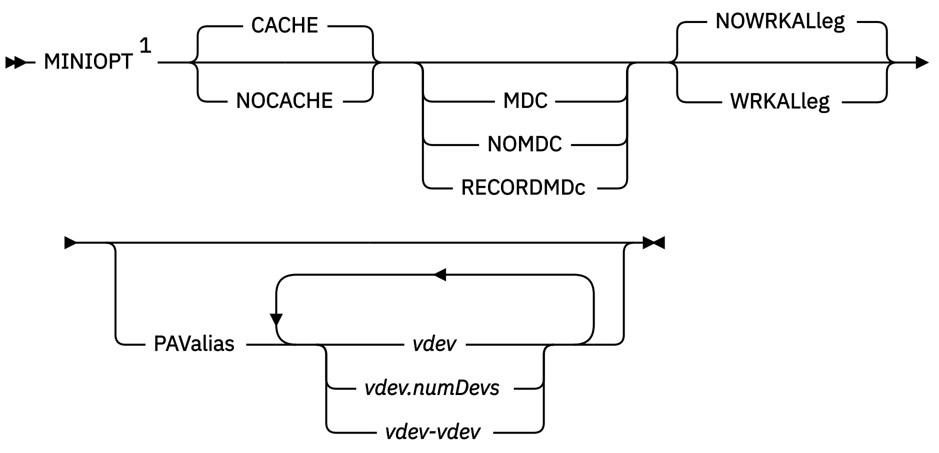
13. Access to cache control units is controlled by the settings of level of control in the DASDOPT directory statement. For more information, see "DASDOPT Directory Statement" on page 495.

14. Minidisks that are defined on an FBA volume and are not page-aligned are not eligible for minidisk cache. Hence, those minidisks do not benefit from a performance improvement. Aligning minidisks on page boundaries is highly recommended. Minidisks and full-pack minidisk volumes that are not page-aligned might result in residual blocks that are not formatted and not used. Additional restrictions apply for SFS Filepool minidisks that are not page-aligned. For more information, see *z/VM: CMS File Pool Planning, Administration, and Operation* . If the starting block number is a multiple of eight and the number of blocks is a multiple of eight, then the minidisk is defined to be page-aligned.

    Note: Under the conditions outlined in this usage note, residual blocks on FBA devices that are ICKDSF formatted can still be defined and used as MDISK space.

15. Minidisks that are defined on an FBA volume and are not page-aligned cannot be mapped to an address space by using minidisk mapping.

MINIOPT Directory Statement



Notes:

[1] Options can be in any order.

Purpose

The MINIOPT statement is an extension to the MDISK statement and must immediately follow an MDISK statement that defines a non-full-pack minidisk. If you want to use a full-pack minidisk, you can use a DASDOPT statement with the MDISK statement.

How to Specify

MINIOPT statements are not allowed within a profile entry.

Operands

CACHE
: means that the minidisk has access to the control unit cache. This is the default if neither CACHE or NOCACHE are specified.

NOCACHE
: means that CP forces I/O for the minidisk to bypass the control unit cache.

MDC
: specifies that the minidisk will use the full track minidisk cache as long as caching is set to DFLTON or DFLTOFF for the real device. If neither MDC, NOMDC, nor RECORDMDc is specified, then the minidisk will use the full track minidisk cache as long as caching is set to DFLTON for the real device.

NOMDC
: specifies that the minidisk will not use the full track minidisk cache. If neither MDC, NOMDC, nor RECORDMDc is specified, then the minidisk will use the full track minidisk cache as long as caching is set to DFLTON for the real device.

RECORDMDc
: specifies that the minidisk will use record level minidisk caching rather than normal full track minidisk caching as long as caching is set to DFLTON or DFLTOFF for the real device. If neither MDC, NOMDC, nor RECORDMDc is specified, then the minidisk will use full track minidisk cache as long as caching is set to DFLTON for the real device.

Attention:
Use of this option is a last resort for very special and unusual circumstances. It should only be used as the result of consultation with IBM support personnel who have concluded that no other problem exists. Its use should occur only after trials using the SET MDCACHE MDISK ON command's RECORDMDc option have proven that this truly solves the performance problem. Using this option for any other reason may mask a performance problem which when corrected allows normal full track minidisk cache to perform better than with the RECORDMDc option.

WRKALleg
causes working allegiance to be simulated on the minidisk.

NOWRKALleg
causes no simulated working allegiance for the minidisk. This is the default.

PAValias
defines one or more alias Parallel Access Volumes for the non-full-pack minidisk base Parallel Access Volume specified in the preceding MDISK statement.

vdev
vdev.numDevs
vdev-vdev
is the virtual device address of the alias Parallel Access Volume that you are defining. You can specify a single virtual device (*vdev*), a virtual device combined with a decimal range count (*vdev.numDevs*), a range of virtual devices (*vdev-vdev*), or any combination. The device numbers entered must be hexadecimal numbers between X'0000' and X'FFFF'.

Usage Notes

1. IBM recommends that the defaults of both control unit caching and minidisk caching be used. The default double caching of data should not degrade the performance of normal minidisk operation. MINIOPT statements should be used only in cases of shared DASD and specific data/transaction-related performance problems.
2. WRKALLEG must be used when running two or more z/OS guests as part of a Sysplex configuration using the cross system coupling facility (XCF) component of z/OS. This option must be used for any minidisk containing the XCF couple dataset to maintain cross-system lock integrity (and thereby, data integrity) within the sysplex.
3. WRKALLEG is valid only when the preceding MDISK statement gives the user write access to the minidisk. If the MDISK statement specifies read-only access, CP rejects the WRKALLEG statement and issues an error message. Furthermore, working allegiance is simulated only when a guest with write access initiates I/O.
4. Implementation of caching by the system depends on the following:
   - For the control unit if:
     - It is a cached control unit
     - The subsystem is enabled for cache operation
     - The caching function is turned on for the device
   - For minidisk cache if:
     - Minidisk is on a device supported by minidisk cache.
     - Minidisk caching is enabled for the system.
     - Minidisk caching is set to DFLTON for the real device and either MINIOPT MDC is specified or no minidisk cache option is specified on the MINIOPT statement for the minidisk; or minidisk caching is set to DFLTOFF for the real device and MINIOPT MDC is specified for the minidisk:

Cache setting for real device	MINIOPT Default	MINIOPT MDC	MINIOPT NOMDC
DFLTON	ON	ON	OFF
DFLTOFF	OFF	ON	OFF
OFF	OFF	OFF	OFF

For example:

- If DFLTON and MINIOPT is NOMDC, then caching is off.
- If DFLTOFF and MINIOPT is MDC, then caching is on.

– For FBA devices, the minidisk is page-aligned

5. Optimization of cache use may be made after an analysis of your production use has been made. This is accomplished by adding MINIOPT statements that cause the I/O subsystem to bypass the use of either or both of the caching techniques.
6. MINIOPT is not valid for full-pack minidisks. For full-pack minidisks, use the DDASDOPT directory statement. For more information, see "DASDOPT Directory Statement" on page 495.
7. MINIOPT is ignored if it follows an MDISK statement that defines a virtual disk in storage.
8. The RECORDMDc option should only be used when directed to do so by IBM support personnel. RECORDMDc is for use only when DASD cache and real storage are being overwhelmed by excessive unreferenced records being read by the full track cache support.
9. A minidisk, including a temporary disk, is not eligible for minidisk cache if it has been defined with greater than 32767 cylinders. This does not apply to FBA devices. It applies only to ECKD devices.
10. When using the PAValias parameter, the number of virtual alias Parallel Access Volumes that can be associated with a particular virtual base Parallel Access Volume cannot exceed the number of real alias Parallel Access Volumes that are associated with the real base Parallel Access Volume on which the virtual base is defined.
11. The MINIOPT statement can be continued onto additional lines. For more information, see "Continued Directory Statements" on page 467.
12. The CP LINK command contains information about the WRKALLEG restrictions when linking. For more information, see LINK in *z/VM: CP Commands and Utilities Reference*.

Examples

1. If the minidisk specified by the MDISK statement is to have access to the control unit cache, use the following MINIOPT statement:

   ```
     MiniOpt Cache
   ```

2. If the minidisk specified by the MDISK statement will *not* use the minidisk cache, use the following MINIOPT statement:

   ```
     MiniOpt NoMdc
   ```

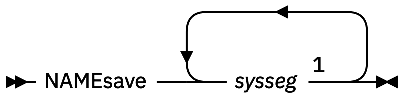
3. If the minidisk specified by the MDISK statement will have simulated working allegiance, use the following MINIOPT statement:

   ```
     MINIOPT WRKALleg
   ```

4. If the minidisk specified by the MDISK statement is a base Parallel Access Volume and you want to define four alias Parallel Access Volumes for it, use the following MINIOPT statement:

   ```
   MINIOPT PAVALIAS 681-684
   ```

NAMESAVE Directory Statement



Notes:

[1] Specify *sysseg* up to 8 times.

Purpose

The NAMESAVE statement authorizes a virtual machine to access a restricted named saved system or saved segment. The NAMESAVE statement also authorizes a virtual machine to access and load a private copy of a nonrestricted saved segment, using the load nonshared function of DIAGNOSE code X'64'. If the virtual machine has the appropriate NAMESAVE entry, it can access a private copy of a saved segment (whether or not the saved segment is restricted) for debugging, using the load nonshared function of DIAGNOSE code X'64'. (For more information, see Usage Note "3" on page 563.)

How to Specify

The NAMESAVE statement is allowed in profile, user, and identity entries. If you specify the NAMESAVE statement, it must precede any device statements you specify in a directory entry. (For a list of device statements, see Table 24 on page 462.)

Multiple NAMESAVE statements are allowed within a profile, user, or identity entry. NAMESAVE statements within a profile entry are added to those in the including user or identity entry with no duplicate checking performed.

Operands

sysseg
is the 1- to 8-character alphanumeric name of the named saved system or saved segment that you are authorizing a virtual machine to access. You can specify a maximum of eight names.

Note: *sysseg* cannot be a member name. Members are authorized by the segment space they are in.

Usage Notes

1. Each NAMESAVE statement can specify up to eight named saved systems and eight saved segments.
2. For information on creating named saved systems, see *z/VM: Virtual Machine Operation*.
3. If you have the appropriate NAMESAVE entry, you can access a private copy of a saved segment (whether or not the saved segment is restricted) for debugging purposes, using the load nonshared function of DIAGNOSE code X'64'. When you do this, you receive exclusive read/write access to all the page ranges in the saved segment, regardless of the type of access specified on the CP DEFSEG command when the saved segment was defined.
4. To access a private copy of a named saved system, you must IPL by device number. You cannot specify that a private copy of a named saved system should be loaded.
5. When packing DCSSs you must have the segment space name in a NAMESAVE statement.

Examples

1. To indicate that a virtual machine can access the restricted named saved systems MVSXA1 and MVSXA2, use the following NAMESAVE statement in the virtual machine definition:

```
NameSave mvsxa1 mvsxa2
```

2. To indicate that a virtual machine can access both of the following:
   a. The restricted named saved systems MVSXA1 and MVSXA2
   b. The restricted saved segment XASEG

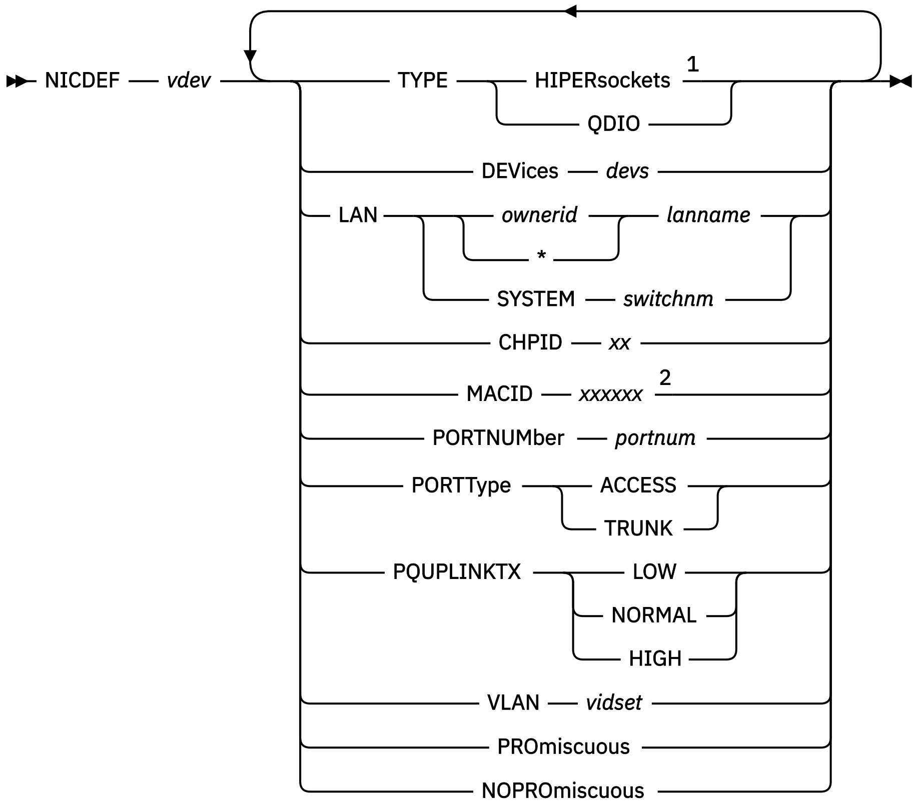
   use the following NAMESAVE statement in the virtual machine definition:

   ```
   NameSave mvsxa1 mvsxa2 xaseg
   ```

3. To indicate that a virtual machine can access a private copy of the nonrestricted saved segment XASEG1, use the following NAMESAVE statement in the virtual machine definition:

   ```
   NameSave xaseg1
   ```

NICDEF Directory Statement



Notes:

[1] TYPE must be specified exactly one time for each virtual NIC.

[2] MACID can be specified only for a type QDIO or HiperSockets adapter.

Purpose

The NICDEF statement defines virtual network adapters that are fully simulated by CP. Use NICDEF to create a specific type of adapter in the virtual machine, and (optionally) connect it to an appropriate guest LAN or virtual switch.

All device characteristics can be specified on one or more NICDEF statements. Characteristics not specified will be inherited from the guest LAN or virtual switch.

How to Specify

The NICDEF statement is allowed in profile, user, identity, and subconfiguration entries. If you specify the NICDEF statement, it must follow any general statements you specify in a directory entry. (For a list of device statements, see Table 24 on page 462.)

The SPECIAL statement can also be used to define virtual network adapters in a virtual machine. However, there are additional configuration options available with the NICDEF statement. NICDEF statements are not compatible with the SPECIAL statements for the same *vdev*. You cannot define some attributes on the SPECIAL statement and some attributes on a NICDEF statement.

At logon time, any NICDEF statements in a subconfiguration entry are processed first, followed by NICDEF statements in a user or identity entry, followed by NICDEF statements in a profile entry. Consecutive NICDEF statements with the same virtual device addresses are allowed within an entry and are treated as modifiers of previous NICDEF statements with the same virtual device address.

Operands

vdev
is the base (or first) device address in a series of virtual I/O devices that belong to the same unit.

TYPE
specifies the type of NIC adapter to be created, specifically the hardware and protocol that the adapter will emulate. TYPE is a required keyword and it must be the first keyword specified. If a LAN is identified in this statement, an attempt is made to couple the adapter to the specified *ownerid lanname*.

HIPERsockets
defines this adapter as a simulated HiperSockets NIC. This adapter will function like the HiperSockets internal adapter (device model 1732-05). A HiperSockets NIC can function without a guest LAN connection, or it can be coupled to a HiperSockets guest LAN.

An error results if you attempt to connect a simulated HiperSockets adapter to a virtual switch.

QDIO
defines this adapter as a simulated QDIO NIC. This adapter will function like the OSA-Express (QDIO) adapter (device model 1732-01). A QDIO NIC is functional only when it is coupled to a QDIO guest LAN or a type QDIO virtual switch.

DEVices *devs*
is the number (decimal) of virtual I/O devices to be created for a simulated network interface card (NIC). This number is evaluated during LOGON processing.

Table 34. Number (Decimal) of Virtual I/O Devices

Adapter TYPE	Minimum	Maximum	Default
HiperSockets	3	3072	3
QDIO	3	240	3

LAN [*ownerid*|* *lanname*] [SYSTEM *switchnm*]
identifies a virtual LAN segment for an immediate connection to the network interface card (NIC). When *ownerid* is specified as an asterisk (*), it is resolved as the user ID of the current virtual machine. When the LAN operand is omitted, the adapter is left in the default (uncoupled) state. When LAN *ownerid lanname* is identified in this statement or another with the same *vdev*, the adapter is connected to the designated virtual LAN segment automatically.

Ownerid may be specified as SYSTEM, indicating the virtual LAN segment may be a virtual switch or a system-owned LAN.

When z/VM is enabled for Directory Network Authorization, a system administrator can configure and consolidate a virtual NIC device and its network properties in a secure, centralized location in z/VM's User Directory. Therefore when a network configuration is added to the NICDEF statement, the MODIFY VSWITCH statement (SYSTEM CONFIG) and CP SET VSWITCH command can be eliminated. In this case, DNA provides the grant authorization methods previously provided by these commands. The network administrator can manage each user connection entirely within the user directory.

LAN is required when a VSwitch-specific operand (PORTNUMBER, PORTTYPE, VLAN, PQUPLINKTX, PROMISCUOUS or NOPROMISCUOUS) is specified. Furthermore, the use of a VSwitch-specific operand restricts the virtual NIC to the designated network. When the virtual NIC is configured for a specific network, the CP COUPLE command will not allow a connection to any other network.

If an External Security Manager (ESM) is in control of the virtual switch, it may override the CP authorization.

Note: Ensure that the defined NIC adapter type is compatible with the intended guest LAN or virtual switch.

CHPID *xx*
is a 2-digit hexadecimal number that represents the Channel Path ID (CHPID) number to be allocated in the virtual machine I/O configuration for this adapter. If **CHPID** is omitted, an available CHPID is automatically assigned to this adapter. This option is required when a HiperSockets adapter is being created for a z/OS guest because z/OS configurations require a predictable CHPID number. During LOGON, CP attempts to use the specified CHPID number. If the specified CHPID number is already in use, this adapter is not defined. To correct this situation, you must eliminate the conflicting device or select a different CHPID.

MACID *xxxxxx*
is a unique identifier (up to 6 hexadecimal digits in the range 000001 - FFFFFF) that is to be used as part of the adapter MAC address for a QDIO or HiperSockets type NIC adapter.

During LOGON, the specified MACID (3 bytes) is appended to the system MACPREFIX or USERPREFIX (3 bytes) to form a unique MAC address for this adapter. If MACID is omitted from this definition, CP generates a unique identifier for this adapter using the system MACPREFIX. If the specified MACID is already in use, this adapter is not defined. To correct this situation, you must eliminate the conflicting device or select a different MACID.

If the MACPREFIX and USERPREFIX are set to the identical value, the specified MACID must fall within the USER subset of the MACIDRANGE SYSTEM range defined on the VMLAN configuration statement.

PORTNUMber *portnum*
is the VSwitch-specific port number (a decimal number in the range 1 - 2048) to be used when this virtual NIC is connected to the NICDEF LAN.

Note: User-defined port numbers are not recommended for a user based relocation in an SSI configuration.

PORTType ACCESS | TRUNK
is the VSwitch-specific port type to determine whether VLAN tags should be visible to the guest

ACCESS
defines a connection that exchanges untagged frames with the guest. PORTTYPE ACCESS is only valid when the interface is configured for a single VLAN. z/VM adds (or removes) VLAN tags as necessary.

TRUNK
defines the type of connections that are established to be a trunk port. The guest is VLAN aware and sends and receives only tagged traffic for those VLANs to which the guest is authorized. If the guest is also authorized for the native VLAN untagged traffic sent or received by the guest is associated with the native VLAN ID of the virtual switch.

PQUPLINKTX LOW | NORMAL | HIGH
For a virtual switch with priority queuing enabled, PQUPLINKTX sets the priority for all packets sent from a NIC's network connection to an external network. If PQUPLINKTX is not specified, all outbound traffic to the external network will be sent at a normal priority on virtual switch uplink port. If PQUPLINKTX is configured for a virtual switch that does not have priority queuing enabled the setting will be saved and used if priority queuing is enabled at a later time. For a HiperSockets type NIC, the PQUPLINKTX operand is ignored.

LOW
specifies that outbound traffic to the external network will be sent at a low priority. This traffic will use the low priority queue which is serviced less frequently than the normal or high priority queues.

NORMAL
specifies that outbound traffic to the external network will be sent at a normal priority. This traffic will use the normal priority queue which is serviced less frequently than the high priority queue but more frequently than the low priority queue.

HIGH
specifies that outbound traffic to the external network will be sent at a high priority. This traffic will use the high priority queue which is serviced more frequently than the normal or low priority queues.

VLAN *vidset*
identifies the VLAN ID (or set of VLAN IDs) to which this user is restricted while connected to *switchname*. If VLAN is not specified, the default VLAN for this user is the default VLAN ID as specified on the DEFINE VSWITCH command or statement. Note that when a virtual switch is defined as VLAN AWARE, a default VLAN ID is not assigned. If a default VLAN ID is not assigned when the virtual switch is defined, then all inbound or outbound frames are discarded until a VLAN ID is assigned.

The *vidset* may be a simple VLAN ID (for example: "VLAN 1"), a VLAN range (for example: "VLAN 10-19"), or a complex set (for example: "VLAN 1 10-19 100-109"). A VLAN is a number between 1 and 4094.

Note: If the VLAN specification is too long to fit on a single line, the VLAN keyword must be repeated on a subsequent "NICDEF vdev" line to introduce each addition to the *vidset*.

PROmiscuous | NOPROmiscuous
When PROMISCUOUS is specified, the guest is authorized to enable promiscuous mode (allowing this interface to receive a copy of every network packet on the simulated LAN segment).

Usage Notes

1. When a simulated NIC adapter is defined, the NICDEF statement results in the creation of a series of I/O devices. The base device is validated by directory processing, but the remaining devices in the range are validated during LOGON processing. If another device is found in the range established by *vdev* and *devs*, the simulated NIC cannot be created.
2. It is possible to define a simulated NIC that will be automatically coupled to a guest LAN or virtual switch by adding the optional LAN parameter. However, if the designated guest LAN or virtual switch is not available when this user signs on, the COUPLE function cannot be performed. To make effective use of this feature, you must consider adding a DEFINE LAN or DEFINE VSWITCH statement in the SYSTEM CONFIG file to create the target guest LAN or virtual switch during system initialization.
3. z/VM supports virtual QDIO networking connections comprised of one read control device, one write control device, and up to eight data devices. This provides the ability to configure up to ten virtual devices per host QDIO connection.
4. LAN is required when a VSwitch-specific operand (PORTNUMBER, PORTTYPE, PQUPLINKTX, VLAN, PROMISCUOUS or NOPROMISCUOUS) is specified. Furthermore, the use of a VSwitch-specific operand restricts the virtual NIC to the designated network. When the virtual NIC is configured for a specific network, the CP COUPLE command will not allow a connection to any other network.
5. When Directory Network Authorization is disabled, PORTNUMBER, PORTTYPE, PQUPLINKTX, VLAN, PROMISCUOUS or NOPROMISCUOUS are ignored.
6. When Directory Network Authorization is enabled, and SET VSWITCH commands are used in conjunction with NICDEF statements to configure the network attributes, the following rules apply:
   a. No SET VSWITCH configuration is required if the NICDEF statement provides all necessary network configuration.
   b. NICDEF attributes override any prior SET VSWITCH configuration (and this is reflected in subsequent QUERY VSWITCH output).
   c. After a device is connected to the virtual switch, subsequent SET VSWITCH commands change the active configuration (but do not alter the USER DIRECT source).
   d. Each time the virtual NIC is created (or coupled) to the network, network attributes from the NICDEF statement are refreshed (replacing any dynamic changes made using the SET VSWITCH command).
7. For details about how this directory statement interacts with real CHPIDs on the LPAR and how the CHPID option can make that interaction more predictable, see "Channel path usage" on page 569.

Channel path usage

It is important to have a good understanding of how a Channel Path ID (CHPID) is associated with the set of simulated NIC devices created with a single invocation of this directory statement. A CHPID is a required element for the virtual devices created by this directory statement. If the CHPID operand is not specified on this directory statement, VM will automatically assign a CHPID. Except for guests defined with the OPTION CHPIDV user directory statement, a VM guest's set of CHPIDs is directly related to the VM LPAR's real set of CHPIDs. That is, the VM LPAR's real set of CHPIDs pass-through to the guest.

Since the devices created by this directory statement have no real device counterpart, VM uses an algorithm to assign a CHPID to these virtual devices that can coexist with VM's *real* set of CHPIDs. This algorithm first looks for a CHPID that is unassigned in both VM's *real* set of CHPIDs and the guest's set of CHPIDs. This lookup starts at *real* CHPID 0 and increases to 255 (xFF) until a match is found. If an unassigned CHPID is **not** found, a new lookup will start attempting to find a *real*, assigned CHPID that fulfills these requirements:

1. It has the same type needed by the devices created by this directory statement.
2. It is a CHPID not already assigned to the guest.

This lookup also starts at real CHPID 0 and increases to 255 (xFF) until a match is found. The NICDEF directory statement will fail if a match is **not** found.

This algorithm usually works seamlessly unless a dynamic I/O event occurs that impacts the *real* CHPID picked up and assigned to devices created by this directory statement. For example, if *real* CHPID 0 is assigned to guest virtual NICs 600-602, and a later dynamic I/O event adds *real* CHPID 0 with a non-compatible type, like a FICON CHPID, the guest using these virtual devices will likely start to experience error conditions. The errors likely will be experienced later when the guest operating system (OS) executes self-discovery code and notices the incompatible CHPID type. For example, a previously-working, second-level VM guest might start to encounter VSWITCH initialization errors on a re-IPL.

If a guest experiences a CHPID conflict due to a dynamic I/O change, one can shut down the guest OS, and then perform a "LOGOFF, LOGON" recycle of the VM userid to let VM locate a new *real* CHPID to associate with the virtual devices.

The best way to avoid these CHPID conflicts is to keep a free set of CHPIDs available that can be used with the CHPID operand on each invocation of this command. The rule is that each invocation of NICDEF must have a new available CHPID for the guest; note, however, that the same *real* CHPIDs can be used across different guests. The privileged Q CHPIDS command will display the *real* CHPID layout used by a VM LPAR. The guest OS will likely also have a means for determining its CHPID layout. For example, a second-level VM system can use the same Q CHPIDS command second-level.

If it is not possible to keep a free set of *real* CHPIDs available, remember that assigned CHPIDs can also be used if they are a compatible type. For example, an "Open Systems Adapter Direct Express" CHPID can be used for a NICDEF with the TYPE QDIO operands. An "Internal Queued Direct Communications" (IQDC) CHPID can be used for a NICDEF with the default TYPE HIPERSOCKETS operands. The type associated with a *real* CHPID can be determined using the privileged Q CHPID command with TYPE operand.

It should be noted that use of the OPTION CHPIDV user directory statement for a guest will avoid the pass-through of the real CHPID set to the guest. However, although use of CHPIDV completely virtualizes a guest's CHPID set, it should only be specified when a guest is going to be relocated and should not be specified for a z/OS guest.

Examples

1. Define a simulated QDIO adapter using I/O devices 0500 - 0507 (eight devices) which will be coupled to the SYSTEM-owned INEWS LAN during LOGON processing:

   ```
   NICDEF 500 TYPE QDIO DEV 8 LAN SYSTEM INEWS
   ```

2. Define a simulated HiperSockets adapter using I/O devices FD20 - FD2F (16 devices) which will be coupled to the user's own HSTEST LAN during LOGON processing:

   ```
   NICDEF FD20 TYPE HIPERS
   NICDEF FD20 DEVICES 16 LAN * HSTEST
   ```

   Note that this adapter cannot couple to the designated LAN during LOGON unless it has been defined earlier. This can be accomplished by adding the necessary DEFINE LAN statement to the SYSTEM CONFIG file.
3. Define a simulated QDIO adapter using I/O devices 0500 - 0502 that is configured with VSwitch-specific options for the SYSTEM-owned INEWS VSWITCH during LOGON processing:

   ```
   NICDEF 500 TYPE QDIO MACID 050021 LAN SYSTEM INEWS
   NICDEF 500 PORTTYPE TRUNK VLAN 1 100-121
   NICDEF 500 VLAN 200-221
   ```

NOPDATA Directory Statement

```
►► NOPDATA ►◄
```

Purpose

The NOPDATA statement authorizes a virtual machine to use NOP CCWs to transfer data to CP spool files. (Such data is not printed or punched with the file, and is not visible if the file is later read through a virtual card reader. It can only be accessed using DIAGNOSE code X'14'.)

How to Specify

The NOPDATA statement is allowed in profile, user, and identity entries. If you specify the NOPDATA statement, it must precede any device statements you specify in a directory entry. (For a list of device statements, see Table 24 on page 462.)

When processing the NOPDATA statement, DIRECTXA does not check for extra tokens. If you specify anything after the NOPDATA keyword, DIRECTXA ignores the extra tokens and system processing continues as if you had not specified the extra tokens.

Usage Notes

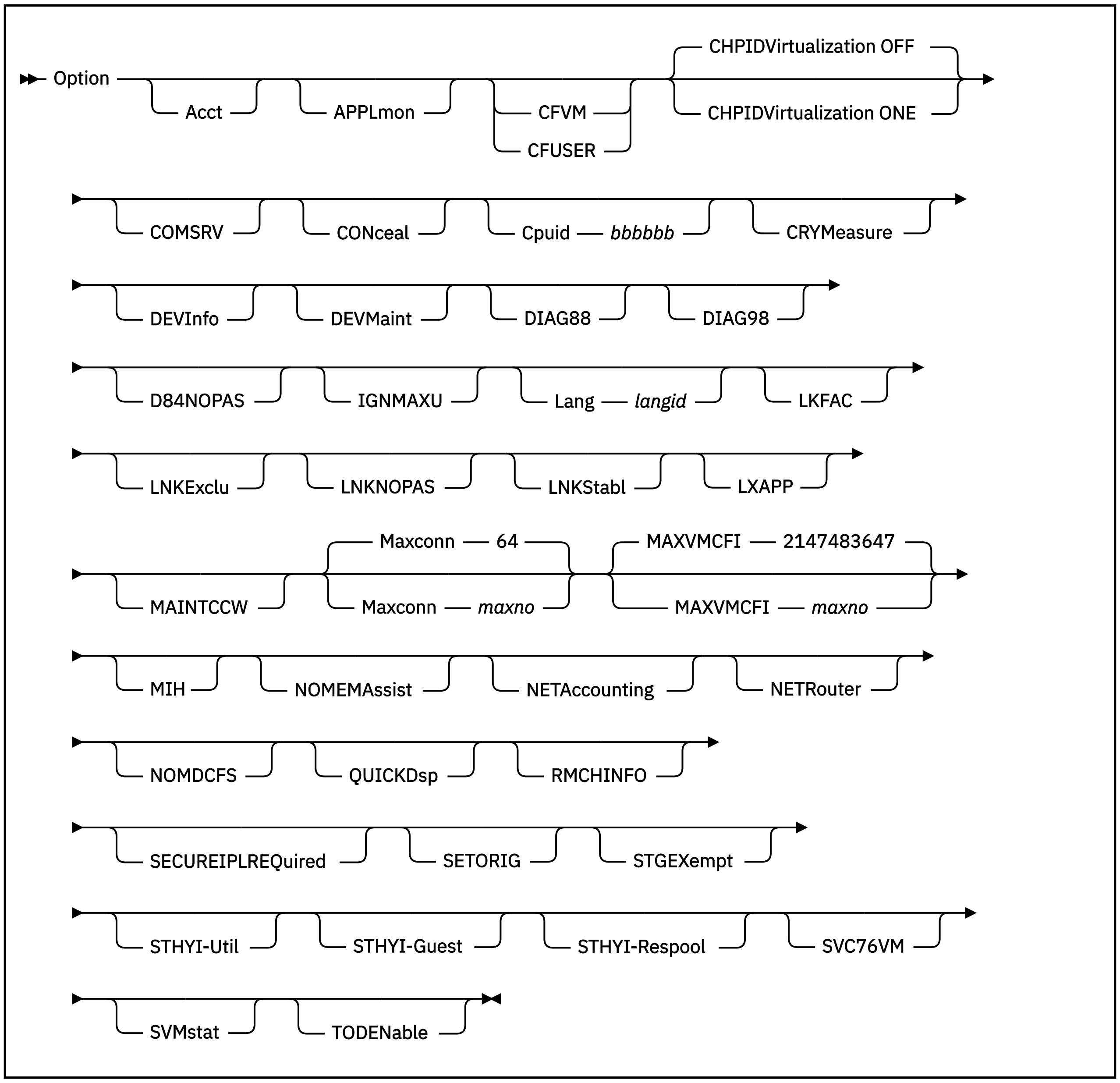
1. The NOPDATA statement should appear in the virtual machine definition of the RSCS virtual machine. For more information, see *z/VM: RSCS Networking Planning and Configuration*.
2. Do not use data chaining with NOP CCWs.

Examples

To authorize a virtual machine to transfer data to CP spool files using NOP CCWs, use the following NOPDATA statement in the virtual machine definition:

```
NoPData
```

OPTION Directory Statement



Purpose

The OPTION statement specifies special characteristics available to the virtual machine.

How to Specify

The OPTION statement is allowed in profile, user, identity, and subconfiguration entries; however, some operands are not allowed in all entries. See the description of each operand for restrictions. If you specify the OPTION statement, it must precede any device statements that you specify in a directory entry. (For a list of device statements, see Table 24 on page 462.)

Any number of OPTION statements are allowed in each entry. The options in each entry are added to any OPTION statements in other entries that make up the virtual machine instance when the user is logged on. If the option is specified with a value:

- A specification in a user or identity entry overrides a specification in the profile.
- A specification in a subconfiguration entry overrides a specification in the identity and included profile.

Additionally, within an entry, if a specific option keyword is specified more than once, then the associated value, if any, from the first occurrence is the one that is used and subsequent occurrences are ignored.

Operands

Acct

specifies that the virtual machine can issue a DIAGNOSE code X'4C' to generate accounting records. For more information, see DIAGNOSE Code X'4C' – Generate Accounting Records in *z/VM: CP Programming Services*.

The Acct operand is not allowed in subconfiguration entries.

APPLmon

authorizes the virtual machine to issue DIAGNOSE code X'DC' by which an application can declare or delete a buffer for CP monitoring. After the buffer has been declared for monitoring, all data in it is collected by CP into monitor records at each sample interval. For more information, see DIAGNOSE Code X'DC' – Control Application Monitor Record Collection in *z/VM: CP Programming Services*.

The APPLmon operand is not allowed in subconfiguration entries.

CFVM

defines the user as a virtual machine that will only run as a Coupling Facility Service Machine (CFVM). A CF Service Machine is a special disconnected virtual machine that is set up to IPL the Coupling Facility Control Code (CFCC). CFVM can not be specified with CFUSER, RMCHINFO, or CHPIDVirtualization.

The CFVM operand is not allowed in subconfiguration entries.

CFUSER

indicates that the user is allowed to define and couple Message Devices to a Coupling Facility Service Machine. CFUSER cannot be specified with CFVM or CHPIDVirtualizaiton.

The CFUSER operand is not allowed in subconfiguration entries.

CHPIDVirtualization

Only one CHPIDVirtualization option can be specified. If more than one is specified, the first is used and any subsequent ones are ignored.

OPTION CHPIDV in a user, identity, or included profile entry overrides GLOBALOPTS CHIPIDV for the user.

OFF

CHPID virtualization is not active for the virtual machine. For dedicated devices and minidisks, real channel path identifiers (CHPIDs) through which those devices are accessible are visible as virtual CHPIDs to the guest. This is the default.

ONE

Single path CHPID virtualization is active for the virtual machine. A single, virtualized CHPID is presented to the guest for dedicated devices and minidisks, independent of the real system CHPID. All online paths will still be used to support the guest I/O. This option is a requirement for a guest to be eligible for live guest relocation. For more information, see Chapter 31, "Preparing for Guest Relocations in a z/VM SSI Cluster," on page 753.

OPTION CHPIDV ONE is not allowed with OPTION CFVM or OPTION CFUSER, whether OPTION CHPIDV ONE is set within the definition by using the OPTION statement or is set by default from the GLOBALOPT setting.

This option should be specified only when the guest is going to be relocated. In addition to virtualizing the CHPID numbers, CHPIDV ONE results in DASD path group ID virtualization. CHPIDV ONE should not be specified for z/OS guests.

The CHPIDVirtualization operand is not allowed in subconfiguration entries.

COMSRV
authorizes the indicated virtual machine to act as a communication server, to:

- Route communications on behalf of other virtual machines to other servers
- Establish connections to other servers while handling requests for other users.

With this option, the TSAF virtual machine or any other communications server can put the user ID of the virtual machine that issued the APPC/VM CONNECT in the CONNECT parameter list.

When TSAF sends the connect request to the target virtual machine, the request contains this information about the originating virtual machine. Without this operand, CP would send the connect request with the communications server's user ID.

The COMSRV operand is not allowed in subconfiguration entries.

CONceal
specifies that the user's virtual machine is to be placed in the protected application environment when the user logs on. The environment remains active until the user logs off or until it is made inactive by a CP SET CONCEAL OFF or a CP DEFINE CPU command. This environment is intended for the application end user who does not want to interact with CP. Instead of presenting the user with an unexpected CP READ on occurrence of certain error conditions (for example, paging error, soft abend, disabled wait PSW loaded), the protected application facility forces an automatic re-IPL of the virtual machine. The re-IPL is initiated by using the IPL statement last used for the virtual machine.

To prevent a re-IPL loop, CP does not force a re-IPL if a previous automatic re-IPL has occurred less than a minute before. Not more than 10 automatic re-IPLs are tried between activation and deactivation of the protected application environment. If a re-IPL is not done, CP issues the same message(s) as when the protected application environment is not active and then puts the terminal into CP READ state.

A user whose virtual machine definition specifies OPTION CONCEAL has the break key (for 3270 terminals) disabled at logon. Subsequently, if the user deactivates the protected application environment by using the CP SET CONCEAL OFF command, the break key is enabled again.

Cpuid *bbbbbb*
specifies a processor identification number to be stored as part of the information stored by the STIDP instruction. This operand sets the CPU ID for all of the processors in a virtual MP configuration. This operand is overridden if a CPU ID is specified on the CPU statement for this virtual CPU. If the guest is relocated, this field does not change as a result of the relocation, even if the FORCE ARCHITECTURE or FORCE DOMAIN options were used on the VMRELOCATE command.

CRYMeasure
authorizes the specified virtual machine to obtain crypto measurement data from the crypto hardware on the real machine.

The CRYMeasure operand is not allowed in a subconfiguration entry.

DEVInfo
authorizes the specified virtual machine to use DIAGNOSE code X'E4' with subcodes X'00' and X'01' to access the relocation and real device information of the specified DASD that is owned by this or another user ID. For more information, see DIAGNOSE Code X'E4' - Return Minidisk Information/Define Full-Pack Overlay in *z/VM: CP Programming Services*.

The DEVinfo operand is not allowed in subconfiguration entries.

DEVMaint
authorizes the specified virtual machine to use DIAGNOSE code X'E4' with subcode X'02' and X'03' to get a full-pack overlay read/write minidisk of the volume on which the specified minidisk or cylinder/block resides. A user with this option is also authorized to execute any function protected by the DEVINFO and MAINTCCW options. For more information, see DIAGNOSE Code X'E4' - Return Minidisk Information/Define Full-Pack Overlay in *z/VM: CP Programming Services*.

The DEVMaint operand is not allowed in subconfiguration entries.

DIAG88
specifies that the virtual machine is authorized to use DIAGNOSE code X'88' to validate user authorizations and link minidisks. A user whose virtual machine definition contains this operand can run programs that access other users' minidisks without requiring passwords. For more information, see DIAGNOSE Code X'88' - Validate User Authorization/Link Minidisk in *z/VM: CP Programming Services*.

The DIAG88 operand is not allowed in subconfiguration entries.

DIAG98
specifies that the virtual machine is authorized to use the DIAGNOSE code X'98' real I/O facility. A user whose virtual machine definition contains this operand can run programs that use the page locking and start real I/O subfunctions to enhance I/O performance to dedicated devices. For more information, see DIAGNOSE Code X'98' - Real I/O in *z/VM: CP Programming Services*.

The DIAG98 operand is not allowed in subconfiguration entries.

D84NOPAS
specifies that this virtual machine has the ability to issue all subfunctions of DIAGNOSE code X'84' (except LOGPASS and MDISK) without the logon password of the target virtual machine ID. This option takes effect only if there is no access control mechanism, such as RACF, in place. For more information, see DIAGNOSE Code X'84' - Directory Update-in-Place in *z/VM: CP Programming Services*.

The D84NOPAS operand is not allowed in subconfiguration entries.

IGNMAXU
indicates that this virtual machine can log on to the system even if the number of users already logged-on is equal to or greater than the maximum allowed. You set this maximum in one of the following ways:

1. Using the FEATURES statement with the MAXUSERS operand in your system configuration file. For more information, see "FEATURES Statement" on page 158.
2. Using the CP SET MAXUSERS command. For more information, see SET MAXUSERS in *z/VM: CP Programming Services*.

The IGNMAXU operand is not allowed in subconfiguration entries.

Lang *langid*
specifies the 1- to 5-character name of the language that should be set for the virtual machine during logon.

LKFAC
indicates that the specified virtual machine is authorized to use:

- Multi-Path Lock Facility RPQ (MPLF) simulation support. MPLF is a lock facility that is simulated for TPF guests.
- Real Multi-Path Lock Facility (MPLF) support. MPLF is a lock facility available on IBM DASD subsystems.

This support allows multiple TPF guests, running in a loosely coupled configuration, to share a common database through simulation or real MPLF locking channel commands that are used by the TPF system.

The LKFAC operand is not allowed in subconfiguration entries.

LNKExclu
specifies that the virtual machine is authorized to use the stable and exclusive access modes, ER (exclusive read) or EW (exclusive write) of the CP LINK command, the CMS VMLINK command, and DIAGNOSE code X'E4'. This is a global authority that allows the virtual machine to perform stable or exclusive links to any minidisk to which it has password level authorization. For minidisk-specific authorization of stable and exclusive access modes, see "LINK Directory Statement" on page 533 and "MDISK Directory Statement" on page 550. The exclusive access modes negate the ability of other

users to acquire any access to the minidisk. This ensures that the issuing user is the only one with access to the minidisk.

The LNKExclu operand is not allowed in a subconfiguration entry.

LNKNOPAS
specifies that this virtual machine can link to any other virtual machine's DASD without password authorization. When LNKNOPAS is specified, password authorization can still be required when an external security manager (ESM) is installed. For more information, refer to the documentation provided by your ESM.

The LNKNOPAS operand is not allowed in a subconfiguration entry.

LNKStabl
specifies that the virtual machine is authorized to use the stable access modes, SR (stable read-only), and SW and SM (stable write), of the CP LINK command, the CMS VMLINK command, and DIAGNOSE code X'E4'. This is a global authority allowing the virtual machine to perform a stable link to any minidisk to which it has password-level authorization. For minidisk-specific authorization of stable and exclusive access modes, see "LINK Directory Statement" on page 533 and "MDISK Directory Statement" on page 550. These modes negate the ability of other users to acquire write access the specified minidisk. This ensures that the data cannot be changed underneath the user.

The LNKStabl operand is not allowed in subconfiguration entries.

LXAPP
specifies that the virtual machine can access the SE hard drive by using DIAGNOSE X'2C4'. For more information, see DIAGNOSE Code X'2C4' - FTP Services in *z/VM: CP Programming Services*.

MAINTCCW
authorizes the specified virtual machine to use diagnostic CCWs, including:

- Diagnostic Write Home Address
- Diagnostic Read Home Address
- Write Record Zero
- Write Home Address
- Diagnostic Load
- Diagnostic Write
- Diagnostic Sense/Read
- Diagnostic Control
- Perform Storage Controller ERP Action.

The MAINTCCW operand is not allowed in subconfiguration entries.

Maxconn *maxno*
specifies the maximum number of IUCV and APPC/VM connections that are allowed for this virtual machine. If the MAXCONN operand is omitted, the default is 64. The maximum is 65,535. The MAXCONN value has implications for users of file pool servers. For more information, see *z/VM: CMS File Pool Planning, Administration, and Operation*.

MAXVMCFI *maxno*
specifies the maximum number of VMCF inbound send messages, including those messages that are initiated by SMSG, plus IDENTIFY final response interrupts that might be queued for processing on this virtual machine. The variable *maxno* is a decimal number from 1 to 2147483647. If the MAXVMCFI operand is omitted, the default is 2147483647.

MIH
specifies that CP simulate an interrupt for the virtual machine whenever it detects a missing interrupt condition for an I/O operation it does on behalf of the virtual machine.

NOMEMAssist
disables the collaborative memory management assist for this virtual machine. If NOMEMASSIST is omitted, the default is to enable the assist if the real machine supports it; otherwise, the assist is disabled.

For more information, see SET MEMASSIST in *z/VM: CP Commands and Utilities Reference* and Collaborative memory management assist in *z/VM: CP Programming Services*.

NETAccounting
specifies that Network Data Transmission account records (type 0C) should be generated for this user. For more information, see Accounting Records Network Data Transmissions (Record Type C) in *z/VM: CP Programming Services*.

The NETAccounting operand is not allowed in subconfiguration entries.

NETRouter
specifies that this user is a network router, and Network Data Transmission account records (type 0C) should be generated for this user. Transmissions to and from a user who is designated as a router (over a guest LAN) is reported in account records that are separate from transmissions with non-routers. For more information, see Accounting Records Network Data Transmissions (Record Type C) in *z/VM: CP Programming Services*.

The NETRouter operand is not allowed in subconfiguration entries.

NOMDCFS
specifies that the virtual machine can use minidisk cache at a rate that is not limited by the fair share limit. This allows virtual machines that have a high I/O rate (such as the shared file system) to get the full benefit of minidisk cache.

QUICKDsp
causes a virtual machine to be added to the dispatch list immediately when it has work to do, without waiting in the eligible list.

Note: CP's virtual processor management ensures that no user stays in the eligible list more than an instant before the user is added to the dispatch list. Therefore, the QUICKDSP option is less meaningful, although it does get the virtual machine a different size elapsed-time slice.

RMCHINFO
indicates that the specified virtual machine is authorized to access real-machine configuration information, without regard to the virtual machine's configuration. It should be used when OSA/SF (or any other application that might require real machine configuration information) is running under CMS. RMCHINFO should not be specified for guest operating systems.

The RMCHINFO operand is not allowed in subconfiguration entries.

SECUREIPLREQuired
specifies that the virtual machine can be IPLed only by using a secure, list-directed IPL. See IPL in *z/VM: CP Commands and Utilities Reference* or "IPL Directory Statement" on page 519.

The SECUREIPLREQUIRED option is not enforced on a system that does not support the SECUREIPLREQUIRED option. Such a system can share an object directory with other systems that do support the SECUREIPLREQUIRED option. Until all systems where a user can log on support the SECUREIPLREQUIRED option, do not rely on the SECUREIPLREQUIRED option to enforce list-directed secure IPL for the user.

SETORIG
specifies that the virtual machine can issue DIAGNOSE code X'F8' subfunction code X'00' to associate originating node and user ID information with a virtual output device. For more information, see DIAGNOSE Code X'F8' - Spool File Origin Information in *z/VM: CP Programming Services*.

The SETORIG operand is not allowed in subconfiguration entries.

STGEXempt
specifies that the virtual machine is exempt from CP free storage limit detection. This ensures that the machine will not be suspended or forced off if it causes CP to consume too much free storage on its behalf. This option is recommended for non-general purpose virtual machines.

The STGEXempt operand is not allowed in subconfiguration entries.

STHYI-Util
authorizes the virtual machine to start utilization-related functions of the Store Hypervisor Information (STHYI) instruction.

This operand is not allowed in subconfiguration entries.

STHYI-Guest
authorizes the virtual machine to start guest related functions of the Store Hypervisor Information (STHYI) instruction.

This operand is not allowed in subconfiguration entries.

STHYI-Respool
authorizes the virtual machine to start resource-pool related functions of the Store Hypervisor Information (STHYI) instruction.

This operand is not allowed in subconfiguration entries.

SVC76VM
specifies that errors are not recorded by CP. All guest SVC 76 operations are processed by the virtual machine that issued the SVC 76.

See Usage Note “11” on page 579.

SVMstat
specifies that the virtual machine is a service virtual machine. The monitor data records associated with this virtual machine include the SVMSTAT setting. The only purpose is to allow products that process monitor data to report on service virtual machines separate from end-user virtual machines. No other operations, such as transaction or wait state classification, are affected by this operand.

The SVMstat operand is not allowed in subconfiguration entries.

TODENable
specifies that the user can change the virtual machine's Time-of-Day (TOD) clock. The TOD enablement characteristic is required only when the CP SET VTOD command is used with the FROMUSER or MSGPROC keywords.

Usage Notes

1. To run the device support facilities (ICKDSF) program in a virtual machine, you must specify the MAINTCCW operand.
2. The LKFAC option must be specified to authorize the user to use the SET LKFAC CP command to join a lock facility I/O configuration.
3. You cannot specify OPTION CONCEAL if the virtual machine definition for this user defines a multiprocessor configuration. The protected application facility supports only virtual uniprocessor systems.
4. Use of the IPL statement is recommended for users of the protected application facility. If the IPL statement is used with the OPTION CONCEAL statement and an appropriate SYSPROF EXEC, the user can be put directly into an application environment during logon. The user is thus protected from further CP interaction.
5. When virtual machines with OPTION CFVM in their virtual machine definitions are autologged in a VM mode logical partition, CP will change the types of their virtual CPUs to ICF.
6. Though the break key is initially disabled at logon when the OPTION CONCEAL statement is used, its setting can be enabled by using the TERMINAL BRKKEY command.

7. When a user's break key is disabled, the user whose virtual machine is running a full-screen application cannot enter CP commands or place the virtual machine into CP READ unless the application provides a way. The application can use DIAGNOSE code X'08' to pass CP commands or to pass no command at all to enter into CP READ.
8. The TERMINAL MODE CP command allows entry to CP on a console attention interrupt for users running in line mode. In addition, #CP can be used in this mode to enter CP commands and to enter into CP READ.
9. The installation default language is set for the virtual machine if:
   - The LANG option is not specified.
   - A valid CP message repository for the language that is specified by *langid* cannot be accessed.
10. STGEXempt ensures that the user ID is not subject to being suspended or forced due to the amount of CP free storage it causes CP to consume. This option is recommended for:
    - Special purpose user IDs vital to the installation
    - User IDs running trusted code
    - User IDs that should never be forced off.
11. Linux does not use SVC76 to record hardware errors. Therefore, if the guest has explicitly identified itself to z/VM as running Linux, then for that guest it is treated as if the setting SVC76VM was specified.

Examples

The following examples show various ways of specifying the OPTION statement. Note that you can specify more than one operand on a single OPTION statement.

1. To specify that:
   - The virtual machine can issue DIAGNOSE code X'4C'
   - 012345 is the virtual machine's processor identification number
   - CP is to inform the virtual machine of missing interrupts that it detects
   - The virtual machine is authorized to create up to 100 IUCV paths

   use the following OPTION statement in the virtual machine definition:

   ```
   Option   Acct  CpuID  012345  Mih  MaxConn  100
   ```
2. To specify that:
   - The virtual machine can issue DIAGNOSE codes X'4C' and X'98'
   - 012345 is the virtual machine's processor identification number
   - CP is to inform the virtual machine of missing interrupts that it detects
   - The virtual machine is authorized to create up to 100 IUCV paths

   use the following OPTION statement in the virtual machine definition:

   ```
   Option  Acct  CpuID  012345  Mih  MaxConn  100  Diag98
   ```
3. To place a virtual machine in the protected application environment when the user logs on, use the following OPTION statement in the virtual machine definition:

   ```
   Option Conceal
   ```
4. To indicate that the virtual machine is to be added to the dispatch list without waiting in the eligible list, use:

   ```
   Option QuickDsp
   ```

5. To indicate that the virtual machine can issue DIAGNOSE code X'F8', subfunction code X'00', use:

   ```
   Option SetOrig
   ```

6. To indicate that the virtual machine can issue DIAGNOSE code X'DC', use:

   ```
   Option ApplMon
   ```

7. To give the virtual machine the authority to enter a CP LINK *userid* 291 392 SR command, use:

   ```
   Option LnkStabl
   ```

8. To allow a virtual machine to operate as a communications server, use:

   ```
   Option ComSrv
   ```

9. To give the virtual machine the authority to enter a CP SET LKFAC command, use:

   ```
   Option LKFAC
   ```

POOL Directory Statement

```
►► POOL ── LOW ── lowbound ── HIGH ── highbound ── PROFile ── name ─►◄
```

Purpose

The POOL statement allows a set of virtual machines to be defined with the same configuration or characteristics.

How to Specify

The POOL statement is allowed in user and identity entries. If you specify the POOL statement, it must be the only directory statement in the entry.

Operands

LOW *lowbound*
: specifies the suffix number for the user ID of the first virtual machine in the pool. The variable *lowbound* must be a decimal number from 0 to 99999.

HIGH *highbound*
: specifies the suffix number for the user ID of the last virtual machine in the pool. The variable *highbound* must be a decimal number from *lowbound* to 99999.

PROFile *name*
: specifies the name of the profile to be used for the definition of each virtual machine in the pool.

Usage Notes

1. The directory statement before a POOL statement must be a USER or IDENTITY statement.
2. The user ID specified on the preceding USER or IDENTITY statement must be from 1 to 3 characters in length.

Examples

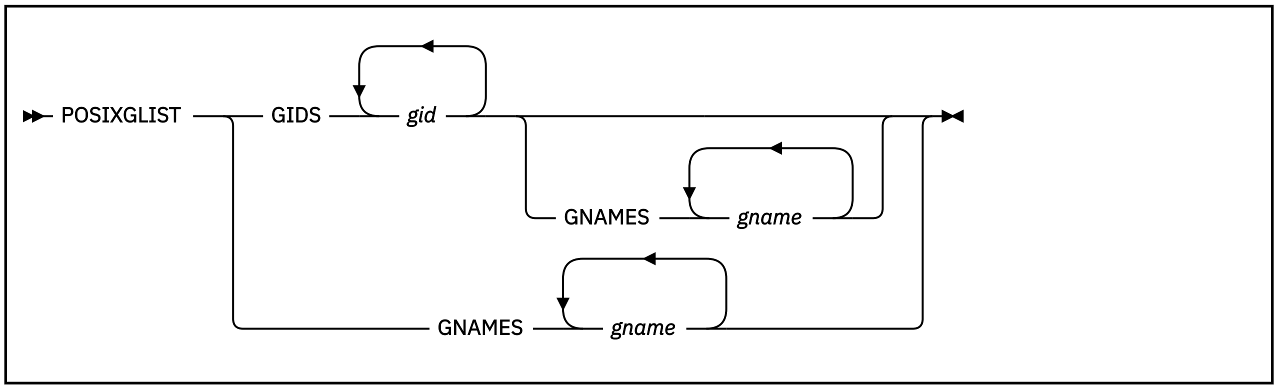
A group of 25 virtual machines with identical configurations could be defined by the following directory entry:

```
User GRP AutoOnly 4m 32m
     Pool Low 1 High 25 Profile groupdef
```

This will generate directory entries for virtual machines GRP00001 through GRP00025.

Each machine would have a password of AUTOONLY, thus restricting it from terminal logon. Each virtual machine would have 4 MB of virtual storage, which may be reset up to a maximum of 32 MB. Also, each virtual machine would have a privilege class of G. The rest of the configuration for each machine would be as defined in the directory profile named GROUPDEF.

POSIXGLIST Directory Statement



Purpose

The POSIXGLIST statement lists POSIX groups of which the user is a member. Each group on the list can be specified by either group ID (*gid*) or group name (*gname*). POSIXGLIST statements are the primary source for the GIDs that form the user's supplementary group list.

How to Specify

The POSIXGLIST statement is allowed in a profile, user, or identity entry. This statement must precede any device statements you specify in a directory entry. (For a list of device statements, see Table 24 on page 462.)

Multiple POSIXGLIST statements are allowed within a user, identity, or profile entry. The POSIXGLIST information from a user or identity completely replaces that from an included profile.

The POSIXGLIST statement may be continued across multiple records in the source directory file. For more information about continued statement rules, see "Continued Directory Statements" on page 467. Note that some operands on this statement are case sensitive, so care should be taken to preserve the case of them when editing the source directory.

Operands

GIDS *gid*
each *gid* must identify a group defined on a POSIXGROUP statement or the default group implicitly defined by DIRECTXA.

GNAMES *gname*
specifies, by group name, POSIX groups of which the user is a member. Each *gname* must identify a group defined on a POSIXGROUP statement or the default group implicitly defined by DIRECTXA. The case of each group name is preserved; it is not converted to upper case by DIRECTXA. Each name must match exactly the group name on a POSIXGROUP statement or the default group name.

Usage Notes

1. If your installation has installed an External Security Manager (ESM), the information specified on this directory statement may be overridden by information provided by the ESM. Consult your ESM documentation for more information.
2. A group's *gid* or *gname* can be specified only one time by GID or GNAME on the POSIXGLIST statements in a user, identity, or profile entry; that is, duplicate *gid*s or *gname*s are not permitted.
3. DIRECTXA does not check for duplicate groups across the GIDS-GNAMES lists.

4. The user's primary group defined by a POSIXINFO statement may be listed on a POSIXGLIST statement, but it is not required to be listed. A user is automatically considered a member of his/her primary group.
5. If there is more than one group defined with the same GID, and the user is a member of any of these groups, then it is recommended that group names be used to list these groups on POSIXGLIST in order to avoid an ambiguous specification.
6. The user's supplementary GID list consists of up to 32 unique GIDs. The primary GID (from the POSIXINFO statement) is always included in the initial list. The remainder are taken from the POSIXGLIST statement(s) in the order in which they were specified.

Examples

1. To define user ID rick to be a member of POSIX groups VMCMS, VMCFT and VMCP, code following POSIXGLIST statement:

```
Globaldefs
 PosixGroup VMCFT 100
 PosixGroup VMCMS 200
 PosixGroup VMCP 300
 :
 User rick ...
 PosixGList gids 200 gnames VMCFT VMCP
```

POSIXGROUP Directory Statement

```
►► POSIXGROUP — gname — gid ►◄
```

Purpose

The POSIXGROUP statement defines the group name and group ID (GID) for a POSIX group.

How to Specify

When a POSIXGROUP statement is specified, it must be within the global definition section of the source directory (after the GLOBALDEFS directory statement and before any user, identity, or profile entries). There are no restrictions on the number of POSIXGROUP statements that may be specified. Note that some operands on this statement are case sensitive, so care should be taken to preserve the case of them when editing the user directory file.

Operands

gname

specifies a unique 1- to 8-character mixed-case POSIX group name. The single or double quotation mark is invalid in this operand. The case of this operand is preserved; it is not converted to upper case by DIRECTXA. Any group name specified on other directory statements that refer to this group must match this group's name exactly.

The group name DEFAULT identifies the default group and is reserved by DIRECTXA; it may not be specified on a POSIXGROUP statement. DIRECTXA implicitly defines a default group with a name of DEFAULT and a GID of 4294967295 (X'FFFFFFFF'). This name or GID may be specified on POSIXINFO and POSIXGLIST statements to identify the default group.

gid

specifies the POSIX group ID (GID) associated with *gname*. The *gid* is a numeric value from 0 to 4294967295 (X'FFFFFFFF').

Usage Notes

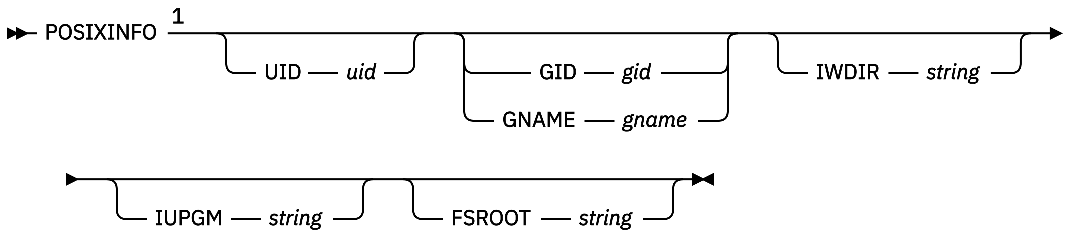
1. If your installation has installed an External Security Manager (ESM), the information specified on this directory statement may be overridden by information provided by the ESM. Consult your ESM documentation for more information.
2. Duplicate group names are not permitted, but multiple groups may have the same GID. If you define multiple groups with the same GID, do so with caution, because GIDs are used for various authority checks. Also, certain queries that require a GID as input may return information about a group other than the intended one.
3. It is permitted to define a group with no members. That is, a POSIXGROUP statement may define a group that no users identify as one of which they are a member.

Examples

1. To define a POSIX group named VMcp and with a GID of 200, code the following POSIXGROUP statement:

```
Globaldefs
  PosixGroup VMcp 200
```

POSIXINFO Directory Statement



Notes:

[1] You must specify at least one of the following operands. If you want to specify more than one operand, you can specify them in any order.

Purpose

The POSIXINFO statement specifies a user ID's POSIX information. It contains POSIX user database information such as POSIX user ID (UID), POSIX group ID (GID) or group name, initial working directory, initial user program and file system root.

How to Specify

If you specify the POSIXINFO statement, it must precede any device statements you specify in a profile, user, or identity entry. (For a list of device statements, see Table 24 on page 462.)

Multiple POSIXINFO statements are allowed within a user, identity, or profile entry. Each operand can be specified only once within a user, identity, or profile entry. POSIXINFO values within a user or identity entry override those in a profile entry.

The POSIXINFO statement can be continued across multiple records in the source directory file. For more information about continued statement rules, see "Continued Directory Statements" on page 467. Note that some operands on this statement are case sensitive, so care should be taken to preserve the case of them when editing the source directory.

Operands

UID *uid*
: specifies the POSIX user ID (UID) assigned to this user. This will be the user's real UID, effective UID and saved set-UID when the user logs on. Care should be taken in assigning zero as the UID, because UID zero is considered to denote *appropriate privileges*. A user with *appropriate privileges* can perform many authorized POSIX functions and will pass many security checks. The *uid* is a numeric value between 0 and '4294967295 (X'FFFFFFFF'). A default value of 4294967295 (X'FFFFFFFF') is assigned if there is no UID specification for a user.

GID *gid*
GNAME *gname*
: specifies the user's primary group. The *gid* is a numeric value between 0 and 4294967295 (X'FFFFFFFF'). The *gname* is a 1- to 8-character mixed-case POSIX group name. The single or double quotation mark is invalid in the *gname*. The case of the group name is preserved; it is not converted to upper case by DIRECTXA. It must match exactly the group name on a POSIXGROUP statement or the default group name implicitly defined by DIRECTXA. The *gid* or *gname* must identify a group defined on a POSIXGROUP directory statement or the default group name implicitly defined by DIRECTXA. The GID for this group will be the user's real GID, effective GID and saved-set GID when the user logs on.

You may specify the group by either GID or GNAME, but not both. If GNAME is specified, the user's primary GID is obtained from the POSIXGROUP statement defining the group with name *gname*. If GID is specified, the user's primary group name is obtained from a POSIXGROUP statement defining a group with GID *gid*. If there is more than one group with this GID, then one of them is chosen as the user's primary group; it is unpredictable which one is chosen.

If there is no GID or GNAME specification for a user, the user's primary group is the default group, named DEFAULT with GID 4294967295 (X'FFFFFFFF').

IWDIR *string*
specifies the user's initial working directory. Its rules for specification are described below.

IUPGM *string*
specifies the user's initial user program. Its rules for specification are described below.

FSROOT *string*
specifies the user's file system root. Its rules for specification are described below.

The *string* specifying an IWDIR, IUPGM or FSROOT results in a mixed-case 1- to 1023-character string which may contain blanks, single quotation marks, double quotation marks and other special characters. The *string* begins with the first nonblank character following its keyword and may be continued on multiple directory records. If you do not want imbedded blanks or quotation marks in the result, and *string* fits on a single directory record, you may specify *string* as a single blank-delimited token. Otherwise, it must be specified as a *quoted string operand*. For more information about continued statement rules, see "Quoted String Operands" on page 468.

Usage Notes

1. If your installation has installed an External Security Manager (ESM), the information specified on this directory statement may be overridden by information provided by the ESM. Consult your ESM documentation for more information.
2. UIDs are not required to be unique. The same value can be assigned to multiple users, but this is not recommended. If you define multiple users with the same UID, do so with caution, because UIDs are used for various authority checks and individual user control and accountability will be lost.
3. POSIXINFO statements are permitted in profile entries, but you should not specify the UID in a profile entry because it is likely to result in many users with the same UID. Since UIDs are used to identify individuals for the purposes of authorization and permission checking, extreme care should be used when assigning UIDs through a profile entry. Similar consideration should be given to assigning GID/GNAME through a profile entry.
4. POSIX *user names*, sometimes referred to as *login names*, are defined to be the lower case version of the user's VM user ID.

Examples

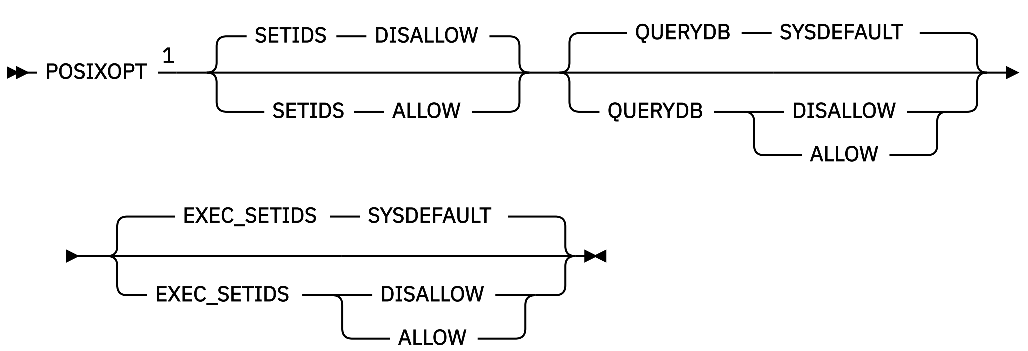
1. To define a userid CLYDE with 100 as UID, 200 as primary GID, /home/clyde as initial working directory, pxshell as initial user program, and /../VMBFS:VMSYS:ROOT/ as the file system root, code the following directory statements:

```
Globaldefs
 PosixGroup VMcp 200
 :
 User clyde ...
 PosixInfo uid 100 gid 200 iwdir /home/clyde ,
           iupgm pxshell
 PosixInfo fsroot /../VMBFS:VMSYS:ROOT/
```

2. Assume you have several virtual machines, among them are SUE and DAMIAN, that have the same initial user program pxshell, and /../VMBFS:VMSYS:ROOT/ as the file system root. SUE's UID is 200, primary group has an ID of 101, and initial working directory /home/sue. DAMIAN's UID is 100, primary group is named VMcp, and initial working directory /home/damian. Code the following directory statements:

```
Globaldefs
 PosixGroup VMcms 100
 PosixGroup VMcp 200
 :
 Profile posixdef
 PosixInfo iwdir /home iupgm pxshell fsroot /../VMBFS:VMSYS:ROOT/
 :
 User sue ...
 Include posixdef
 PosixInfo uid 200 gid 100 iwdir /home/sue
 :
 User damian ...
 Include posixdef
 PosixInfo uid 101 gname VMcp iwdir /home/damian
```

POSIXOPT Directory Statement



Notes:

[1] You must specify at least one of the following operands. If you want to specify more than one operand, you can specify them in any order.

Purpose

The POSIXOPT statement specifies option settings related to a user ID's POSIX capabilities. It includes authorization to query and set POSIX process and database information.

How to Specify

If you specify the POSIXOPT statement, it must precede any device statements you specify in a profile, user, or identity entry. (For a list of device statements, see Table 24 on page 462.)

Multiple POSIXOPT statements are allowed within a user, identity, or profile entry. Each operand can be specified only once within a user, identity, or profile entry. POSIXOPT values within a user or identity entry override those in a profile entry.

The POSIXOPT statement can be continued across multiple records in the source directory file. For more information about continued statement rules, see "Continued Directory Statements" on page 467.

Operands

SETIDS
specifies whether the user is authorized to set other users' POSIX UIDs or GIDs.

DISALLOW
(the default) specifies the user is not allowed to set other users' POSIX security values.

ALLOW
specifies the user is allowed to set other users' POSIX security values on behalf of CP *exec()* processing. The users whose IDs are to be set must have POSIXOPT EXEC_SETIDS ALLOW specified in their virtual machine definitions. You should generally specify this option (SETIDS ALLOW) only for POSIX byte file system (BFS) servers.

QUERYDB
specifies whether the user is authorized to query other users' POSIX database information.

SYSDEFAULT
(the default) specifies the system default authorization for querying POSIX database information.

The system default authorization (SYSDEFAULT) is set up by the USER_DEFAULTS system configuration file statement. For more information, see "USER_DEFAULTS Statement" on page 309. If the QUERYDB option is specified, it will override the system default. If it is not specified, then the system default applies.

DISALLOW
specifies the user is not allowed to query other users' POSIX database information.

ALLOW
specifies the user is allowed to query other users' POSIX database information.

EXEC_SETIDS
specifies whether the user is authorized to have his POSIX security values changed on behalf of a POSIX *exec()* function call designating a file with the *setuid* or *setgid* attributes. The byte file system server must have POSIXOPT SETIDS ALLOW specified in its virtual machine definition.

SYSDEFAULT
(the default) specifies the system default for having a user's POSIX security values changed by other users.

The system default authorization (SYSDEFAULT) is set up by the USER_DEFAULTS system configuration file statement. For more information, see "USER_DEFAULTS Statement" on page 309. If the EXEC_SETIDS option is specified in the directory, it will override the system default. If it is not specified, then the system default applies.

DISALLOW
specifies the user is prohibited from having his POSIX security values changed on behalf of CP *exec()* processing.

ALLOW
specifies the user is permitted to have his POSIX security values changed on behalf of CP *exec()* processing.

Usage Notes

1. If your installation has installed an External Security Manager (ESM), the information specified on this directory statement may be overridden by information provided by the ESM. Consult your ESM documentation for more information.
2. If a user has EXEC_SETIDS DISALLOW specified, then any requests to change this user's POSIX security values on behalf of CP *exec()* processing will be rejected even if the requestor is authorized to change POSIX security values.

Examples

The following examples show various ways of specifying the POSIXOPT statement. Note that you can specify more than one operand on a single POSIXOPT statement.

1. To specify that:
   - A user can set any other users' POSIX security values
   - The user can query any other users' database information
   - The user prohibits having his POSIX security values changed by other users on behalf of CP *exec()* processing.

   use the following POSIXOPT statement in the virtual machine definition:

   ```
   PosixOpt SetIds allow QueryDb allow Exec_Setids disallow
   ```

2. To specify that:
   - The user cannot set other users' POSIX security values
   - The user can not query other users' database information
   - The user takes the system default for the Exec_Setids option

use the following POSIXOPT statement in the virtual machine definition:

```
PosixOpt QueryDb disallow
```

PROFILE Directory Statement

```
►►─ Profile ─ name ─►◄
```

Purpose

A PROFILE statement defines the start of a profile entry in the source directory. Profile entries represent defaults to be set if not specified in the associated user, identity, or subconfiguration entries.

How to Specify

When PROFILE is specified, all profile entries must follow the last DIRECTORY statement and the global definition entry and precede the first USER, IDENTITY, or SUBCONFIG statement. There are no restrictions on the number of profile entries that can be specified.

Directory profiles are implemented by defining a profile entry in the source directory and specifying an INCLUDE statement in the user or identity entries that refer to the profile. For more information, see "INCLUDE Directory Statement" on page 516.

Operands

name
specifies the name assigned to the profile entry and is used to refer to the profile. The variable *name* is an alphanumeric string of up to 8 characters. A valid character is any character that can be used in a user ID name. Only one profile name can be specified on a PROFILE statement.

Usage Notes

1. When directory statements are used in a profile entry, they perform the same function as they do when used in a user, identity, or subconfiguration entry. However, they might operate slightly differently when used in both the profile and user, identity, or subconfiguration entries.
2. The following statements, when placed in the user or identity entry, completely supersede their profile counterparts:

ACCOUNT	IOPRIORITY	SHARE
ACIGROUP	IPL	SPOOLFILE
AUTOLOG	LOADDEV	STDEVOPT
CLASS	LOGONBY	STORAGE
CONSOLE	MACHINE	VMRELOCATE
CRYPTO	MAXSTORAGE	XAUTOLOG
DATEFORMAT	NICDEF	
D8ONECMD	POSIXGLIST	

3. Area operands of the SCREEN statements in the user, identity, or subconfiguration entry supersede their profile counterparts. This relationship also holds true when the user, identity, or subconfiguration entry redefines a superset or subset of those areas defined by the profile.
4. STDEVOPT, POSIXINFO, and POSIXOPT statement operands explicitly coded in a user or identity entry supersede their profile counterparts. However, these statements' operands explicitly coded in a profile supersede uncoded defaults in a user or identity entry.
5. If the CPU number is the same in both the user or identity entry and the profile entry, CPU statement from the user or identity entry supersedes the one in the profile entry. If the CPU numbers

are different, however, the statements are simply appended. The order of concatenation is CPU statements from the user or identity entry followed by those in the profile entry.

6. The XCONFIG statement, when placed in the user, identity, or subconfiguration entry, supersedes its profile counterpart if the first (major) operand is the same; if the first operands are different, the definitions are combined.
7. When the following directory statement types are used in a profile entry, they are processed exactly as if they were multiple occurrences of the statement types within the including user or identity entry. (See Usage Note "5" on page 591.)

APPCPASS	NAMESAVE	OPTION
IUCV	NOPDATA	

8. When the following directory statement types are used in a profile entry, they are processed as normal except that a duplicate definition of an address in the profile entry causes a duplication error. However, duplicating an address from the profile with an address used in the including user, identity, or subconfiguration entry simply creates another device definition to be resolved at logon time. That is, at logon, the including user's device definition for the duplicated address is processed first; if this definition fails, the profile device definition for the duplicated address is attempted.

CONSOLE	DEDICATE	SPECIAL
DASDOPT	LINK	SPOOL

9. The following directory statements are not allowed in a profile entry:

BUILD	INCLUDE	POSIXGROUP
CRYPTO (See note 1)	LOAD	PROFILE
DIRECTORY	MDISK	SUBCONFIG
GLOBALDEFS	MINIOPT	SYSAFFIN
GLOBALOPTS	POOL	USER
IDENTITY		

[1] CRYPTO APVIRT is allowed in profile entries. CRYPTO operands other than APVIRT are not allowed in profile entries.

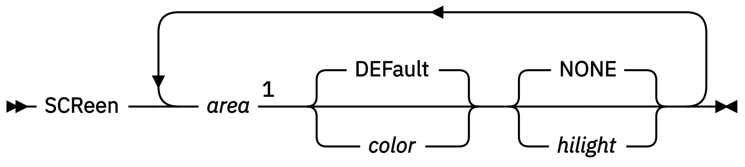
10. A profile entry begins with a PROFILE statement and ends with the next PROFILE statement or the first USER, IDENTITY, or SUBCONFIG statement.

Examples

For this example, assume you have several virtual machines that are CMS users and that are also in a ACIGROUP called GROUP1. JOHN1 is a user in this group. Follow the directory statement by the following profile:

```
Directory vdev devtype volid
Profile grp1user
     AciGroup group1
     Ipl cms
       :                ← other directory statements common to this group
User john1 …                  ← USER statement operands for this user
     Include grp1user
       :                       ← directory statements unique to this user
     Link jack 495 495 rr     ← for example, a link that this user needs
       :                       but other users in this group do not need
User …
       :
```

SCREEN Directory Statement



Notes:

[1] You must specify at least one of the operands. If you want to specify more than one, you can specify them in any order.

Purpose

SCREEN statements in a virtual machine definition define the extended color and extended highlighting options for the virtual machine console.

How to Specify

The SCREEN statement is allowed in a profile, user, identity or subconfiguration entry. If specified, SCREEN statements must precede any device statements in an entry. (For a list of device statements, see Table 24 on page 462.) You can include as many SCREEN statements as you need to define any areas of the screen you want, but the SCREEN statements must be contiguous.

Any number of SCREEN statements are allowed within a profile entry, and are added to any SCREEN statements in the user, identity, or subconfiguration entry. Duplicate area specifications within a profile are not allowed. The user or identity entry overrides a profile entry for any user-to-profile duplicate area specification. This relationship also holds true when the user identity entry redefines a superset or subset of those areas defined by the included profile.

If a SCREEN statement is contained within a subconfiguration entry, then that set of SCREEN statements replaces either explicitly or implicitly any settings for those specific areas that were also set by SCREEN statements in the identity entry. The result is a combination of settings specified in the subconfiguration, identity, and profile entries, with the identity overriding the profile and the subconfiguration overriding the identity for each specific area.

For example, if ALL is specified in the profile as BLUE and BLINK, OUTAREA is specified in the identity as GREEN and REVVIDEO, and VMOUT is specified in the subconfiguration as RED and UNDERLIN, then the settings are:

- INAREA: BLUE and BLINK
- STATAREA: BLUE and BLINK
- CPOUT: GREEN REVVIDEO
- VMOUT: RED and UNDERLIN
- INREDISP: GREEN and REVVIDEO

Operands

area
is the area of the screen. Area can be:

ALL
designating the entire screen. If this area is specified, none of the following areas can be used.

INArea
the input area.

STAtarea
: the system status area.

OUTarea
: the output areas. If this area is specified, none of the following areas can be used.

CPOut
: the output from CP.

VMOut
: the output from CMS or the virtual machine operating system running in the user's virtual machine.

INRedisp
: the input redisplay.

color
is the color attribute to be assigned to an area of the screen. Color can be:

BLUe	GREen	PINk	RED
TURquoise	WHIte	YELlow	DEFault (green and white)

hilight
is the extended highlight value to be assigned to an area of a screen. The hilight value can be:

BLInk
: blinking

NONe
: (the default) no extended highlighting

REVvideo
: reverse video

UNDerlin
: underlining

Usage Notes

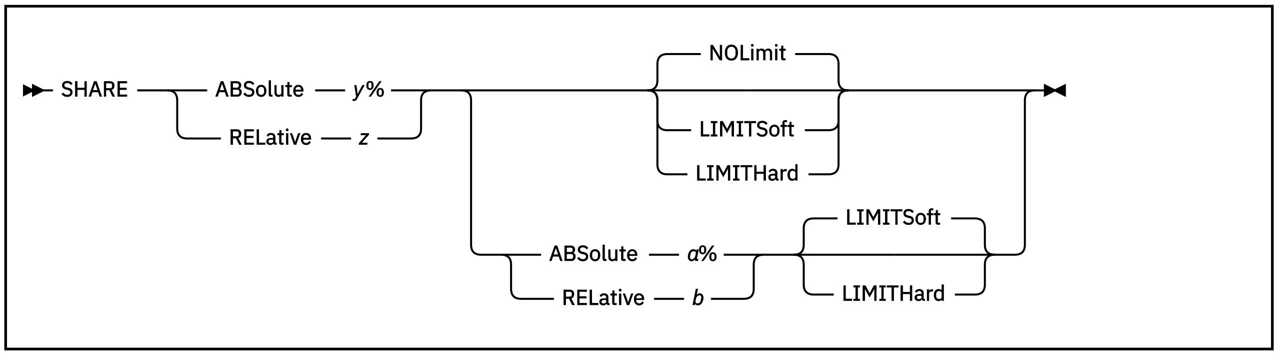
1. A default value of NONE is applied for any unspecified extended highlight attribute. DEFAULT is used for any unspecified extended color attribute. The DEFAULT color is monochrome (green and white).
2. If the ALL operand is used, it must be the only operand specified on the only SCREEN statement for the user.
3. If the OUTAREA operand is used, CPOUT, VMOUT, or INREDISP cannot be specified. If CPOUT, VMOUT, or INREDISP is specified, the OUTAREA operand cannot be specified.
4. No SCREEN statement operands can appear more than once in a b12rctdirectory entry.

Examples

To define the screen output area of a user's virtual console as red and blinking, and the input area yellow and reverse video, use the following SCREEN statement in the virtual machine definition:

```
SCReen  OutArea red blink  InArea RevVideo yellow
```

SHARE Directory Statement



Purpose

The SHARE statement specifies a virtual machine's share of CPU power.

How to Specify

The SHARE statement is allowed in profile, user, identity, and subconfiguration entries. If you specify the SHARE statement, it must precede any device statements you specify in a directory entry. (For a list of device statements, see Table 24 on page 462.)

You can specify only one value (ABSOLUTE or RELATIVE) per statement. Though allowed, you should not put multiple SHARE statements in a directory entry. If multiple SHARE statements are found in a directory entry, only the last one is used.

Multiple SHARE statements are allowed within a profile entry, but are used only if no SHARE statements are in the including user, identity, or subconfiguration entry. If multiple SHARE statements are in an included profile, only the last one is used.

A SHARE statement in a subconfiguration entry overrides one in the identity entry.

If you specify more than five tokens, DIRECTXA ignores the extra tokens and system processing continues as if you had not specified the extra tokens.

Operands

ABSolute *y%*
: specifies the absolute share of all the active processors in a system. The variable *y* is a decimal real number—no or one decimal place—from 0.1 to 100 (for example, 20.5%, or 80%).

RELative *z*
: specifies a relative share of the system. The variable *z* is a decimal integer between 1 and 10000.

ABSolute *a%*
: specifies a user’s maximum absolute share of all the active processors in a system. The variable *a* is a decimal real number—no or one decimal place—from 0.1 to 100 (for example, 20.5%, or 80%). If ABSOLUTE *y*% was specified, *y* must be less than or equal to *a*.

RELative *b*
: specifies a user’s maximum relative share of the system. The variable *b* is a decimal integer between 1 and 10000. If RELATIVE *z* was specified, *z* must be less than or equal to *b*.

NOLimit
: specifies that a user’s share of processing resource is not limited.

LIMITSoft
: specifies that a user’s share of processing resource is limited. However, there are times when LIMITSOFT users will receive more than their limit. This occurs when some users are not using all of their shares (for example, they are waiting for I/O to complete), and there are no NOLIMIT users nor

users who have not yet reached their limit and can use additional processing resource. If a maximum share is not specified, the minimum share is also the maximum.

LIMITHard
specifies that a user's share of processing resource is limited. When LIMITHARD is specified, a user will not receive more than its maximum share of the processing resource. If a maximum share is not specified, the minimum share is also the maximum.

Use the SET SRM LIMITHARD command to set the method that the scheduler will use to enforce the limit on the guest's CPU usage. For more information, see SET SRM in *z/VM: CP Commands and Utilities Reference*.

Usage Notes

1. If you do not specify a SHARE statement for a virtual machine, CP assigns that virtual machine a relative share of 100.
2. For more information on scheduler shares, see *z/VM: Performance*.
3. If a limit is specified without a maximum share being specified, the user's minimum share is also its maximum share.
4. For more information about setting the SHARE of a userid that has multiple CPUs in its configuration, see SET SHARE in *z/VM: CP Commands and Utilities Reference*.
5. Share settings control access to CPU power on a system-wide, per-processor-type basis, but they have no bearing or influence on how the power associated with a RESPOOL is distributed to its members. When the RESPOOL limit is reached, all members of the pool are put onto the limit list for a while, and then, after a delay sufficient to compensate for any excess use, they are all removed. The scheduler makes no effort to distribute the power of the RESPOOL to its members in accordance with the members' share settings.

Examples

1. To assign a minimum absolute share of 50%, use the following SHARE statement in the virtual machine definition:

   ```
   Share Absolute 50%
   ```

2. To assign a minimum relative share of 200, use the following SHARE statement in the virtual machine definition:

   ```
   Share Relative 200
   ```

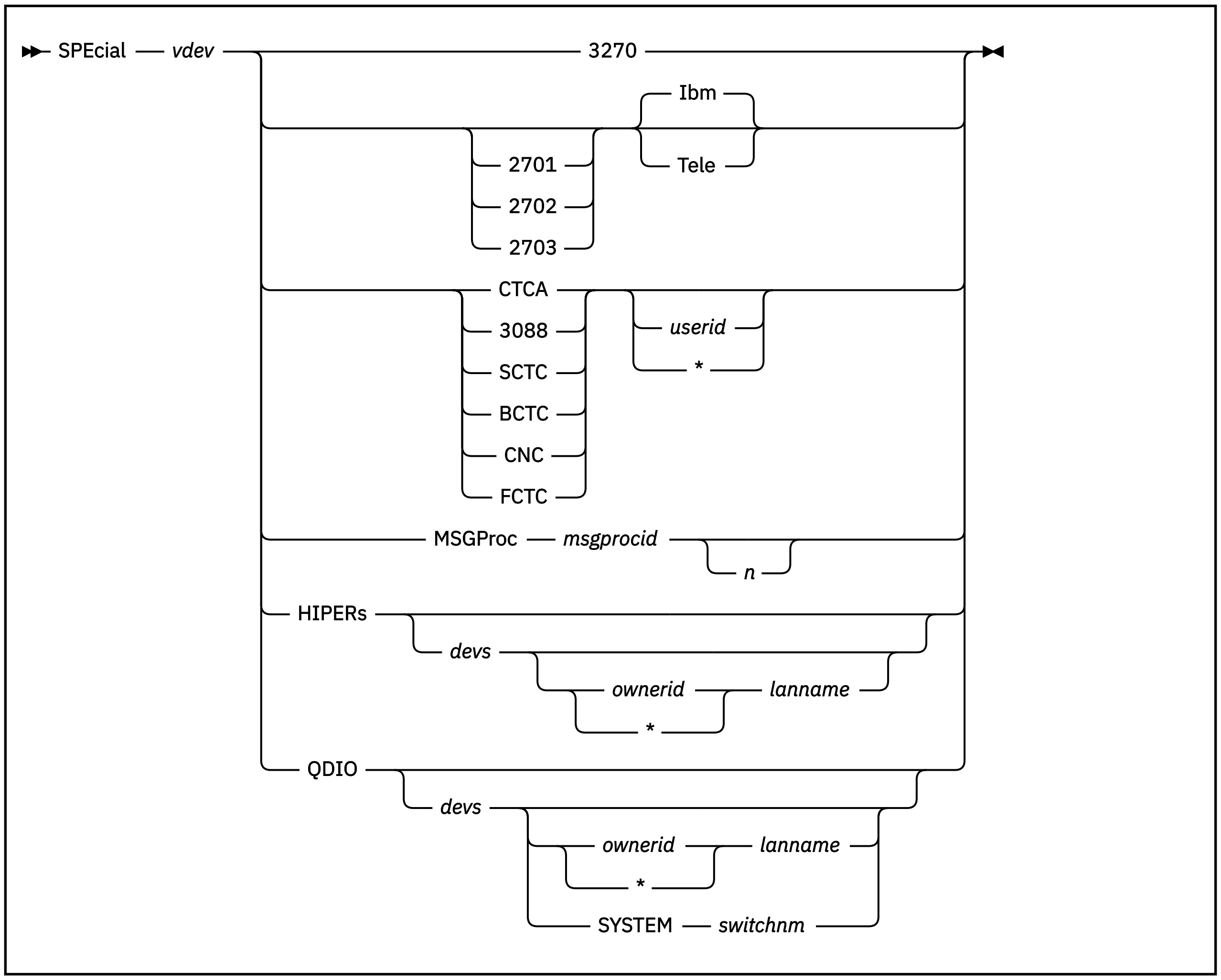
3. To assign a minimum absolute share of 50% and a maximum absolute share of 60%, use the following SHARE statement in the virtual machine definition:

   ```
   Share ABSOLUTE 50% ABSOLUTE 60%
   ```

4. To assign a minimum relative share of 100 and a maximum absolute share of 75%, where the maximum share is a hard limit, use the following SHARE statement in the virtual machine definition:

   ```
   Share RELATIVE 100 ABSOLUTE 75% LIMITHARD
   ```

SPECIAL Directory Statement



Purpose

The SPECIAL statement defines special virtual devices, which are fully simulated by CP and not connected with real devices at definition time. Some SPECIAL devices can be connected with a real device during normal operation. Refer to the DIAL command for 3270 and communication lines and to the COUPLE command for CTCA and 3088 devices. For more information, see DIAL and COUPLE in *z/VM: CP Commands and Utilities Reference*.

Use SPECIAL HIPER to create a virtual HiperSockets adapter in the virtual machine, and (optionally) connect it to a guest LAN. During LOGON processing, this statement is equivalent to a DEFINE NIC command followed by a COUPLE command.

How to Specify

The SPECIAL statement is allowed in profile, user, identity, and subconfiguration entries. If you specify the SPECIAL statement, it must follow any general statements you specify in a directory entry. (For a list of general statements, see Table 24 on page 462.)

Multiple SPECIAL statements are allowed in an entry if no duplicate virtual device numbers are specified in the entry. Any SPECIAL statements in a subconfiguration entry are processed first at logon time, followed by SPECIAL statements in a user or identity entry, followed by SPECIAL statements in a profile entry.

DIRECTXA does not check for extra tokens that might be specified at the end of the SPECIAL statement. If you specify additional tokens, DIRECTXA ignores the extra tokens and system processing continues as if you had not specified extra tokens.

Operands

vdev

For most **SPECIAL** devices, this is the virtual device address of the device to be defined. For a MSGPROC , HIPERS, or QDIO device, this represents the base (or first) device address in a series of virtual I/O devices that belong to the same unit.

A simulated adapter can also be created using the NICDEF directory statement. The NICDEF statement allows additional configuration options not available on the SPECIAL statement. For example, it may be necessary to have more control over which CHPID is used for a simulated QDIO NIC. For more information about how NICDEF interacts with real CHPIDs on the LPAR and how its CHPID parameter can make that interaction more predictable, see "Channel path usage" on page 569 (in the NICDEF statement description).

3270

is the value for a virtual 3270 display device.

CTCA
3088
SCTC
BCTC
CNC
FCTC

specifies a CTCA, 3088, SCTC, BCTC, CNC or FCTC for a virtual 3088 Multisystem Channel Communication Unit logical channel adapter. You can also specify:

userid

is the user ID of a virtual machine that is allowed to connect to this virtual CTCA, 3088, SCTC, BCTC, CNC or FCTC using the CP COUPLE command. For more information, see DIAL in *z/VM: CP Commands and Utilities Reference*.

*****

tells CP that a CP COUPLE command is to be allowed only from another virtual CTCA, 3088, SCTC, BCTC, CNC, or FCTC owned by the same virtual machine that owns the CTCA or 3088 defined by this SPECIAL statement.

If you do not specify a user ID or an asterisk (*), a virtual machine with any user ID can connect to this virtual CTCA.

2701
2702
2703

is the value for a communication line. This value is optional and used for compatibility only.

Ibm

is the virtual device type of the line you are defining. This is a 2741, 3767, or equivalent device. This is the default if it is used with the 270*x* operand.

Tele

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

MSGProc *msgprocid n*

restricts and defines a virtual message processor and associated message devices in the virtual I/O configuration. It creates a message facility environment for the user and establishes a connection to the specified Coupling Facility (CF) Service Machine supplying the message processor function.

If this is being used to define a CFLINK, then *n* must be between 2 and 8; the default is 2. If this is being used to define a MSGPROC, then *n* must be an even number between 4 and 16; the default is 4.

However, if the MSGPROC is not in z/Architecture mode, then the default value (2 or 4, as appropriate) is taken for *n*, no matter what is specified.

Defining a message processor by the SPECIAL MSGPROC directory statement allows the system administrator to restrict the message processors a user is allowed to define with the DEFINE MSGPROC command. Once a SPECIAL MSGPROC directory statement is specified in the virtual machine definition, the user can define message processors specified only by SPECIAL MSGPROC statements.

The virtual message processor will only be defined if the following conditions exist:

- OPTION CFUSER is specified in the virtual machine definition for this user
- the *vdev* specified is the first of four available consecutive device numbers in the users virtual configuration
- the specified CF Service Machine user ID is running prior to the logging on of this virtual machine.

Note: If the above conditions are not met, but OPTION CFUSER was specified in the user directory, the user will have to define the message processor with the DEFINE MSGPROC command after logging on. Since the SPECIAL statement was specified, the user will still be restricted to only defining the message processors specified in his directory.

The MSGPROC option requires OPTION CFUSER, which is not allowed in a subconfiguration entry. Therefore, specifying MSGPROC on a SPECIAL statement in a subconfiguration entry generates an error message.

HIPERs
indicates that a simulated HiperSockets adapter should be created based on this statement. A simulated network interface card (NIC) is defined during LOGON with *devs* devices (beginning with the base, *vdev*). If a guest LAN is identified, the NIC is automatically coupled to *ownerid lanname*.

QDIO
indicates that a simulated QDIO adapter should be created based on this statement. A simulated network interface card (NIC) is defined during LOGON with *devs* devices (beginning with the base device, *vdev*). If a guest LAN or virtual switch is identified, the NIC is automatically coupled to the specified *lanname* or *switchnm*.

devs
is the number (decimal) of virtual I/O devices to be created for a simulated network interface card (NIC). This number is evaluated during LOGON processing. For a simulated HiperSockets adapter, *devs* must be a decimal value between 3 and 3,072 (inclusive). For a simulated QDIO adapter, *devs* must be a decimal value between 3 and 240 (inclusive). The default for either HIPERS or QDIO is three (3) devices.

ownerid|* lanname
identifies a guest LAN for an immediate connection to the network interface card (NIC). When *ownerid* and *lanname* are omitted, the simulated adapter is left in the default (uncoupled) state. When *ownerid* and *lanname* are specified, the simulated adapter is automatically connected to the designated guest LAN. Note that *ownerid* can be specified as asterisk (*) to represent the user ID of the current virtual machine.

SYSTEM *switchnm*
identifies a virtual switch for an immediate connection to the network interface card (NIC). When **SYSTEM** *switchnm* is omitted, the simulated adapter is left in the default (uncoupled) state. When **SYSTEM** *switchnm* is specified, the simulated adapter is automatically connected to the designated virtual switch.

Usage Notes

1. When a simulated NIC is defined (**HIPERS** or **QDIO** device types), the **SPECIAL** statement results in the creation of a series of I/O devices. The base device is validated by directory processing, but the remaining devices in the range are validated during LOGON processing. If another device is found in the range established by *vdev* and *devs*, the simulated NIC cannot be created.

2. It is possible to define a simulated HiperSockets NIC or QDIO NIC that will be automatically coupled to a guest LAN (designated by *ownerid lanname*) or to define a simulated QDIO NIC that will be automatically coupled to a virtual switch (designated by SYSTEM *switchnm*). However, if the designated guest LAN or virtual switch is not available when this user signs on, the COUPLE cannot be performed. To make effective use of this feature, you should consider adding a DEFINE LAN or DEFINE VSWITCH statement in the SYSTEM CONFIG file to create the target guest LAN or virtual switch during system initialization.
3. The virtual channel-to-channel adapter defined will be equivalent to a System/370 CTCA designed after 1 January 1981. To determine which I/O commands a CTCA supports, refer to the *S/360 S/370 Channel-To-Channel Adapter* book.
4. When SPECIAL MSGP is processed, if the creation of the CFLINK is delayed (in case the target was not logged on yet or not ready to be the receiver of a CFLINK), the following message is displayed:

   HCP2821I SPECIAL MSGP processing was deferred

   Each time a coupling facility virtual machine (CFVM) finishes logging on, the existing CFVMs reevaluate their SPECIAL MSGPs to determine if any will nominate the new CFVM. Then each SPECIAL MSGP that matched is handled.
5. The SPECIAL HIPER statement generates multiple virtual I/O devices. If any other device exists in the range required for the *vdev* plus *devs* (minus 1), the HiperSockets adapter cannot be created.
6. When Directory Network Authorization is enabled, the NICDEF directory statement with LAN will be automatically authorized to connect to that network (in the absence of an ESM), but SPECIAL with a network ID is not automatically authorized to connect to the network. If you configure a virtual NIC with the SPECIAL directory control statement you will still have to authorize it via an ESM or by a SET VSWITCH / LAN command (or MODIFY VSWITCH / LAN statement in SYSTEM CONFIG).

Examples

1. Define a simulated QDIO adapter using I/O devices 0500–0507 (eight devices) which will be coupled to the SYSTEM-owned INEWS LAN during LOGON processing:

   ```
   SPECIAL 500 QDIO 8 SYSTEM INEWS
   ```

2. Define a simulated HiperSockets adapter using I/O devices FD20–FD2F (16 devices) which will be coupled to the user's own HSTEST LAN during LOGON processing:

   ```
   SPECIAL FD20 HIPERS 16 * HSTEST
   ```

   Note that this adapter cannot couple to the designated LAN during LOGON unless it has been defined earlier. This can be accomplished by adding the necessary **DEFINE LAN** statement to the SYSTEM CONFIG file.
3. To define a CTCA at virtual device number 2FF that can be coupled by LINKMAN, specify one of the following SPECIAL statements in the virtual machine definition:

   ```
     Special  2ff  Ctca  linkman
     Special  2ff  3088  linkman
   ```

4. To define a 3088 at virtual device number 3F0 that can be coupled to by any virtual machine, specify one of the following SPECIAL statements in the virtual machine definition:

   ```
     Special  3f0  3088
     Special  3f0  Ctca
   ```

5. To define virtual 3270 display devices at virtual device numbers 101, 102, and 103, specify the following SPECIAL statements in the virtual machine definition:

   ```
     Special  101  3270
     Special  102  3270
     Special  103  3270
   ```

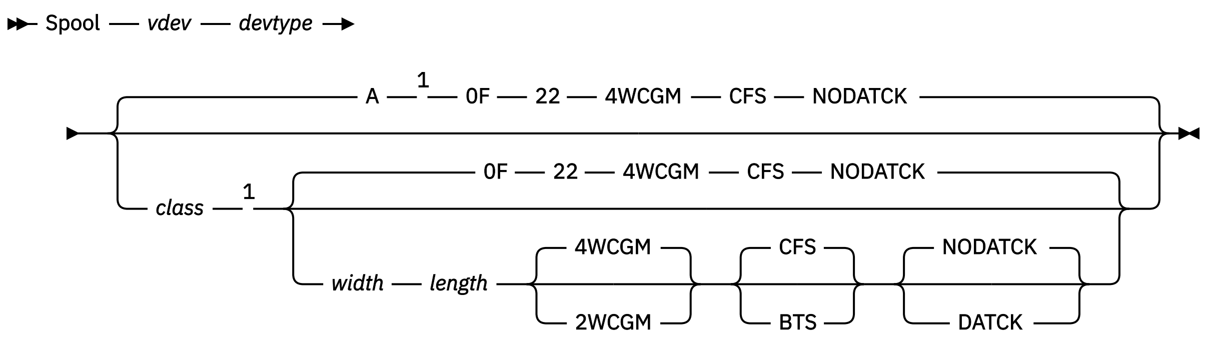
6. To define a virtual communication line at virtual device numbers 301, 302, and 303, specify the following SPECIAL statements in the virtual machine definition:

   ```
   Special  301  2702  Ibm
   Special  302  2702
   Special  303  Ibm
   ```

   Note: Although the statements differ in content, they all define the same type of line at their respective device numbers.
7. To define a virtual HiperSockets adapter with I/O devices 0500–0507 (eight devices) and connect it to the system-owned LAN named INEWS during LOGON processing, specify the following SPECIAL statement in the virtual machine definition:

   ```
   Special 500 hiper 8 system inews
   ```

SPOOL Directory Statement



Notes:

[1] The following variables are only recognized for a 3800 printer.

Purpose

The SPOOL statement defines virtual unit record devices.

How to Specify

The SPOOL statement is allowed in profile, user, identity, and subconfiguration entries. If you specify the SPOOL statement, it must follow any general statements you specify in a directory entry. (For a list of device statements, see Table 24 on page 462.)

Multiple SPOOL statements are allowed in an entry if no duplicate virtual device numbers are specified in the entry. Any virtual device number in a profile that duplicates a virtual device number in a user, identity, or subconfiguration entry is resolved at logon time. Any SPOOL statements in a subconfiguration entry are processed first at logon time, followed by SPOOL statements in a user or identity entry, followed by SPOOL statements in a profile entry.

When processing the SPOOL statement for 3800 printers, DIRECTXA checks for a maximum of eight tokens: virtual device number, device type, spooling class, width of the paper, length of the paper, number of writable characters, stacker type, and data check setting, in that order. If you specify more than eight tokens, DIRECTXA produces an error message and the directory will not be updated.

When processing the SPOOL statement for all other devices, DIRECTXA checks for a maximum of three tokens: virtual device number, device type, and volume serial number. If you specify more than three tokens, DIRECTXA ignores the extra tokens and system processing continues as if you had not specified the extra tokens.

Operands

vdev
is the virtual device number for the spooling device.

devtype
is the device type. Valid device types are:

1403	3262	3800-3 (See note 4)	Printer (See note 5)
2501	3505	4245	PRT (See note 5)
2540 (See note 1)	3525	4248	RDR (See note 7)
3203	3800 (See note 2)	PCH (See note 6)	Reader (See note 7)

3211	3800-1 (See note 3)	PUnch (See note 6)	VAFP (See note 8)

Notes:

1. Specify 2540 for a reader or a punch.
2. 3800 defaults to a 3800 Model 1 printer.
3. Specify 3800-1 for a 3800 Model 1 printer.
4. Specify 3800-3 for a 3800 Model 3 printer.
5. Printer or PRT defaults to a 1403 printer.
6. PCH or PUnch defaults to a 2540 punch.
7. RDR or Reader defaults to a 2540 reader.
8. Specify VAFP for a virtual advanced functional printer. For more information, see DEFINE (Spooling device) in *z/VM: CP Commands and Utilities Reference*.

class

is the spooling class. If omitted, A is used as a default. The variable *class* is a 1-digit alphanumeric character from A to Z, from 0 to 9, or an asterisk (*). A reader is the only valid device that can use an asterisk (*).

For spooled output devices, the class governs the punching or printing of the real spooled output. For spooled input devices, the class controls access to spool files by virtual card readers.

width length

specifies the physical characteristics of the paper to be loaded into the 3800 printer. The variable *width* is the hexadecimal form width code of the paper, and *length* indicates the decimal length of the paper. Specify *length* as a decimal number using *half-inches*. If *width* and *length* are omitted, 14-7/8 x 11 inches is used as a default.

The following table lists available form-width codes (values other than those listed below are rejected):

Code	Width in Inches	Width in Millimeters
01	6-1/2	165
02	Reserved	180
04	8-1/2	215
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

2WCGM
4WCGM

specifies the number of writable character generation modules (WCGM) for the virtual 3800 printer. A WCGM is a 64-position portion of the 3800's character generation storage that holds the scan elements of one character set. A 3800-1 can have either two or four WCGMs. A 3800-3 has four WCGMs. If omitted, the default is 4WCGM.

BTS
CFS
specifies the type of stacker for the virtual 3800 printer. You may specify either CFS (continuous forms stacker) or BTS (burster trimmer stacker). If omitted, the default is CFS.

DATCK
NODATCK
specifies whether CP processes certain virtual 3800 data checks for the virtual machine. If you specify DATCK, CP reflects all data checks to the virtual machine (if the Block Data Check CCW is not issued). If you specify NODATCK, only data checks due to invalid translate table specifications or unmatched FCB codes are reflected to the virtual machine. If omitted, the default is NODATCK.

Note: Specifying DATCK 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. Therefore, specify DATCK only when absolutely necessary.

Examples

1. To define at 00C a spooled reader that can read any class of spool file, specify the following SPOOL statement in the virtual machine definition:

   ```
   Spool 00c Reader *
   ```

2. To define at 00D a spooled punch that creates only class P spool files, specify the following SPOOL statement in the virtual machine definition:

   ```
   Spool 00d Punch p
   ```

3. To define at 00E a spooled 1403 printer that processes class E spool files, specify the following SPOOL statement in the virtual machine definition:

   ```
   Spool 00e 1403 e
   ```

4. To define at 00E a virtual 3800 printer with these characteristics:

   - Only class Z spool files
   - 11 X 11 inch paper
   - Two WCGMs
   - The continuous forms stacker
   - No CP reflection of data checks

   specify the following SPOOL statement in the virtual machine definition:

   ```
   Spool 00e 3800 z 09 22 2wcgm Cfs NoDatCk
   ```

SPOOLFILE Directory Statement

```
►► SPOOLFile ── MAXSPOOL ── nnnn ►◄
```

Purpose

The SPOOLFILE statement describes virtual machine spool file characteristics. Use the MAXSPOOL *nnnn* operand to specify the maximum number of spool files allowed for a virtual machine.

How to Specify

The SPOOLFILE statement is allowed in profile, user, identity, and subconfiguration entries. If specified, it must precede any device statements in the entry. (For a list of device statements, see Table 24 on page 462.) If the SPOOLFILE statement is omitted, the default maximum number of spool files is 9999.

One SPOOLFILE statement is allowed in an entry. A SPOOLFILE statement in a user or identity entry overrides one in a profile entry. A SPOOLFILE statement in a subconfiguration entry overrides one in an identity entry.

When processing the SPOOLFILE statement, DIRECTXA checks for a maximum of two tokens: the MAXSPOOL keyword and the decimal number of spool files, in that order. If you specify more than two tokens, DIRECTXA ignores the extra tokens and system processing continues as if you had not specified the extra tokens.

Operands

MAXSPOOL *nnnn*
specifies the maximum number of spool files the user can have at one time. The variable *nnnn* is a decimal number from 1 to 9999. The spool file ID numbers for this user will be in this range.

Usage Notes

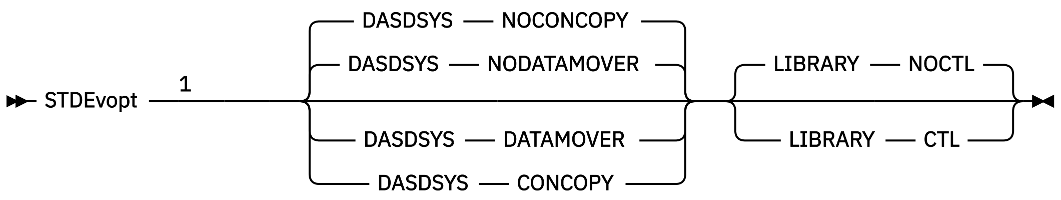
1. If a virtual machine's maximum is reset to a value below currently allocated spool IDs, those files and spool IDs remain valid. No additional files will be allocated above the new maximum, and when an old file is deleted, that spool ID will no longer be available.
2. In an SSI cluster, a single-configuration virtual machine can own up to the MAXSPOOL number of spool files, but only a fraction of the MAXSPOOL number of spool files can be created on any one particular member of the cluster. The fraction depends on the setting of the SPOOL_MEMBERS operand of the SSI_CONTROLS statement in the system configuration file. If this operand is set to 4, the fraction is one-fourth. If this operand is set to 8, the fraction is one-eighth. A multiconfiguration virtual machine can own up to the MAXSPOOL number of spool files on each member of the cluster, but each instance of the virtual machine can view only the spool files originated on the member where the instance is logged on. For more information, see “Spool File ID Assignment and Limits” on page 751.

Examples

To indicate that the virtual machine has a limit of 1000 spool files, specify the following SPOOLFILE statement in the virtual machine definition:

```
SpoolFile MaxSpool 1000
```

STDEVOPT Directory Statement



Notes:

[1] You must specify at least one of the operands. If you want to specify both, you can specify them in any order.

Purpose

The STDEVOPT statement specifies the optional storage device management functions available to a virtual machine.

How to Specify

The STDEVOPT statement is allowed in profile, user, and identity entries. If you specify the STDEVOPT statement, it must precede any device statements you specify in the user, identity, or profile entries. For a list of device statements, see Table 24 on page 462.

Multiple STDEVOPT statements are allowed within an entry. Conflicting operands within an entry are flagged as errors.

Operands

DASDSYS
tells CP whether the virtual machine is authorized to control and process Concurrent Copy and peer-to-peer remote copy CCWs.

NODATAMOVER
(the default) tells CP that the virtual machine is not authorized for Concurrent Copy or peer-to-peer remote copy.

NOCONCOPY
same as NODATAMOVER (left here to be compatible with previous releases of VM).

DATAMOVER
tells CP that the virtual machine is authorized to issue Concurrent Copy and peer-to-peer remote copy CCWs.

CONCOPY
same as DATAMOVER (left here to be compatible with previous releases of VM).

LIBRARY
tells CP whether the virtual machine is authorized to control a 3494 or 3495 Tape Library Dataserver.

NOCTL
(the default) tells CP that the virtual machine is not authorized to control a tape library.

CTL
tells CP that the virtual machine is authorized to issue tape library control commands.

Usage Notes

1. The status of a storage device management function not explicitly specified in the user or identity entry defaults to nonauthorization (NODATAMOVER or NOCTL), unless the authorization (DATAMOVER or CTL) is defined in a profile. An authorization or nonauthorization explicitly specified in the user or identity entry supersedes the authorization status defined in a profile.
2. With LIBRARY CTL authorization, tape library control commands can be used to access any volume in the library. To ensure integrity of the volumes in the library, LIBRARY CTL authorization should be granted only to virtual machines that control the mounting and demounting of volumes, such as a guest virtual machine for an operating system, or a service virtual machine for a tape library control program that provides automated mounting and demounting functions.
3. LIBRARY CTL should not be used to authorize a virtual machine to access a particular tape library. LIBRARY CTL allows a virtual machine to issue tape library commands to any tape device, regardless of the library in which the device resides.
4. CONCOPY and NOCONCOPY are still accepted for compatibility reasons. They provide the same authorization control as DATAMOVER and NODATAMOVER, respectively. That is, specifying CONCOPY will authorize a virtual machine for both Concurrent Copy commands and peer-to-peer remote copy commands. Specifying NOCONCOPY will disallow all Concurrent Copy commands and peer-to-peer remote copy commands.
5. Only one DASDSYS option (DATAMOVER, NODATAMOVER, CONCOPY, NOCONCOPY) is allowed in a user or profile directory. Specifying more than one option will result in error message 776E being sent to the user console and the directory will not get updated.

Examples

1. To specify that a guest virtual machine can initiate and control Concurrent Copy and peer-to-peer remote copy, specify the following STDEVOPT statement in the virtual machine definition:

   ```
   STDEvopt DASDSYS DATAMOVER
   ```

2. To specify that a virtual machine can control a 3494 or 3495 Tape Library Dataserver, specify the following STDEVOPT statement in the virtual machine definition:

   ```
   STDEvopt LIBRARY CTL
   ```

3. To specify that a guest virtual machine can control Concurrent Copy sessions and peer-to-peer remote copy, and can control a tape library, specify the following STDEVOPT statement in the virtual machine definition:

   ```
   STDEvopt LIBRARY CTL DASDSYS DATAMOVER
   ```

STORAGE Directory Statement

```
►►─ STORAGE ── size ─►◄
```

Purpose

The STORAGE statement specifies the default (logon) virtual storage size for a user.

How to Specify

The STORAGE statement is allowed in profile, user, identity, and subconfiguration entries. The STORAGE statement, if used, must precede any device statements. (For a list of device statements, see Table 24 on page 462.) A STORAGE statement is allowed in a profile entry if the USER or IDENTITY statement default storage size field of each including virtual machine is either omitted or specified as an asterisk (*). A STORAGE statement in a user or identity entry overrides one in a profile entry.

If STORAGE or MAXSTORAGE statements are specified in the subconfiguration entry, then those statements override any in the identity entry. If either a STORAGE or MAXSTORAGE statement is specified, but not the other, within a subconfiguration entry, then the default is applied to the other in the subconfiguration entry. The combination of the specified value and the default then override whatever settings were specified in the identity entry.

Operands

size

is the default (logon) storage size of the virtual machine. Specify *size* as *nu*, where *n* is a 1- to 7-digit decimal number and *u* is the 1-character storage unit suffix (see Table 35 on page 608).

Table 35. Maximum Input Values for Storage Units

Storage unit suffix (*u*)	**Maximum input value (*n*)**
K - kilobytes	9999999
M - megabytes	9999999
G - gigabytes	9999999
T - terabytes	9999999
P - petabytes	16384
E - exabytes	16

Notes:

1. A K specification is rounded up to a MB value.
2. The maximum input value of 9999999 for the K, M, G, or T suffix is not a size limit but the physical limit of the operand (7 digits plus suffix). If the maximum input value for one of these suffixes does not allow you to define the amount of storage you want, you need to use a larger storage unit.
3. The maximum size you can specify is 16E or 16384P, although the actual maximum size supported may be restricted by the model of the server where the directory is used.
4. An XC virtual machine can address up to 2047 MB of storage in its base address space.
5. Allowing many virtual machines to have large storage sizes might affect real storage availability. See usage note "2" on page 609.

Usage Notes

1. A default storage setting on a USER or IDENTITY statement overrides a STORAGE statement in a profile.
2. Allowing many virtual machines to have large storage sizes might affect real storage availability. For each virtual machine, CP creates dynamic address translation (DAT) tables to reference the virtual machine storage. DAT tables include page tables, segment tables, and higher level (region) tables.

   CP keeps the page tables in page management blocks (PGMBKs). Each 8 KB PGMBK references 1 MB of virtual machine storage. PGMBKs might be pageable; as such, their impact on real storage depends on how frequently the MBs of storage they reference are used.

   Segment tables and region tables are allocated from host real storage and are not pageable:

   - To reference the page tables for a primary address space or data space up to 2 GB, 1 - 4 contiguous frames are allocated for the segment table, one frame for each 512 MB of storage.
   - For a primary address space larger than 2 GB, multiple segment tables are created, plus one or more region tables to reference the segment tables. Each region table occupies 1 - 4 contiguous frames. If needed, multiple levels of region tables are created.

SUBCONFIG Directory Statement

```
►► SUBCONFIG — id ►◄
```

Purpose

The SUBCONFIG statement starts a subconfiguration entry. This entry contains a set of directory statements in a multiconfiguration virtual machine definition that are specific to one of its virtual machine instances within the SSI cluster.

How to Specify

No additional operands are allowed after the *id*. If you specify extra operands, DIRECTXA produces an error message and the directory will not be updated.

Operands

id

defines the virtual machine's 1- to 8-character SUBCONFIG ID. LOGN*xxxx*, LOGL*xxxx*, LOGV*xxxx*, LOGNSYSC, LOGNSYSG, SYSTEM, and SYSTEMMP are reserved for CP use.

The SUBCONFIG ID cannot be the same as a user ID specified on an IDENTITY statement or USER statement.

You should not assign SUBCONFIG IDs that are system keywords (such as command names, command operands, or 1- to 4-digit user IDs that could be spool IDs). Assigning system keywords as IDs can cause unpredictable results. For a list of restricted user IDs, see Restricted User IDs in *z/VM: CP Commands and Utilities Reference*.

You should not assign SUBCONFIG IDs that contain colons (:), periods (.), semicolons (;), or slashes (/). These characters are used, or might be used in the future, as delimiters in several CP commands that also take a user ID as a parameter. Results are unpredictable when these commands are entered with a SUBCONFIG ID containing any of these delimiter characters.

Translation test tables in HCPTBL allow only the following characters in SUBCONFIG IDs:

alphabetics
A-Z

numerics
0-9

The following, comma-delimited characters are also allowed:
@, #, $, _ (underscore), - (hyphen)

Note: It is recommended that the characters @, #, ¢, and " not be used in the *id* because the system, by default, assigns these characters as logical line editing symbols. For more information, see Logical Line Editing Symbols in *z/VM: CP Commands and Utilities Reference*.

You can override the translation tables in HCPTBL. For more information, see "TRANSLATE_TABLE Statement" on page 304.

Usage Notes

1. The SUBCONFIG statement is used in combination with the IDENTITY and BUILD statements to specify a multiconfiguration virtual machine definition. For more information, see "IDENTITY Directory Statement" on page 510 and "BUILD Directory Statement" on page 480.

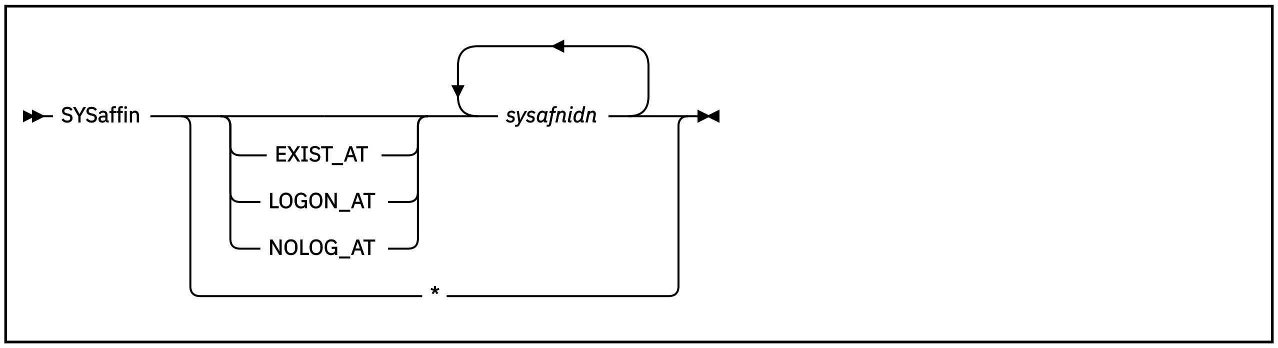
2. If duplicate specifications exist in an identity entry and its subconfiguration entry, the specification in the subconfiguration entry overrides the one in the identity entry. Refer to the individual statement descriptions for specific override rules.
3. The ID on a SUBCONFIG entry must match the ID specified on a BUILD statement. SUBCONFIG IDs that are not referenced by BUILD statements are not allowed.
4. A null SUBCONFIG definition is allowed. This enables a multiconfiguration virtual machine instance to be logged on to different members in the SSI cluster simultaneously by using just the statements in the identity entry.
5. The following statements are not allowed within a subconfiguration entry:
   - ACCOUNT
   - ACIGROUP
   - APPCPASS
   - AUTOLOG
   - BUILD
   - CLASS
   - CPU
   - D8ONECMD
   - INCLUDE
   - IOPRIORITY
   - IUCV
   - LOGONBY
   - MACHINE
   - NAMESAVE
   - NOPDATA
   - POOL
   - POSIXGLIST
   - POSIXINFO
   - POSIXOPT
   - STDEVOPT
   - XAUTOLOG
6. The following OPTIONs are not allowed within a SUBCONFIG definition:
   - ACCT
   - APPLMON
   - CFUSER
   - CFVM
   - CHPIDVirtualization
   - COMSRV
   - CRYMEASURE
   - DEVINFO
   - DEVMAINT
   - DIAG88
   - DIAG98
   - D84NOPASS
   - IGNMAXU

- LKFAC
- LNKEXCL
- LNKNOPAS
- LNKSTBL
- MAINTCCW
- NETACCOUNTING
- NETROUTER
- RMCHINFO
- SETORIG
- STGEXEMPT
- SVMSTAT

Examples

1. See examples in "IDENTITY Directory Statement" on page 510.

SYSAFFIN Directory Statement



Purpose

The SYSAFFIN statement is an optional statement that can be used to define how and to which systems of a multisystem complex the subsequent statements apply.

SYSAFFIN specifications are allowed only in a non-SSI source directory. If SYSAFFIN specifications are found in an SSI-ready or SSI-enabled source directory, then an error results.

How to Specify

A SYSAFFIN statement has two forms, prefix and internal. The prefix form is placed before user entries to describe where or how the user is to be applied to the systems of the complex. The internal form can be used to define which statements apply to which systems in the complex. Only the internal form of the SYSAFFIN statement is allowed within a global definition or profile entry.

Each SYSAFFIN prefix statement applies only to the USER statement that immediately follows it.

A SYSAFFIN internal statement applies to the statements that immediately follow it. It is active until the next SYSAFFIN prefix or internal statement, or until the next PROFILE or USER statement, whichever comes first.

Note: The system affinity ID, *sysafnid*, that is used in the following statements is a 1- to 8-character alphanumeric value assigned to a system and used on a DIRECTORY statement in the beginning of the source directory. The SYSAFFIN statement, and the statements to which it applies, affect the system identified by *sysafnid*.

Operands

EXIST_AT *sysafnidn*
: specifies that the following user entry is to be compiled into the object directory by the DIRECTXA utility only when running on a system with one of the system affinity IDs (*sysafnidn*) listed on the SYSAFFIN statement. When DIRECTXA is running on any other system, this user entry is not to be compiled into the object directory. This means that any resources owned by this user entry (for example, minidisks) do not appear in the object directory of the other systems and, therefore, cannot be linked to by users on those systems.

LOGON_AT *sysafnidn*
: specifies that the virtual machine definition is to be compiled into the object directories of all systems, but the associated virtual machine is to operate (for example, LOGON) only on specified systems. On any other system, the logon password is replaced with the keyword NOLOG. However, the virtual machine with this user ID and all its resources, including minidisks, exist in the object directories of the other systems and can be referenced by users on those systems.

NOLOG_AT *sysafnidn*
: specifies that the specified virtual machine definition is to be compiled into the object directories of all systems, but the associated virtual machine is not to be operable on the identified systems

(*sysafnidn*). However, the virtual machine and all its resources, including minidisks, are to exist as virtual machine definitions in the object directories of these systems and can be referred to by users on those systems.

*****
sysafnidn
is a 1- to 8-character alphanumeric string that identifies the system that the subsequent statements apply to. Valid *sysafnidn*s are defined on the DIRECTORY statement. * indicates that the subsequent statements apply to all systems. A maximum of 56 system affinity ids can be specified.

Usage Notes

1. The default system affinity for user entries not otherwise prefixed by a SYSAFFIN statement is LOGON_AT *. This means that the user entry is applied to and that the defined user can log on at all systems in the complex. The default system affinity for all other statements is *, meaning that all subsequent statements apply to all systems in the complex.
2. In a system that uses the LOGON_AT or NOLOG_AT operands of the SYSAFFIN statement, the class B virtual machine that does the DIAGNOSE code X'84' operations should have D84NOPAS specified on the OPTION statement.
3. Except for the logon status of the virtual machine (as defined by the use of a SYSAFFIN prefix statement), the data on the USER statement is common to all systems in the complex where the user is defined to exist.
4. The internal form of the SYSAFFIN statement can be used in user entries, profile entries, and the global definition entries. These statements define which portions of the various definitions apply to which systems in the complex.

Examples

In the examples that follow, assume this multisystem complex definition:

```
Directory 07a4 3390 vmaipl  *01234-3906 vma
Directory 0b64 3390 vmbipl  104567-3906 vmb
Directory 0cF4 3390 vmcipl  *15211-3906 vmc
Directory 0664 3390 vmdipl  *00012-3906 vmd
```

1. The PROFILE DEPT1234 definition will be available on all systems. Profiles cannot use the prefix form of the SYSAFFIN statement. They can, however, be tailored using the internal form:

   ```
   Profile dept1234
   SysAffin vma
   Account 00000001 00000002
   SysAffin vmb
   Account 00000003 00000004
   SysAffin *
   Link deptdb 100 100 rr
   Link deptdb 101 101 rr
   SysAffin vma
   Link deptdb 200 200 rr
   SysAffin vmb
   Link deptdb 201 200 rr
   ```

2. Defined below are users of DEPTDB with different system affinity situations.

   - This is the department database, with common disks 100 and 101 and unique disks 200 and 201, depending on whether the system is VMA or VMB:

     ```
     SysAffin Exist_At vma vmb
     User deptdb …
     Mdisk 100 …
     Mdisk 101 …
     SysAffin vma
     Mdisk 200 …
     SysAffin vmb
     Mdisk 201 …
     ```

- User TOM has access to all systems and through the INCLUDE statement has access to the department database. Depending on whether he is logged on to VMA or VMB, he will have linked a different disk as his 200. He will also have an appropriate set of R/W minidisks for each system.

  ```
  User tom …
  Include dept1234
  Account 12340001
  SysAffin vma
  Mdisk 191 …
  Mdisk 193 …
  SysAffin vmb
  Mdisk 191 …
  Mdisk 193 …
  ```

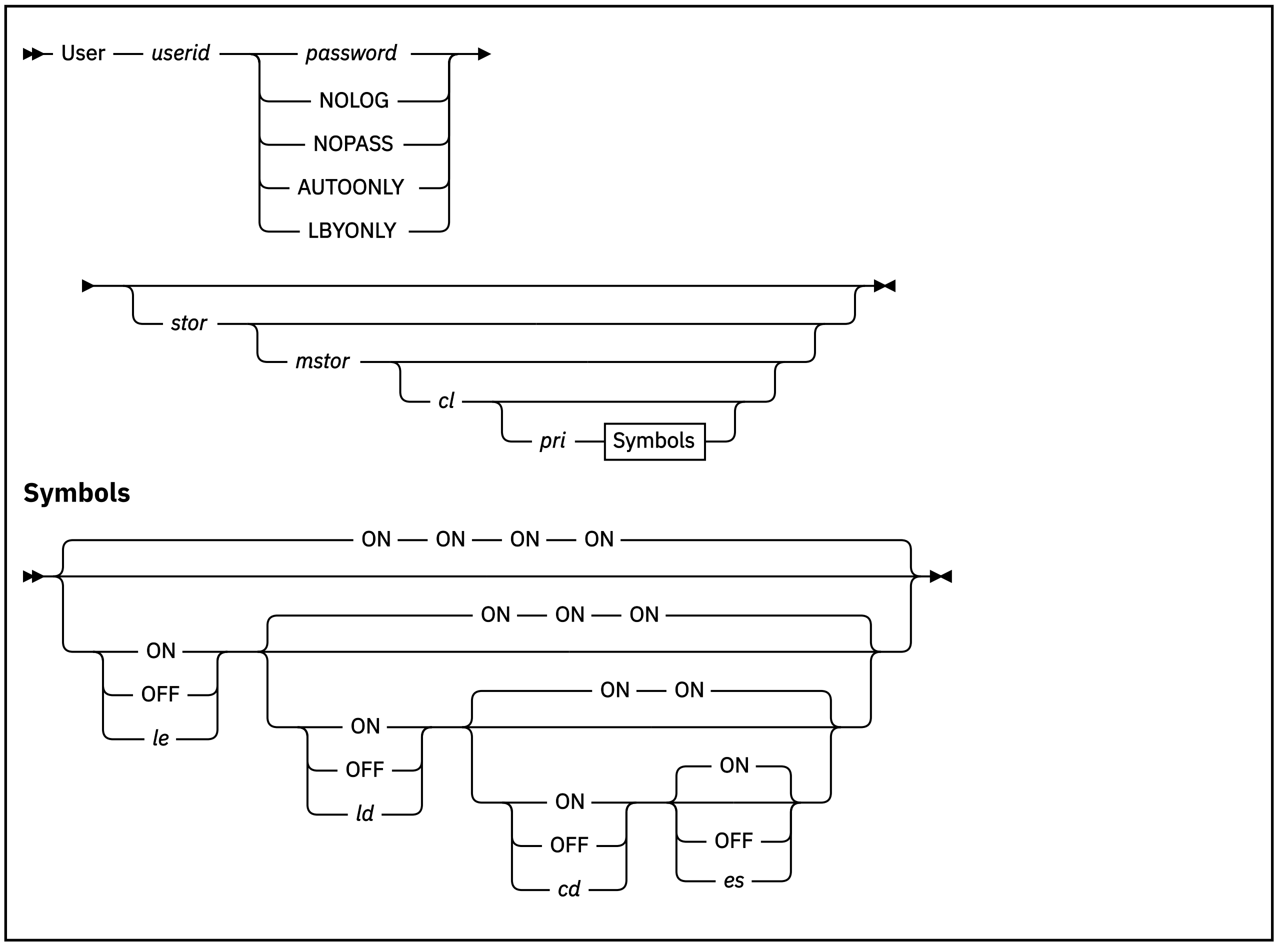
- User MARY is limited to VMA access and will have access to the department database on VMA only.

  ```
  SysAffin Logon_At vma
  User mary …
  Include dept1234
  Account 12340002
  Mdisk 191 …
  Mdisk 193 …
  ```

3. The following global definition entry would define a default machine mode of XA on system vma and a default machine mode of ESA on system vmb.

   ```
   Globaldefs
      Sysaffin vma
      Globalopts machine xa
      Sysaffin vmb
      Globalopts machine esa
   ```

USER Directory Statement



Purpose

The USER statement starts a single-configuration virtual machine definition. The statement also defines a user ID and password, a logon and maximum virtual machine storage size, command privileges, and special symbol usage for the virtual machine.

How to Specify

When processing the USER statement, DIRECTXA checks for a maximum of ten tokens: userid, password, logon storage size, maximum storage size, privilege class, a placeholder for compatibility with VM/SP, line-end symbol, line-delete symbol, character-delete symbol, and escape character, in that order. If you specify more than ten tokens, DIRECTXA ignores the extra tokens and system processing continues as if you had not specified the extra tokens.

Operands

userid

defines the virtual machine's 1- to 8-character user ID. LOGN*xxxx*, LOGL*xxxx*, LOGV*xxxx*, SYSTEM, and SYSTEMMP are reserved for CP use. The *xxxx* can be any character, number, or symbol.

The user ID cannot be the same as a user ID specified on an IDENTITY statement or an ID specified on a SUBCONFIG statement.

You should not assign user IDs that are system keywords (such as command names, command operands, or 1- to 4-digit user IDs that could be spool IDs). Assigning system keywords as user IDs can cause unpredictable results. For a list of restricted user IDs, see Restricted User IDs in *z/VM: CP Commands and Utilities Reference*.

You should not assign user IDs that contain colons (:), periods (.), semicolons (;), or slashes (/). These characters are used, or can be used in the future, as delimiters in several CP commands that also take a user ID as a parameter. Results are unpredictable when these commands are entered with a user ID containing any of these delimiter characters.

Ttranslation test tables in HCPTBL allow only the following characters in user IDs:

alphabetics
A-Z

numerics
0-9

The following, comma-delimited characters are also allowed:
@, #, $, _ (underscore), - (hyphen)

Note: It is recommended that the characters @, #, ¢, and " not be used in the *userid* because the system, by default, assigns these characters as logical line editing symbols. For more information, see Logical Line Editing Symbols in *z/VM: CP Commands and Utilities Reference*.

You can override the translation tables in HCPTBL. See the "TRANSLATE_TABLE Statement" on page 304.

password
specifies a 1- to 8-character password that a user enters during the logon procedure.

Note: If DELETEUSER is specified as the password of a user when the DELTA option of the DIRECTXA utility is in effect, the user will be marked for deletion in its user index entry. For more information, see DIRECTXA in *z/VM: CP Commands and Utilities Reference*.

NOLOG
specifies that a user cannot log on. You can use the NOLOG option to create virtual machines to which no one can log on. For example, you can place special system DASD areas (allocation, warm-start, and checkpoint areas) on minidisks and then assign the minidisks to virtual machines with NOLOG passwords. This helps you identify areas on a DASD.

Also, you can create minidisks for common user data and then assign the minidisks to virtual machines with NOLOG passwords. Users can then enter a CP LINK command to access those minidisks when they need the data.

Note: It is recommended that the characters @, #, ¢, and " not be used in the *userid* because the system, by default, assigns these characters as logical line editing symbols. For more information, see Logical Line Editing Symbols in *z/VM: CP Commands and Utilities Reference*.

NOPASS
specifies that a user does not require a password to log on. It also specifies that a virtual machine can be automatically logged on using the XAUTOLOG command without password authorization.

Even though a user ID is defined with the NOPASS operand, LOGON password authorization might be required when an External Security Manager is installed. For more information, refer to documentation provided by your External Security Manager.

AUTOONLY
specifies that a user can be autologged, but not logged on at a terminal.

LBYONLY
specifies that:

- Logging on to this virtual machine with the LOGON command requires use of the BY option.
- This user ID cannot be used to log on to any virtual machine with the BY option.

Furthermore, this user ID might not be able to perform functions that require password validation, because LBYONLY is not accepted as a valid password.

CP allows an external security manager (ESM) to override these restrictions, so they are effective only when no ESM is installed or when the ESM defers to CP's processing. (See the ESM documentation for more information.)

For more information, see LOGONBY Directory Statement.

stor

specifies the virtual machine's primary address space size at logon. Specify *stor* as *nu*, where *n* is a 1- to 7-digit decimal number and *u* is the 1-character storage unit suffix (see Table 36 on page 618).

This operand can contain an asterisk (*) as a placeholder. If the operand is omitted or specified as an asterisk, this indicates that the storage size is defined by a STORAGE statement in either the user entry or a profile entry.

Table 36. Maximum Input Values for Storage Units

Storage unit suffix (*u*)	**Maximum input value (*n*)**
K - kilobytes	9999999
M - megabytes	9999999
G - gigabytes	9999999
T - terabytes	9999999
P - petabytes	16384
E - exabytes	16

Notes:

1. A K specification is rounded up to a MB value.
2. The maximum input value of 9999999 for the K, M, G, or T suffix is not a size limit but the physical limit of the operand (7 digits plus suffix). If the maximum input value for one of these suffixes does not allow you to define the amount of storage you want, you need to use a larger storage unit.
3. The maximum size you can specify is 16E or 16384P, although the actual maximum size supported might be restricted by the model of the server where the directory is used.
4. An XC virtual machine can address up to 2047 MB of storage in its base address space.
5. Allowing many virtual machines to have large storage sizes might affect real storage availability. See usage note "1" on page 620.

mstor

specifies the maximum size of the virtual machine's primary address space. Specify *mstor* as *nu*, where *n* is a 1- to 7-digit decimal number and *u* is the 1-character storage unit suffix (see Table 36 on page 618). The default is 1M (1 MB).

This operand can contain an asterisk (*) as a placeholder. If the operand is omitted or specified as an asterisk, this indicates that the maximum storage size is defined by a MAXSTORAGE statement in either the user entry or a profile entry.

cl

specifies the privilege class or classes of CP commands a user can enter. Command classes are A through Z, and 1 through 6. You can specify up to 32 classes (if they fit). Each character represents a single privilege class and can appear in any order, without duplication, and must not be separated by blanks. The class field can also contain an asterisk (*) as a placeholder. If the field is not specified, or if it contains an asterisk (*) and there is no CLASS statement, CP uses the default privilege class or classes that are assigned by the PRIV_CLASSES statement in the system configuration file. For more information, see "PRIV_CLASSES Statement" on page 223.

Note: If you have not changed your classes from the IBM-defined defaults, only classes A through G are valid. For information on defining your own user class structure, see Chapter 19, "Redefining Command Privilege Classes," on page 449. For a description of the different command classes, see Privilege Classes. For a description of the different command classes, see Privilege Classes in *z/VM: CP Commands and Utilities Reference*.

pri

is for VM/SP compatibility and serves no z/VM function. It must be a number from 0 to 99. If the *pri* specification is not entered, the line-end (*le*), line-delete (*ld*), character-delete (*cd*), and escape (*es*) characters default to ON.

le
ON
OFF

le is a 1-character line-end symbol or a 2-digit hexadecimal equivalent of the symbol. Input following a line-end symbol begins a new logical line. ON means that the virtual machine uses the system value. OFF means that CP will not recognize line-end symbols. If omitted, the default is ON.

Note: If you want to set a system-wide default for the line-end symbol, use the CHARACTER_DEFAULTS statement in the system configuration file. For more information, see "CHARACTER_DEFAULTS Statement" on page 67.

ld
ON
OFF

ld is a 1-character line-delete symbol or a 2-digit hexadecimal equivalent of the symbol. A line-delete symbol causes the previous logical line input to be ignored. ON means that the virtual machine uses the system value. OFF means that CP will not recognize line-delete symbols. If omitted, the default is ON.

Note: If you want to set a system-wide default for the line-delete symbol, use the CHARACTER_DEFAULTS statement in the system configuration file. For more information, see "CHARACTER_DEFAULTS Statement" on page 67.

cd
ON
OFF

cd is a 1-character delete symbol or a 2-digit hexadecimal equivalent of the symbol. A character-delete symbol causes the previous character input to be ignored. ON means the virtual machine uses the system value. OFF means that CP will not recognize character-delete symbols. If omitted, the default is ON.

Note: If you want to set a system-wide default for the character-delete symbol, use the CHARACTER_DEFAULTS statement in the system configuration file. For more information, see "CHARACTER_DEFAULTS Statement" on page 67.

es
ON
OFF

es is a 1-character escape symbol or a 2-digit hexadecimal equivalent of the symbol. The escape symbol causes CP to treat the following character literally, without consideration as an *le*, *ld*, or *cd* character. ON means that the virtual machine uses the system value. OFF means that CP will not recognize escape symbols. If omitted, the default is ON.

Note: If you want to set a system-wide default for the escape symbol, use the CHARACTER_DEFAULTS statement in the system configuration file. For more information, see "CHARACTER_DEFAULTS Statement" on page 67.

Usage Notes

1. Allowing many virtual machines to have large storage sizes might affect real storage availability. For each virtual machine, CP creates dynamic address translation (DAT) tables to reference the virtual machine storage. DAT tables include page tables, segment tables, and higher level (region) tables.

   CP keeps the page tables in page management blocks (PGMBKs). Each 8 KB PGMBK references 1 MB of virtual machine storage. PGMBKs might be pageable; as such, their impact on real storage depends on how frequently the MBs of storage they reference are used.

   Segment tables and region tables are allocated from host real storage and are not pageable:

   - To reference the page tables for a primary address space or data space up to 2 GB, 1 - 4 contiguous frames are allocated for the segment table, one frame for each 512 MB of storage.
   - For a primary address space larger than 2 GB, multiple segment tables are created, plus one or more region tables to reference the segment tables. Each region table occupies 1 - 4 contiguous frames. If needed, multiple levels of region tables are created.
2. The *cl* field can contain an asterisk (*) as a placeholder. If no further options are to be specified on the USER statement, the asterisk (*) is not required. In either case, if the CLASS statement is not supplied, CP uses the default class or classes assigned by the PRIV_CLASSES statement in the system configuration file. For more information, see "PRIV_CLASSES Statement" on page 223.
3. If any of the options are specified, all prior options must also be specified. The default for unspecified options is ON.
4. The USER statement defines the special symbol usage for a specific virtual machine. To override the special symbol usage for a specific virtual machine, have the user issue the CP TERMINAL command. For more information, see TERMINAL in *z/VM: CP Commands and Utilities Reference*. To set default special symbol usage for the entire system, use the CHARACTER_DEFAULTS statement in the system configuration file. For more information, see "CHARACTER_DEFAULTS Statement" on page 67.

   You cannot use any of the letters A through Z, the numbers 0 through 9, or the bytes X'0E' (shift out) or X'0F' (shift in) for the line-end, line-delete, character-delete, or escape symbols.

Examples

1. To start a directory entry for user ID BONES that specifies:
   - WW11QQ as BONES's password
   - 8M as BONES's virtual machine storage size at logon
   - 16M as the maximum virtual machine storage size BONES can define
   - BONES can enter only class S commands

   specify the following USER statement:

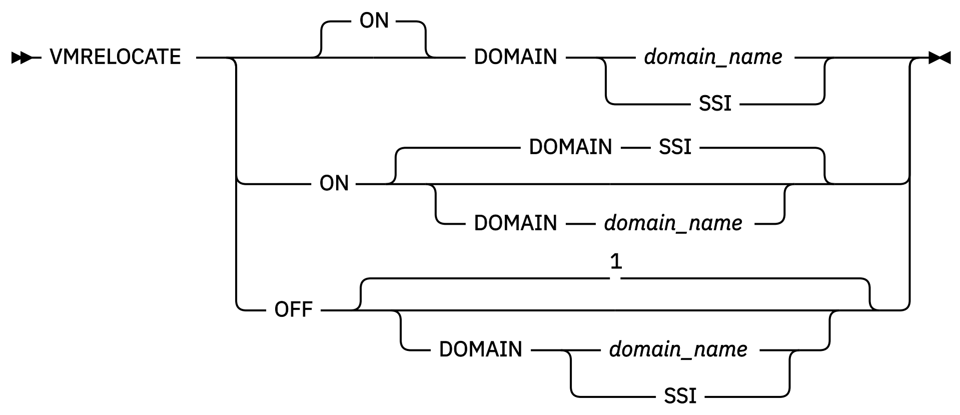
   ```
   User bones ww11qq 8m 16m s
   ```
2. To start a directory entry for user ID WIZARD that specifies:
   - JK76TG as WIZARD's password
   - 4M as WIZARD's virtual machine storage size at logon
   - 6M as the maximum virtual machine storage size WIZARD can define
   - WIZARD can enter class A, B, C, Z, 3, 4, and G commands

   specify the following USER statement:

   ```
   User wizard jk76tg 4m 6m abcz34g
   ```
3. To start a directory entry for user ID SHARDATA that specifies the NOLOG option, specify the following USER statement:

```
User shardata NoLog
```

VMRELOCATE Directory Statement



Notes:

[1] The default domain for the OFF setting is the single-member domain of the system where the user logs on.

Purpose

The VMRELOCATE statement sets a user's relocation domain, indicating the set of members of the SSI cluster to which it may be relocated, or disallows relocation of this virtual machine. When the user logs on, the virtual machine is assigned a virtual architecture level that is the maximal common subset of the architectural features of the cluster members in the specified or default relocation domain.

How to Specify

The VMRELOCATE statement is allowed in a user entry, or a profile included in a user entry, in an SSI-ready or SSI-enabled source directory. If specified, the VMRELOCATE statement must go before any device statements in an entry. (For a list of device statements, see Table 24 on page 462.) Only one VMRELOCATE statement is allowed in a user entry or profile entry. A VMRELOCATE statement in a profile entry is used only if no VMRELOCATE statement exists in the user entry.

Operands

ON
: enables relocation for the specified user. If neither ON nor OFF is specified, then ON is the default. When the user first logs on, relocation is enabled. If a relocation domain is not specified, the default is domain SSI.

OFF
: disables relocation for the specified user. When the user first logs on, relocation is disabled. If a relocation domain is not specified, the default is the single-member domain of the cluster member where the user is logging on.

DOMAIN *domain_name*
: specifies the relocation domain associated with the user when the user first logs on.

SSI
: is the relocation domain that includes the entire SSI cluster membership.

Usage Notes

1. The VMRELOCATE statement is not allowed in a multiconfiguration virtual machine definition. A relocation domain cannot be set for such a virtual machine because it cannot be relocated.

2. If you omit the VMRELOCATE statement when you code a virtual machine definition, relocation is enabled for that user by default, and the default relocation domain is domain SSI.
3. The relocation capability of the user can be dynamically changed with the CP SET VMRELOCATE command.
4. For more information about relocation domains and the virtual architecture level, see "Using Relocation Domains" on page 754.
5. APVIRT crypto users can relocate if there is an exact match in crypto capability between source and destination systems.

   APVIRT crypto users can also relocate if there is not an exact match, and if the following conditions are true:

   - APVIRT users are assigned to a relocation domain where the member systems have the Live Guest Relocation (LGR) for mixed APVIRT feature.
   - The destination system is configured with a compatible shared crypto pool. Compatibility requires that crypto facilities that are presented to the guest at the source system are supported at the destination system.

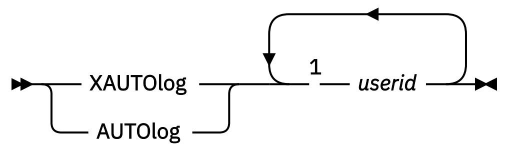
   APVIRT users that are assigned to a relocation domain where the member systems run LGR for mixed APVIRT see a virtual CEX adapter with the following characteristics:

   - The virtual CEX adapter has characteristics that are common to all the members of that relocation domain that run shared pools of the same mode.
   - The virtual CEX adapter presents the maximal common subset of capabilities for that mode.

   For example, the subset of member systems of a relocation domain that run an accelerator mode shared pool present a virtual CEX adapter with accelerator capabilities. The adapter presents the maximal common subset of accelerator capabilities to any guest with a virtual accelerator. Member systems of a relocation domain that run a CCA-coprocessor mode shared pool present a virtual CEX adapter with CCA coprocessor capabilities. That adapter presents the maximal common subset of CCA coprocessor capabilities to any guest with a virtual CCA coprocessor.

   For more information, see "Live Guest Relocation of APVIRT Virtual Machines" on page 37.

XAUTOLOG Directory Statement



Notes:

[1] A total of 8 user IDs can be specified.

Purpose

The XAUTOLOG statement designates up to eight users that can enter the XAUTOLOG command for the virtual machine. All users with class A or class B privileges can enter the XAUTOLOG command for any virtual machine. Therefore, their user IDs do not have to appear on the XAUTOLOG statement.

How to Specify

The XAUTOLOG statement is allowed in profile, user, and identity entries. If you specify the XAUTOLOG statement, it must precede any device statements you specify in a directory entry. (For a list of device statements, see Table 24 on page 462.) A directory entry can contain multiple XAUTOLOG statements, but only a total of eight user IDs (including those in AUTOLOG statements) is allowed.

Multiple XAUTOLOG statements are allowed within a profile entry, but are used only if no XAUTOLOG (or AUTOLOG) statements are in the including user or identity entry.

Operands

userid
specifies a 1- to 8-character user ID of a user whom you authorize to enter the XAUTOLOG command for the virtual machine whose virtual machine definition you are creating. A total of eight user IDs (including those in AUTOLOG statements) can be specified.

Usage Notes

1. AUTOLOG is an acceptable synonym for the XAUTOLOG statement. The minimum abbreviation is AUTO. The XAUTOLOG statement, whether named XAUTOLOG or AUTOLOG, authorizes the use of the XAUTOLOG command and does not authorize the use of the AUTOLOG command. The AUTOLOG command authorization is through the use of a password only.
2. In addition to being listed on an XAUTOLOG directory statement, a user must have class G privileges to enter the XAUTOLOG command for the virtual machine. (Users with class A or B privileges can enter the XAUTOLOG command for any virtual machine without being specified in an XAUTOLOG statement.)

Examples

1. To authorize a user whose ID is LARRY (and has class G privileges) to enter the XAUTOLOG command for a virtual machine, specify the following XAUTOLOG statement in the virtual machine definition:

```
XautoLog larry
```

2. To authorize users whose IDs are OPER1, OPER2, OPER3, and JONES (and have class G privileges) to enter the XAUTOLOG command for a virtual machine, specify the following XAUTOLOG statement in the virtual machine definition:

```
XautoLog oper1 oper2 oper3 jones
```

XCONFIG Directory Statement

Purpose

The XCONFIG statement specifies control parameters for the extended-configuration facilities provided in the XC virtual machine architecture: access lists and address spaces. Some of the control parameters specified on the XCONFIG statement are used only by XC virtual machines, whereas others are used in all types of virtual machines.

Each XCONFIG statement contains one of the mutually exclusive major operands ACCESSLIST or ADDRSPACE and specifies control parameters for a single aspect of the extended-configuration facilities. For example, a single XCONFIG statement can contain the ACCESSLIST operand with parameters for access lists, but cannot also contain the ADDRSPACE option with parameters for address spaces. Descriptions of the ACCESSLIST and ADDRSPACE operands follow.

To specify control parameters for more than one element of the extended-configuration facilities, you can specify multiple XCONFIG statements. However, each major option can each appear on, at most, one XCONFIG statement within a user, identity, or subconfiguration entry, and on, at most, one XCONFIG statement within a profile entry.

Any major option on an XCONFIG statement specified in the user or identity entry override those specified (or defaulted) in the profile entry. Operands that are specified (or defaulted) in the profile entry but not also specified in the user or identity entry are used as specified (or defaulted) in the profile entry.

Each XCONFIG keyword (ACCESSLIST and ADDRSPACE) is treated separately so that an XCONFIG statement in a subconfiguration entry overrides the identity entry only for the keywords specified on XCONFIG statements in the subconfiguration.

How to Specify

If you specify any XCONFIG statements, they must precede any device statements, both when specified within a user, identity, or subconfiguration entry and when specified within a profile entry. (For a list of device statements, see Table 24 on page 462.)

Usage Notes

1. The XCONFIG statements are processed by CP at the time of a user's logon, and the control parameters in effect at logon are used for the user's entire session. If changes are made to XCONFIG statements in a virtual machine definition, the user must log off and log on again for the changed parameters to take effect.

XCONFIG ACCESSLIST Operand

```
▶▶─ XCONfig ── ACCEsslist ── ALSIZE ── alecount ─▶◀
```

Purpose

The XCONFIG ACCESSLIST operand specifies the size of the host access list to be provided for this virtual machine. CP creates a host access list for each virtual machine. For XC virtual machines, each entry in the virtual machine's host access list designates an address space that can be accessed when the virtual machine is in the access-register mode. For XA, ESA, or Z virtual machines, each entry in the virtual machine's host access list designates an address space that can be accessed using DIAGNOSE code X'248' (Copy to primary service). Valid host access list entries are added and removed using the ALSERV macroinstruction (access list services). For more information, see ALSERV — Access List Services in *z/VM: CP Programming Services*.

How to Specify

The XCONFIG ACCESSLIST statement is allowed in profile, user, identity, and subconfiguration entries. If you do not specify an XCONFIG statement with an ACCESSLIST operand, CP will give the virtual machine a 62-entry host access list, by default.

Operands

ALSIZE *alecount*

specifies the size, in number of access-list entries, for this virtual machine's host access list. The variable *alecount* is a decimal number from 1 to 1022.

CP allocates host access lists in sizes that contain 6, 14, 30, 62, 126, 254, 510 or 1022 entries. If you do not specify *alecount* as one of these numbers, then CP rounds *alecount* up to the next largest number in this set.

Usage Notes

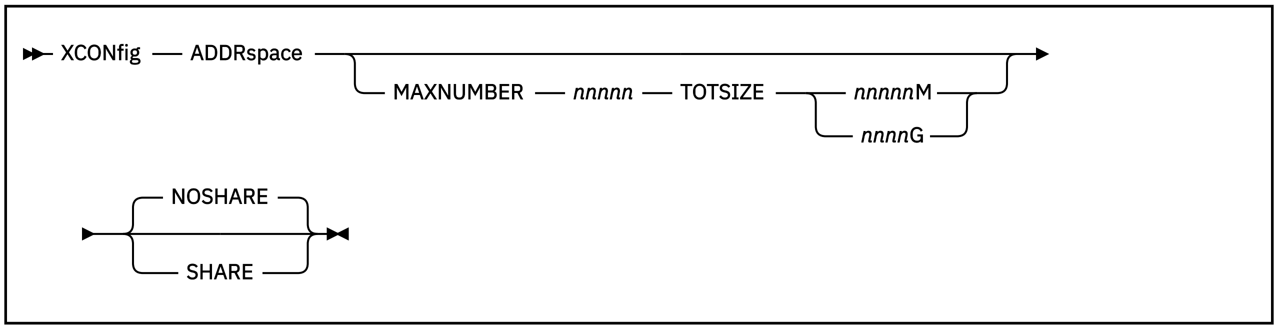
1. A virtual machine's host access list resides in host storage. When a host access list is initially created for a virtual machine, sufficient host storage is allocated for a 6-entry host access list. As more host access list entries are required, additional host storage is allocated to hold the next larger-sized host access list, until the access-list size reaches the size specified by this directory statement. CP will reuse any host access list entry that is in the invalid state before expanding the host access list in this manner, but once a host access list has been expanded, it will not be compressed if host access list entries subsequently return to being in the invalid state.
2. A 254-entry host access list can require up to one page of host storage and a 1022-entry host access list can require up to four pages of host storage.
3. The host storage allocated for a virtual machine's host access list is not available for use by CP for paging or other operations. To minimize the host storage used for access lists, allocate the smallest-sized host access list that meets the requirements for a particular virtual machine.

Examples

To specify that a virtual machine is to have a 126-entry host access list, use the following statement in the virtual machine definition:

```
Xconfig AccessList ALsize 126
```

XCONFIG ADDRSPACE Operand



Purpose

The XCONFIG ADDRSPACE operand specifies the maximum number of non-primary address spaces (sometimes known as data spaces) and the total size in bytes of all non-primary address spaces that the virtual machine can own simultaneously. These address spaces are created by using the ADRSPACE macroinstruction (address-space services) CREATE function. It also specifies whether the virtual machine can allow other virtual machines to share its address spaces. Only XC virtual machines can create and destroy address spaces using the CREATE and DESTROY functions of the ADRSPACE macroinstruction. ESA, XA, and XC virtual machines can allow other virtual machines to share their address spaces by using the PERMIT function of the ADRSPACE macroinstruction. (This is limited to sharing the virtual machine's primary address space.) For more information, see ADRSPACE CREATE, ADRSPACE DESTROY, and ADRSPACE PERMIT in *z/VM: CP Programming Services*.

How to Specify

XCONFIG ADDRSPACE is allowed in profile, user, identity, and subconfiguration entries. If you do not specify an XCONFIG ADDRSPACE statement, CP uses a default of 0 for MAXNUMBER, which indicates no additional address spaces can be created.

Operands

MAXNUMBER *nnnnn*
: specifies the maximum number of non-primary address spaces that this virtual machine can create and have existing concurrently. The variable *nnnnn* is a decimal number from 0 to 32767.

  If you omit the MAXNUMBER and TOTSIZE operands from the XCONFIG ADDRSPACE statement, the default is MAXNUMBER 0, that is, no additional address spaces can be created.

TOTSIZE *nnnnn*M
TOTSIZE *nnnn*G
: specifies the maximum total size, in bytes, of all address spaces that this virtual machine can create and have existing concurrently. The variable *nnnnn*M is a decimal number from 1 megabyte to 99999 megabytes. The variable *nnnn*G is a decimal number from 1 gigabyte to 8192 gigabytes.

  The minimum value for TOTSIZE is 1M for each address space permitted by the MAXNUMBER operand; if the TOTSIZE operand specifies a total less than this number, then an error message is issued. The maximum that can be specified for TOTSIZE is 8192G.

NOSHARE
: tells CP that the virtual machine cannot use the PERMIT function of the ADRSPACE macroinstruction to make its address spaces available for access by other virtual machines. The default is NOSHARE.

SHARE
: tells CP that the virtual machine can use the PERMIT function of the ADRSPACE macroinstruction to make its address spaces available for access by other virtual machines.

Usage Notes

1. Allowing users to create large numbers of address spaces or very large size address spaces might affect real storage availability. CP creates dynamic address translation (DAT) tables to reference the virtual machine storage. DAT tables include page tables and segment tables.

   CP keeps the page tables in page management blocks (PGMBKs). Each 8 KB PGMBK references 1 MB of virtual machine storage. PGMBKs might be pageable; as such, their impact on real storage depends on how frequently the MBs of storage they reference are used.

   Segment tables are allocated from host real storage and are not pageable. To reference the page tables for an address space, 1 - 4 contiguous frames are allocated for the segment table, one frame for each 512 MB of storage.

Examples

1. To authorize a virtual machine to create up to 128 address spaces, with a total of 1000 MB of storage, use the following statement in the virtual machine definition:

   ```
   Xconfig AddrSpace MaxNumber 128 TotSize 1000M
   ```

2. To authorize a virtual machine to create up to 25 address spaces, with a total of 2 GB of storage, and allow the virtual machine to share its primary address space and any of its created address spaces with other virtual machines, use the following statement in the virtual machine definition:

   ```
   Xconfig AddrSpace MaxNumber 25 TotSize 2G Share
   ```

3. To authorize a virtual machine to share its primary address space with other virtual machines, but not to create any additional address spaces, use the following statement in the virtual machine definition:

   ```
   Xconfig AddrSpace Share
   ```

Part 4. Storage Planning and Administration

Chapter 21. Host Storage Planning and Administration

This chapter provides information on assessing your real storage needs and managing your storage configuration.

Real Storage Management

The amount of real storage you need depends upon the type of work your installation will be performing and the number and size of the virtual machines you define. Your system might include many virtual machines, and each one requires some real storage.

For example, you may find that the processor does not have enough storage for your system to provide acceptable response times. If you find your system's response time to be slow or erratic, you may need to do one or both of the following:

- Add more real storage.
- Run fewer or smaller virtual machines.

The minimum required amount of real storage depends on whether z/VM runs as a first-level or second-level system. The minimum required amount of virtual storage depends on whether a second-level system is IPLed from a SCSI LUN. A first-level system is IPLed in a logical partition, a second-level system is IPLed in a virtual machine. If z/VM is IPLed with less than the minimum required real storage, the system enters a disabled wait state, code 955.

Table 37 on page 633 indicates real and virtual memory minimum requirements and maximum limits for z/VM first and second-level systems.

Using the IBM Hardware Management Console (HMC), you can define the `INITIAL` and `RESERVED` real storage for the logical partition in which z/VM is installed.

You can use the following tools to control the amount of storage that z/VM configures at system initialization:

- The STORAGE statement in the system configuration file
- The STORE=nnnnu IPL parameter that is passed to CP by the Stand-Alone Program Loader (SAPL)

For more information, see "STORAGE Statement" on page 278 and Passing IPL Parameters.

To display the status of real storage and how the storage frames are currently allocated, issue the QUERY FRAMES command.

Table 37. z/VM memory limits: Maximum limits and minimums to install and run z/VM

Resource	**z/VM 7.3**
Memory[1]: Maximum limits and minimum requirements to install and run z/VM	
Maximum real Memory See memory limits note "2" on page 634.	4 TB
Maximum virtual memory in a single virtual machine See memory limits notes "3" on page 634, "4" on page 634, and "6" on page 634.	2 TB with the PTF for APAR VM66673 See memory limits note "6" on page 634.
Minimum real memory size of the LPAR for installing z/VM as a first-level system	768 MB

Table 37. z/VM memory limits: Maximum limits and minimums to install and run z/VM (continued)

Resource	z/VM 7.3
Minimum real memory size to IPL z/VM as a first-level system	512 MB
Minimum virtual memory to install z/VM as a second-level system	128 MB
Minimum virtual memory to IPL z/VM as a second-level system See memory limits note "5" on page 634.	128 MB

Maximum memory limits and minimum memory requirements to install and run z/VM: Notes:

1. z/VM publications use the term *storage*.
2. Virtual to real memory ratio (practical) is about 2:1 or 3:1.

   With a definition for "Virtual to real memory" of total virtual machine size of started virtual machines to real memory configured to z/VM.

   Practical over commitment is dependent on factors such as Active:Idle virtual machines, Workload/Service Level Agreement sensitivity to delays, performance of paging subsystem (e.g. flash, HyperPAV, channels, etc.), accuracy of sizing of the virtual machines, and exploitation of memory saving/exploitation capabilities (e.g. CMM, DIM).
3. Practical limit can be gated by performance of Dumping a VM system, Live Guest Relocation requirements, and production level performance requirements.
4. Due to the number of supporting DAT tables needed, the limit of the sum of all virtual machines' instantiated pages is 64T.
5. If the second-level system is IPLed from an FCP SCSI LUN, the minimum virtual memory required is dependent on the model of the processor on which the z/VM system is running. To ensure success on all processor models, define at least 768 MB of virtual storage.
6. 2TB virtual storage is supported with 7.3 APAR VM66673 under a set of restrictions. Details about restrictions can be found at z/VM Memory Management (https://www.vm.ibm.com/memman/gt1guest.html).

For more information, see the following topics:

- "STORAGE Statement" on page 278
- "Virtual Machine Considerations" on page 641
- Passing IPL Parameters in *z/VM: System Operation*
- QUERY FRAMES in *z/VM: CP Commands and Utilities Reference*
- QUERY STORAGE/STORE in *z/VM: CP Commands and Utilities Reference*
- SET STORAGE in *z/VM: CP Commands and Utilities Reference*

Adding and Removing Memory Dynamically

Notes:

- In this chapter, the term *memory* refers to real memory. In z/VM commands and in some other references, real memory might also be called *storage*. However, *storage* can also refer to real disks, virtual disks, and storage volumes.
- *Dynamic memory downgrade (DMD)*, also known as *memory reclamation* or *storage reclamation*, extends the real storage dynamic management characteristics of z/VM to include removing the real storage from a running z/VM system.

z/VM memory reconfiguration gives a system administrator the ability to add or remove memory dynamically from a z/VM partition, making it available to other partitions in the central processor complex (CPC), without the need to re-IPL the system. This flexibility allows installations to react to changing workloads. In some cases, you might require extra memory on a regular basis (to process the weekly payroll, for example), while in other cases, an increased workload might be seasonal (due to the yearly tax season, for example). The maximum percentage of memory that you can define as being eligible for removal is indicated in Table 38 on page 637. Memory reclamation by a first-level z/VM system can be performed on the IBM z14 (or later) family of servers (or equivalent).

z/VM memory is classified as follows:

Permanent memory
is memory that cannot be taken offline.

Reconfigurable memory
is memory that can be taken offline.

You can define these two types of memory dynamically, using SET STORAGE commands, or at IPL time, using the STORAGE statement in the system configuration file. If the IPL parameter `STORE=` is used, it overrides the settings in the system configuration file. All memory defined via `STORE=` is brought online as permanent memory. The storage values on the STORAGE statement and the IPL parameter `STORE=` are rounded up to a multiple of the storage increment size, if necessary.

If neither the STORAGE statement nor the IPL parameter `STORE=` has been specified, all of the storage that z/VM finds configured in the partition is initialized as permanent storage. If this is the first IPL after the logical partition was activated, the amount of configured storage equals the initial storage allocation. If this is a subsequent IPL, the amount of storage configured during IPL depends on the configuration left by the operating system that was previously running in the partition.

For an IPL that is the result of an automatic restart, the AFTER_RESTART and AFTER_SHUTDOWN_REIPL operands on the STORAGE statement in the system configuration file determine z/VM's behavior in establishing the memory configuration. The AFTER_RESTART operand applies to an automatic re-IPL resulting from a CP abend or a PSW restart. The AFTER_SHUTDOWN_REIPL operand applies to an automatic re-IPL resulting from the SHUTDOWN command with the REIPL operand. Parameters on these operands provide the ability to initialize the storage configuration as though it is a fresh IPL, or to keep the storage configuration as it was at system termination prior to the re-IPL. See the “STORAGE Statement” on page 278 for more information.

Before reconfigurable memory can be brought online to z/VM, there must be a minimum of permanent memory already in use by z/VM, as indicated by Table 38 on page 637. The minimum provides enough memory above 2 GB to create most CP structures that require permanent memory, except for PGMBKs and long-term locked pages, which are discussed in “Workloads that Require Additional Permanent Memory” on page 639. Before memory can be reclaimed, CP will copy the page content within the memory elsewhere. Because only pageable structures can be easily moved, CP is selective about what is allocated in reconfigurable memory as compared to permanent memory. It is expected that even with the maximum percent of the system's storage defined as reconfigurable, there will not be a measurably adverse effect to the system or to a workload, except for workloads that are heavily using directly-attached FCPs, or those where memory touch patterns are especially sparse. See “Workloads that Require Additional Permanent Memory” on page 639 for more information.

After an IPL, you can use the SET STORAGE command to initiate memory reconfigurations to bring more permanent memory, reconfigurable memory, or both online to z/VM, or to take some or all of the reconfigurable memory offline. The following fields displayed by the RECONFIGURATION operand of the QUERY STORAGE command assist you in determining the extent to which memory is reconfigurable:

- `STORAGE=` is the amount of real memory currently online to z/VM. It is the sum of the `Permanent=` and `Reconfigurable=` values. Additional permanent or reconfigurable memory can be brought online, up to the `Maximum STORAGE=` value displayed. This maximum is the sum of the `Initial` and `Reserved` settings for the logical partition in the image profile, or the maximum amount of real memory that this z/VM system supports, whichever is less.

- The sum of CONFIGURED=, STANDBY=, and RESERVED= equals the sum of the `Initial` and `Reserved` storage allocations in the image profile for the logical partition, or the z/VM-supported maximum, whichever is less.
- Any added or removed memory must be specified in multiples of the storage increment size as indicated by the INC= value.
- Additional memory brought online to z/VM is obtained from standby memory (STANDBY=). The amount of standby memory at any time is determined by the machine, based on the amount of installed memory that is not currently claimed by other active logical partitions in the CPC, along with the `Initial` and `Reserved` memory allocations in the image profile for the partition. Although the sum of `Initial` and `Reserved` memory allocations in the image profile can exceed the z/VM-supported maximum, the STANDBY= value does not allow the system to grow beyond the z/VM-supported maximum. You can take manual actions to move memory from the RESERVED= state to the STANDBY= state, such as deactivating another logical partition or installing more real memory.
- Any amount of reconfigurable memory that is a multiple of the INC= value can be removed from the z/VM configuration. The removed memory increases the STANDBY= value, the RESERVED= value, or both.

For more information, see QUERY STORAGE/STORE in *z/VM: CP Commands and Utilities Reference*

Attention:

Good change management procedures suggest that you should avoid simultaneous storage configuration operations. Before issuing the SET STORAGE command:

1. Ensure that all other storage reconfigurations have completed. This includes logical partition activation or deactivation, storage configuration actions by any other operating system, and any other SET STORAGE command.
2. Issue the QUERY STORAGE command as necessary to ensure that the reported STANDBY= and RESERVED= values are stable.
3. If you are increasing storage, make sure the desired increase in storage size is available:

   ```
   increase <= STANDBY
   ```

   If you are decreasing storage, ensure that QUERY STORAGE shows that the desired decrease in storage size is available:

   ```
   decrease <= Reconfigurable
   ```

4. Only then issue the SET STORAGE command.

   Note: Removing storage can cause additional paging space usage. Always make sure your paging space is sized appropriately for your system, before and after a storage change. For more information about sizing your paging space, see Chapter 23, "Allocating DASD Space," on page 647.

z/VM supports central storage configurations established by logical partition activation, or through use of the SET STORAGE command by a previous instance of z/VM running in the logical partition. In addition, when running at second level, z/VM supports any virtual configuration established through the DEFINE STORAGE command. 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.

Table 38. z/VM memory limits: Dynamic memory management

Memory: Dynamic memory management limits	
Resource	**z/VM 7.3**
Minimum memory defined as Permanent Storage in order to use Dynamic Memory	4 GB
Maximum percentage of memory that can be defined as Reconfigurable memory with Dynamic Memory Management See memory limits note "1" on page 637.	50%

Dynamic memory management limits notes:

1. A minimum hardware bundle level is required to avoid a possible downgrade stall; refer to: https://www.vm.ibm.com/memman/dmd.html.

Considerations for Adding Memory Dynamically

To increase the likelihood of a successful memory add operation, the following steps are recommended before you issue the SET STORAGE command:

- Make sure all other memory reconfigurations have completed. This includes logical partition activation or deactivation, as well as memory configuration actions by other operating systems in the CPC.
- Issue the QUERY STORAGE command as necessary to make sure the reported STANDBY= and RESERVED= values are stable.
- Make sure available standby memory is equal to or greater than the amount of memory you want to add to the z/VM configuration.

If memory is being added to other partitions at the same time as a SET STORAGE command is being issued, it is possible that not all requested memory will be available to be added, because the STANDBY= value might still be changing. Adding memory is accomplished in two phases:

1. Changing standby memory from offline to online CONFIGURED.
2. Initializing online memory for use.

While Phase 1 is active, no other SET STORAGE reconfiguration command can be entered. It is recommended that you wait until the SET STORAGE command completes and memory is brought online before adding workload to the system by logging on additional users or by using live guest relocation (LGR). This ensures that the new workload runs as smoothly as possible. When this phase completes, the SET STORAGE command also completes. At this time, the command issuer and the system operator will receive an asynchronous message. You can stop this phase using the SET STORAGE HALT command.

Phase 2 operates in the background. Frames are initialized and made available as system needs arise. The `NotInitialized=` value displayed in the QUERY FRAMES command response is the count of uninitialized frames. The system initializes and uses these frames during normal operation.

After increasing real memory, reevaluate and appropriately increase the settings specified for SET RESERVED, SET MDCACHE STORAGE, and the SET VDISK system limit.

Considerations for Removing Memory Dynamically

Because you can use the SET STORAGE command at any time to remove reconfigurable storage (memory), z/VM limits allocations within reconfigurable storage to data that can be moved easily. To provide the most flexibility to z/VM in allocating memory, you should limit the amount of reconfigurable memory defined to be the amount you expect to eventually be removed. See "Workloads that Require Additional Permanent Memory" on page 639 for more information.

z/VM can convert some reconfigurable memory to permanent memory when there is an insufficient amount of permanent memory available for normal system operation. When this is done, a message is issued to the system operator and a monitor record is created. This permanent memory is not eligible to be removed from the system; that is, it will never be converted back to reconfigurable memory. If there is sufficient paging space and capacity to handle the system's permanent memory need, z/VM will not convert a zone of memory to permanent memory.

To increase the likelihood of a successful memory reclamation operation and to make sure the process completes as quickly as possible, the following steps are recommended before you issue the SET STORAGE command:

1. Finish any excess workload and, if possible, log off all users who were running the excess workload. Issue QUERY NAMES to make sure the logoff process has finished for these users.
2. If enough memory exists to do so, finish any live guest relocation operations to or from the system. If memory is being reclaimed because users are relocating away from the system, see "Live Guest Relocation and Memory Reclamation" on page 640.
3. As much as possible, make sure the system is running at a steady state. This allows the CP reclamation viability check to be as accurate as possible.
4. Use the TEST operand on the SET STORAGE command to estimate how much more paging space might be required after a reclamation, and evaluate whether this is acceptable in your environment.

The MAXPAGEFULL operand on the SET STORAGE command specifies the upper bound on the number of paging slots in use as a percentage of the total number of slots on all paging volumes. The *nnn* value is a decimal integer in the range of 0 to 100. This value is used in the storage reclamation viability checks performed during execution of this SET STORAGE command to make sure that the removal of memory will not cause the current workload to exceed the available space in the storage hierarchy (real storage plus paging space). The viability check is performed when the TEST operand is specified, and also during a storage reclamation process (at the start of and periodically throughout the reclamation). When a MAXPAGEFULL value is not specified, the default value used is the warning percentage specified on the PAGING system configuration statement or on a previous SET PAGING command. Specification of the MAXPAGEFULL parameter on a SET STORAGE command does not alter the warning value established by the PAGING system configuration statement or by a previous SET PAGING command. If at any time the viability check fails, the system does not remove any additional memory and it issues a message to the user who initiated the memory reclamation and to the system operator, if they are different. The viability check can be overridden with the FORCE option.

If AGELIST KEEPSLOT YES is specified or defaulted to in the system configuration file, pageable pages can have copies in real memory and also on paging volumes. In this case, the reclamation viability check might be overly-conservative because it does not consider the number of guest pages having two copies in the memory hierarchy. If the amount of unused real memory is less than the amount of memory being removed, a memory reclamation will push pages out to paging media. The viability estimation assumes that every page written to paging media is not already associated with a slot, and will therefore increase the amount of paging space in use. Depending on the amount of paging space currently in use in the configuration, this could result in a failed viability check and an unsuccessful memory reclamation.

When considering whether to use the FORCE option, take into account the complete memory picture on your system, including real memory, paging capacity and usage, and virtual memory (for example: instantiated, resident, and DASD page counts for users, as well as writeable NSS/DCSS pages, CP pageable pages, and CP control structures). The VIR2REAL and CHKRECLM EXECs can be helpful in assessing memory usage and determining if the FORCE option is appropriate. The VIR2REAL EXEC gathers information about the real and virtual memory on the system as it currently appears (that is, based on the guests that are logged on and the current paging usage). The CHKRECLM EXEC accepts a memory amount as input, and uses information about the current memory configuration and utilization to model how your system might look after a reclamation. It shows how the reclamation could change the virtual to real ratios, real memory, and paging. Both EXECs are available on the z/VM downloads page: VM Download Packages (https://www.vm.ibm.com/download/packages/).

The amount of time needed to complete a memory reclamation depends on a number of factors, including:

- The amount of memory being removed
- The state of the guest pages within the storage
- The availability of frames in which page content must be moved
- The paging bandwidth.

Issue the QUERY STORAGE command with the RECONFIGURABLE operand to display the status of a memory reclamation in progress. The command issuer will receive an asynchronous message when all memory is vacated and taken offline.

There is no limit on how much reconfigurable memory can be taken away in a single SET STORAGE command. However, all memory is divided into zones and not all zones that were chosen to be removed will be vacated at once. The system limits the number of zones being vacated at any time, based on how many processors are available to do the work. So, although you might request that 500 GB of memory be removed, the removal might take place in smaller chunks to avoid affecting other workloads on the system.

While the memory is being vacated and removed, it is recommended that you avoid logging on and logging off users, because these actions can cause additional paging activity, which can slow down the memory reclamation and the user activity. Additionally, increasing the workload or active memory footprint of guests during a reclamation can result in increased paging and slower memory reclamations. As memory is vacated, pages are moved from one zone to another, so reclamations do not necessarily cause increased paging. However, increased paging might be necessary to create available frames to receive the vacating memory. Thus, adding workload during a reclamation can result in unexpected paging spikes, because the workload and the reclamation are searching for available frames. The MAXPAGEFULL operand on the SET STORAGE command helps you avoid increasing paging media use beyond the threshold with which you are comfortable. After reclaiming real memory, reevaluate and appropriately decrease the settings specified for SET RESERVED, SET MDCACHE STORAGE, and the SET VDISK system limit.

Configuring Permanent and Reconfigurable Memory

During a fresh IPL (an IPL that is not an automatic restart), CP uses the PERMANENT and RECONFIGURABLE specifications on the STORAGE statements in the system configuration file to determine how to initialize memory. A minimum amount of permanent memory online is required before any reconfigurable memory can be configured. See Table 38 on page 637. In general, you should limit the amount of reconfigurable memory you define to the amount you expect to eventually remove. For additional guidelines for configuring permanent and reconfigurable memory, see the Usage Notes section of the "STORAGE Statement" on page 278.

It is recommended that you use the STORAGE statement to specify permanent and reconfigurable memory. While you can use the IPL parameter STORE= to specify the amount of permanent memory that comes online during IPL, the use of STORE= is not recommended. If you use this parameter, the STORAGE PERMANENT and STORAGE RECONFIGURABLE statements in the system configuration file will be ignored. After an IPL, you can use the SET STORAGE command to add more permanent or reconfigurable memory, or to remove reconfigurable memory.

For more information, see the description of the SET STORAGE command in *z/VM: CP Commands and Utilities Reference*.

Workloads that Require Additional Permanent Memory

Most workloads can be run in the configured real memory of between 50% and 100% of permanent memory. However, certain types of workloads are known to need more permanent memory:

- Workloads with heavy PGMBK use. PGMBKs are structures that describe virtual memory. Two frames are required for each PGMBK. Certain test workloads with a high overcommit ratio (much more virtual memory than real memory) with a sparse memory touching pattern (touching only a few 4 KB pages per 1 MB segment, for example) create more PGMBKs. In extreme cases, PGMBK use can make up a majority of real memory use. Workloads with heavy PGMBK use, with little other guest memory use, should be run with all permanent memory.

- Heavy use of certain CP functions and commands can cause additional permanent memory use. These include:
  - The CP LOCK command
  - The DISPLAY and DUMP guest storage commands
  - The SPXTAPE command
  - The VMDUMP command
  - Guest coupling facility use
  - Heavy use of CTCAs for guest use or ISFC (outside of SSI usage, see "Live Guest Relocation and Memory Reclamation" on page 640)

These functions are a concern only if they are likely to use a majority of guest pages.

It is important to plan for how much memory should be designated as reconfigurable. If a system with reconfigurable memory performs paging more often than you would like it to, you will need to take into account the types of workloads described in this topic. If you have one of these types of workloads, adding reconfigurable memory might not result in less paging. There are monitor records available to help determine if reconfigurable memory is being used fully. If reconfigurable memory is not being used fully, adding permanent memory is the best way to reduce paging. For more information, see the DMD performance report at IBM: z/VM Performance Report (https://www.ibm.com/vm/perf/reports/).

Live Guest Relocation and Memory Reclamation

You might want to use live guest relocation (LGR) within an SSI cluster, along with memory reclamation, to move workload and memory within a CPC. This can help mitigate the effect of planned outages without requiring massive amounts of excess memory.

It is not recommended that you remove all planned reconfigurable memory from the source system and to then begin moving users. The best practice is to alternate removing some memory from the source system, adding some memory to the target system, and then moving some guests until all guests and memory are moved. For IBM-provided examples of this, see the DMD performance report at IBM: z/VM Performance Report (https://www.ibm.com/vm/perf/reports/).

You should always keep the system at a steady state while doing memory adds or removes, to ensure that any LGR operations have finished, before you issue the SET STORAGE command. Failure to do so can result in increased time needed for commands to complete and/or increased paging activity.

When considering a live guest relocation, you should also take into account the size of permanent memory on the source and target. There are eligibility checks available as part of a live guest relocation that determine whether the guest's permanent memory usage will overwhelm available permanent memory and paging space on the target system. For more information about eligibility checking, see the VMRELOCATE command in *z/VM: CP Commands and Utilities Reference*.

Understanding the DSR Unit Size

A dynamic storage reconfiguration (DSR) unit size is associated with the ability of a second-level z/VM system running in a virtual machine to increase or decrease the size of the central storage configuration using CP commands. Memory must be added or removed using values that are multiples of the DSR unit size, which is the greater of 128 MB and the increment size.

For example, if the increment size you specify on the DEFINE STORAGE command using the INCREMENT operand, or the increment size determined by the DEFINE STORAGE command, is less than 128 MB, a second-level z/VM system running in a virtual machine will apply a DSR unit size of 128 MB in place of the storage increment size. When this occurs, the increment size (INC=) displayed using the QUERY STORAGE command on the second-level z/VM system will be different from the increment size (INC=) displayed by the QUERY VIRTUAL command issued by the first-level z/VM instance. Because storage must be added or removed from an operating system's storage configuration in units equal to the storage increment size, z/VM's usage of the DSR unit size as the effective storage increment size can make a portion of a virtual storage configuration unusable by a second-level z/VM system. In addition, storage values that normally round up to a storage-increment multiple are rounded down when insufficient configured or standby storage exists to round up.

For more information, see the descriptions of the DEFINE STORAGE and SET STORAGE commands in *z/VM: CP Commands and Utilities Reference*.

Storage Requirements for Generating a CP Module

The CP module requires approximately 17 MB of real storage. This amount will vary slightly based on the parameters specified on the STORAGE statement and the number of devices specified on RDEVICE statements in the system configuration file. Additional virtual storage in the virtual machine generating the CP module is required for CMS, the VMFBLD program, and access to the CMS disks containing the CP module text decks and related files. A virtual machine size of 40 MB should be sufficient.

In order to IPL the system, there must be sufficient available storage to hold the warmstart area. This storage is returned to general system use once spooling is initialized. Usually, one cylinder of CKD DASD or 75 four-kilobyte (4 KB) pages of FBA DASD are defined for the spooling warmstart area. This is easily contained in a small virtual machine. However, if you define a large warmstart area, the virtual machine size must be increased to accommodate it.

Other factors affecting storage requirements include the size of the trace tables and the number of devices defined in the system.

Host Logical Storage

The address space in which CP executes is called the system execution space. This address space (also known as host logical storage) consists of three parts:

- System execution area
- System execution space page management table
- System frame table

The system execution area contains the CP nucleus, the prefix pages, dynamically allocated free storage and other CP-use pages, aliases (references to guest pages), and routines loaded with the CPXLOAD system configuration statement or the CP CPXLOAD command. The system execution area resides below 2 GB; its size depends on the real storage size. The system execution space page management table and the system frame table reside above 2 GB. To display the host logical storage address ranges that are in use or usable by the system for the system execution space, issue the CP QUERY SXSSTORAGE command.

Host logical storage addresses in the CP nucleus and the prefix pages are identity mapped to real storage (that is, the host logical storage addresses and host real storage addresses are identical). The rest of the system execution area is dynamic (not identity mapped). Host logical storage pages must be backed with frames in real storage before they can be used, but backing frames are not necessarily contiguous. Frames below 2 GB are used only when required, freeing CP to exploit backing frames above 2 GB for most operations. To check the status of pages in the system execution area, issue the CP QUERY SXSPAGES command. This command displays information such as the number of pages that are available or in use; the number of pages that are backed below 2 GB, above 2 GB, or unbacked; and the number of pages that are locked or reserved.

For descriptions of QUERY SXSSTORAGE and QUERY SXSPAGES, see *z/VM: CP Commands and Utilities Reference*.

Virtual Machine Considerations

Each virtual machine requires some real storage. Allowing many virtual machines to have large storage sizes (primary address spaces or data spaces) might affect real storage availability. CP creates dynamic address translation (DAT) tables to reference the virtual machine storage. DAT tables include page tables, segment tables, and higher level (region) tables.

CP keeps the page tables in page management blocks (PGMBKs). Each 8 KB PGMBK references 1 MB of virtual machine storage. PGMBKs might be pageable; for example, PGMBKs for nonshared guest pages are pageable. CP creates a PGMBK only when the corresponding MB of virtual machine storage is first referenced. However, if many large virtual machines are very active, it could affect the availability of real storage.

Segment tables and region tables are allocated from host real storage and are not pageable:

- To reference the page tables for a primary address space or data space up to 2 GB, 1 to 4 contiguous frames are allocated for the segment table, one frame for each 512 MB of storage.
- For a primary address space larger than 2 GB, multiple segment tables are created, plus one or more region tables to reference the segment tables. Each region table occupies 1 to 4 contiguous frames. If needed, multiple levels of region tables are created.

PGMBKs, segment tables, and region tables are allocated above or below 2 GB by round robin on ring 3.

You should consider the combined effect of giving many users large storage sizes and allowing users to create many data spaces, large data spaces, or both. Considerable real storage might be consumed by the PGMBKs, segment tables, and region tables.

Chapter 22. Minidisk Cache Planning and Administration

For performance recommendations and guidance in the usage of central storage for minidisk cache, see "z/VM Minidisk Cache" on the z/VM performance web site at IBM: VM Performance Resources (https://www.ibm.com/vm/perf/).

Central storage can be used as a write-through minidisk cache for CMS or other guests' data. (Write-through means that anything written to the minidisk is also reflected in the cache.) This use of central storage is self-tuning and totally transparent. It can help eliminate I/O bottlenecks and improve system response time. Response time may be improved because many reads become synchronous instructions and less CP time is spent on I/O related tasks.

Using minidisk cache may reduce the complexity of configuring I/O and tuning. You may be able to eliminate replication of heavily used read-only shared minidisks such as the CMS S and Y disks and the tools disks. You can limit load balancing to write activity on DASD volumes. You can allow more DASD per control unit, while maintaining a given response time.

Central storage is faster than DASD control unit caches.

CP-managed centralized cache has automatic load balancing. There is no need to decide which DASD to cover. It also uses space more efficiently.

For cacheable I/O, data integrity is maintained by always updating the cache after an I/O operation. (This includes I/O error conditions.) Where minidisk caching has been specifically prohibited for an otherwise cacheable interface, cache invalidation may be used to ensure data integrity.

For non-cacheable I/O, CP cannot always determine which records on the minidisk are being referenced. CP causes cache invalidation if, after investigation of the channel program it determines the data cannot be cached. Data integrity is maintained by cache invalidation.

Blocks are discarded from the cache when the data on the physical volume is changed in a way that bypasses the cache. The cache is invalidated when:

- An I/O error occurs on cacheable I/O
- A noncacheable I/O operation occurs to a writeable minidisk that either:
  - May have data in the cache
  - Overlaps another minidisk which may have data in the cache
- DASD is detached from the system

Unnecessary cache invalidation decreases performance by reducing the cache hit ratio. To avoid problems, see "Planning for and Using Minidisk Caching" on page 645.

CP Minidisk Cache Enhancements

The following are potential items to consider when upgrading from a previous VM release:

- Review storage allocation for minidisk cache.

  The amount of real storage allocated for minidisk cache can be fixed or bounded (minimum and maximum) with the SET MDCACHE command. When the amount of storage allocated for minidisk cache is not a fixed amount, the arbiter function of minidisk cache determines how much storage to allocate.

  If the size of the minidisk cache was previously fixed or bounded, you will want to either allow the arbiter to dynamically determine the minidisk cache size or you will want to alter the values in consideration of real storage usage.

A method of influencing the arbiter is to set a bias with the SET MDCACHE STORAGE BIAS command. This can be preferable to setting a fixed minimum or maximum for systems where paging and/or minidisk cache size requirements vary significantly over time.

- Try letting the minidisk cache size default.

  For MDC, the default is no cache size restrictions and no biasing (that is, a bias setting of 1.00). Under these conditions, the real storage arbiter usually strikes a good balance between using real storage for MDC and using it for demand paging. The default settings should be good for most systems and are at least a good starting point. One potential exception is when storage is very unconstrained. In that case, bias settings in the range of 0.1 to 0.5 may give better results.

- Use SET MDCACHE or SET RDEVICE commands to enable minidisk cache on volumes.

  You cannot define all DASD volumes as shared volumes in the system generation or system configuration file and then enable particular volumes for minidisk cache with the SET SHARED OFF command. You must explicitly use the SET MDCACHE RDEV command or the SET RDEVICE command with MDC option to enable minidisk cache.

- Enable caching for minidisks that were poor candidates in the past.

  There may be minidisks that were disabled for caching because most of the I/O to them were writes. It may now be reasonable to enable these minidisks because the writes will not cause inserts in the cache and the reads might benefit from caching. Some overhead to check for data in the cache when performing a write I/O (to maintain integrity) does exist. Therefore, disabling minidisk cache for write-only minidisks would save some system resources.

- Disable caching for minidisks that are poor candidates.

  You may want to revisit and change the devices for which you disabled or enabled minidisk caching. There may be some minidisks that are poor candidates for minidisk cache, but did not matter in the past since the type of I/O or format made them ineligible for minidisk cache. With several restrictions being lifted, (such as the 4 KB block size) it may be worthwhile to revisit these minidisks and make sure they have minidisk cache disabled. VSE paging volumes may be candidates.

- Consider using minidisk caching with guest operating systems.

  Minidisk caching can often be used to improve the performance of guest operating systems. The main benefits of minidisk caching are to reduce the load on the I/O subsystem and to reduce elapsed time by eliminating read I/Os when the requested records are found in the cache.

  Minidisk caching uses a fair share concept to prevent any one virtual machine from flooding cache storage by inserting large amounts of data. As with servers that support many users, you should normally use the directory option NOMDCFS (no minidisk cache fair share) to turn off the minidisk cache fair share limit for production guest virtual machines.

- Prepare for minidisk caching on devices shared between first and second level systems.

  Care should be used for minidisks shared with first level systems in order to have all changes reflected to the second level system. For example, a minidisk is cached by a second level system and a change is made to the minidisk by the first level system. The change will not be reflected on the second level system if the data had been in the second level system cache. You must purge the cache on the second level system with the SET MDCACHE MDISK FLUSH command and reaccess the minidisk to see the changes.

- Avoid mixing standard format and non-standard format records on the same cylinder.

  Mixing different format records on the same cylinder makes the minidisk cache feature less efficient. The minidisk cache is more efficient for standard format records. However, when both formats exist on the same cylinder, CP treats all I/O to that cylinder as non-standard.

What Devices Can Be Cached

Minidisk caching does not apply to:

- FBA minidisks that are not page-aligned

- Virtual disks in storage
- Dedicated or attached DASD
- Shared DASD
- Extended address volume (EAV) DASD
- Devices other than DASD

Minidisk caching applies to linked minidisks. This includes:

- Normal user minidisks
- CMS S and Y disks
- Temporary disks
- Overlapping minidisks
- Minidisks formatted and reserved for use by the *BLOCKIO system service, for example:
  - SQL/DS minidisks
  - Shared file system minidisks.

Turning Caching Off

By default, CP caches all eligible minidisks on all volumes unless minidisk caching has been turned off. To turn off minidisk caching for the system, enter the command as follows:

```
set mdcache system off
```

To turn off minidisk caching for all minidisks on a volume, you can do one of the following:

- SET MDCACHE RDEV
- Define SHARED YES on the RDEVICE TYPE DASD system configuration file statement for that volume

To turn off minidisk caching for an individual minidisk, code the NOMDC option on the MINIOPT directory statement in the CP directory. For more information, see "MINIOPT Directory Statement" on page 560.

Planning for and Using Minidisk Caching

To control the sizes of the minidisk cache, use the SET MDCACHE command to set the minimum and maximum size. The size of the minidisk cache can be fixed by setting the maximum equal to the minimum. Use the QUERY MDCACHE command to display the current size of the minidisk cache and the minimum and maximum of its possible size.

To display the rate per second at which information is read and written to the minidisk cache, use the INDICATE LOAD command. This command also displays the number of successful cache lookups.

Attention: If different systems share a minidisk, use the MINIOPT directory statement to avoid caching those minidisks. If the minidisk is used in read-only status, it can be shared between two systems.

Special Considerations for CMS Minidisk Volumes Shared between Processors

If you have DASD volumes that contain CMS minidisks that are shared among different processors, you must take action before IPLing z/VM. This also applies if you are running z/VM as a guest when the guest will be using minidisk caching and the first-level system will be caching using minidisk caching.

For each DASD volume that contains CMS minidisks and that more than one processor physically shares, specify one of the following:

- SHARED YES on the RDEVICE TYPE DASD statement in the system configuration file

- SHARED YES on the SET RDEVICE TYPE DASD command

This prevents all minidisks on those volumes from being eligible for minidisk caching. If you do not specify YES for the SHARED parameter or operand, a data integrity exposure can exist. The exposure is caused if any system other than the one that is using minidisk caching to cache data has write access to the disk.

The SET SHARED ON command disables minidisk caching for all minidisks on a real DASD volume. The SET MDC RDEV command disables minidisk caching on a volume, and the SET MDC MDISK disables caching for a single minidisk. Defining a shared device using either the SET RDEVICE command or the RDEVICE system configuration file statement, described in the previous list, is the recommended method for shared volumes. It prevents any caching of data that might occur before the SET SHARED command can be entered. For example, caching might occur as part of initialization of operator and autologged virtual machines.

Notes:

1. One possible exception is that if only one processor among the sharing processors is writing, and the same processor is the only one trying to cache data for that volume, for that processor only, minidisk caching may be allowed (YES need not be specified for the SHARED parameter or operand). For all other sharing processors, YES should be specified for the SHARED parameter or operand.
2. The SET SHARED command and either the SET RDEVICE SHARED YES command or the RDEVICE SHARED YES system configuration file statement can also be used to specify whether virtual reserve/release CCWs should be translated to real reserve/release CCWs. For more information, see Chapter 25, “DASD Sharing,” on page 667.

Performance Considerations

Minidisk cache should be turned off for minidisks that are only read and written via MAPMDISK and the corresponding data space. Minidisks which use a combination of data space access/MAPMDISK and virtual I/O (Diagnose and channel program) should be evaluated on an individual basis to determine if minidisk cache should remain enabled.

To prevent unnecessary cache invalidation, do the following:

- Perform backup with read-only links whenever possible.
- Consider restoring minidisks without full-pack writeable access as follows:
  - Link to the user's minidisk rather than relocating it within a full-pack minidisk
  - Define another minidisk to exactly overlap the one being restored.
- Consider disabling caching for minidisks that are accessed by both CMS users running on the first-level z/VM system and by CMS users running on a second-level guest where the second-level system has write access.

In addition, to improve performance when using minidisk caching, do the following:

- Eliminate replicated volumes.
- Align FBA minidisks on page boundaries.
- Specify NOMDC on the MINIOPT directory statement for minidisks not to be cached. Specifically, do this for write-only minidisks such as those used for journaling, logs, and tracking.

Minidisk caching is generally self-tuning. You may either need to try some manual tuning (such as the recommendations above) or add real storage if your installation experiences:

- A large oscillation in the number of blocks in the cache. (Paging is causing less expanded storage from being used for caching.)
- A high I/O activity to DASDs that contain cacheable minidisks.
- A low minidisk caching hit ratio.

Chapter 23. Allocating DASD Space

This chapter tells you how to calculate disk space for the following CP storage areas:

- CP module
- Warm start data
- Checkpoint data
- User directory
- Paging space
- Spooling space.

In addition, it includes an explanation of adding CP DASD space to a running z/VM system.

Direct Access Storage Requirements

In the following paragraphs you will find many references to disk space. It is important to understand the fundamentals of disk space to avoid confusion when dealing with various DASD types.

It is helpful to understand the z/VM requirements for disk space in general. It is also helpful to understand the differences and similarities between the z/VM view of Count-Key-Data (CKD) and fixed block architecture (FBA) devices. Examples of these devices are:

CKD
3380, 3390

FBA
9336, emulated 9336 on SCSI disk

Note: z/VM may support some DASD types only as an emulation mode on other DASD. See the list of supported devices in *z/VM: General Information*.

z/VM's reference to disk space is always done in units called pages. A disk page is a 4096-byte record on disk. This means that z/VM requires that all its disk space (warm start data, directory, and so on) be formatted as 4096-byte records. z/VM also requires that you identify the use for which specific pages on disk are allocated to each type of z/VM reference. For example, you must identify pages for paging, for the directory, and for other areas.

CKD Device Geometry

z/VM views CKD devices by their geometric characteristics (such as number of cylinders). Each cylinder has a fixed page capacity, meaning that a fixed number of 4 KB records fit on a cylinder. This number varies for each CKD DASD type. z/VM references its data by a cylinder number and a page number on that cylinder. For z/VM space you must format and allocate pages in groups of cylinders.

Note: z/VM supports only CKD devices that accept the Extended-Count-Key-Data (ECKD) channel program protocol. Space on these devices is also expressed in units of cylinders. For more efficient disk I/O, an extended channel command architecture is used. This architecture is based primarily on the CKD command set. Therefore the term CKD, as used in this and other z/VM books, really refers to ECKD.

Later in this chapter you are asked to figure your particular DASD requirements as a number of records or pages, then convert this number to an equivalent number of cylinders. This means dividing the page requirements by the number of pages per cylinder for particular device types. Allocating space for minidisks on z/VM-owned ECKD volumes is also done in units of cylinders. Use of space within this allocation is also done in units of cylinders by way of the MDISK directory statement. This is convenient because no conversion is required in this case.

FBA Device Geometry

FBA devices appear as linear address spaces of 512-byte blocks. The blocks are consecutively numbered from 0 to n, where n is the highest block number on the volume. z/VM groups eight consecutive blocks to form a z/VM page and this page must begin on a 4 KB boundary. z/VM then views the volume as a collection of pages numbered from 0 to (n-7)/8. For example, blocks 0-7 make up page 0. There is no concept of cylinder boundaries in this structure.

You must allocate space on FBA volumes in units of pages (in contrast to units of cylinders on CKD volumes). When you figure your disk space requirements as a number of pages, you can use these numbers directly in system configuration file statements or in the ICKDSF service program (pages). Page numbers must be multiplied by 8 to convert them into block numbers used by the MDISK directory statement. When assigning minidisk space, you must know the available extents of your disk in block numbers (see Table 40 on page 649).

Note: It is recommended that all FBA minidisks as well as full-pack minidisk volumes be aligned on page boundaries for optimization of disk space. Otherwise, the resulting residual blocks of disk space might not be formatted and utilized. Additional restrictions apply for SFS file pool minidisks that are not page-aligned. For more information, see *z/VM: CMS File Pool Planning, Administration, and Operation*. If the starting block number is a multiple of eight and the number of blocks is a multiple of eight, then the minidisk is defined to be on a page boundary.

It is helpful to understand how much disk space you need to allocate for CP storage areas. This chapter shows you how to calculate CP storage areas in terms of the number of cylinders on the various types of CKD devices and the number of 4 KB pages on FBA devices.

Storage Calculations

For each area of storage, a calculation to determine the number of cylinders or 4 KB pages needed is given. You will need to refer to information in the system configuration file for some of the calculations.

Table 39 on page 648 lists the storage capacity of various CKD devices. Table 40 on page 649 lists the storage capacity of FBA devices.

Table 39. CKD DASD Storage Capacities

DASD Type	Pages per Track	Tracks per Cylinder	Cylinders per Disk	Pages per Cylinder
3380 Model K	10	15	2655	150
3380 Model E	10	15	1770	150
3380 All other models	10	15	885	150
3390-1	12	15	1113	180
3390-2	12	15	2226	180
3390-3	12	15	3339	180
3390-9	12	15	10017[1]	180
3390-A	12	15	Note [2]	180

Table 39. CKD DASD Storage Capacities (continued)

DASD Type	Pages per Track	Tracks per Cylinder	Cylinders per Disk	Pages per Cylinder

Note:

[1] Value can be up to the maximum number of cylinders on the disk.
[2] As allowed by the hardware.

Note: One page of host storage is equivalent to 4096 bytes of contiguous disk space.

Table 40. FBA DASD Storage Capacities

DASD Type	Number of Blocks	Number of Access Positions	Number of Blocks Per Access Position	Number of Pages Per Access Position
9336-20	1,672,881	2153	777	111
Emulated 9336-20 on SCSI	2,147,483,640[1, 2]	–	777[3]	111[3]

Notes:

[1] FBA DASD space allocated by CP for use as directory, paging, and spooling must reside within the first 16,777,215 pages (64 GB minus 1 page) of the volume.
[2] Because the CMS file system requires file status and control information to reside below 16 MB in virtual storage, there is a practical limitation on the size of CMS minidisks. As a minidisk increases in size, or more files reside on the disk, the amount of virtual storage associated with the disk for CMS file system status and control increases in storage below 16 MB. The current ECKD DASD limitation for CMS is 65520 cylinders for a 3390 disk on an IBM TotalStorage DASD subsystem, or about 45 GB of data. The maximum size for FBA SCSI disks supported for use by CMS or GCS is 381 GB. IBM suggests that customers defining disks for use by CMS should set a practical limit of about 22 GB. If larger disks are defined, they should be limited to contain very few files, or the CMS file system may not be able to obtain enough virtual storage below 16 MB to format or access those disks. For more information, see the ACCESS command in *z/VM: CMS Commands and Utilities Reference*.
[3] These values are simulated for emulated 9336-20 on SCSI and are not directly related to the total number of blocks allowed on real SCSI hardware.

CP Module

The CP module resides on a CMS-formatted minidisk, usually the parm disk. You need to allocate sufficient disk space for the parm disk, which may contain:

- System configuration files
- One or more CP modules
- Logo files
- Text decks for CP exits

The size of the CP module depends on the size of:

- IBM-supplied text decks
- Customized HCPSYS (system definition) and HCPRIO (real I/O configuration) ASSEMBLE files, if used
- Other customized text decks included in the CP module
- Use of the HCPLDR option PAGEB

The size of the system configuration file does not affect the size of the CP module. Therefore, if you are upgrading from a previous VM release in which you used the HCPSYS and HCPRIO files to define your system, IBM strongly recommends that you migrate all of the data from your HCPSYS and HCPRIO files to the system configuration file.

Table 41 on page 650 shows the minimum contiguous cylinders/4 KB pages required for the CP module when you are not using the HCPSYS and HCPRIO files or the HCPLDR PAGEB option.

Table 41. Minimum Space Required for the CP Module

Device	**Number of Cylinders or Pages**
3390	18 cylinders
FBA	3200 pages

Warm Start Data

You specify the number of cylinders or 4 KB pages to allocate to the warm start area. The number required is based on the maximum number of spool files that can exist at one time, which in turn is determined by the number of users defined in the directory and the number of spool files permitted each user, or the number of cylinders or pages available for spool files. You can specify up to 9 cylinders (CKD DASD) or 2000 4 KB pages (FBA DASD) for the warm start area. The warm start and checkpoint save area cannot begin at cylinder 0 (CKD DASD) or before page 16 (FBA DASD).

Use the WARMSTART operand of the SYSTEM_RESIDENCE statement in the system configuration file to specify the location and size of the area to be used for warm start data:

- The starting location is specified as the cylinder or page number that designates the real starting location where the CP warm start information is to be saved. For CKD device types, this value is a nonzero decimal cylinder number ranging from 1 to 65511. For FBA devices, this value is a 4 KB page number beginning with page 16 or greater.
- The maximum size of the warm start area is specified as the number of cylinders or 4 KB pages to be allocated for the warm start information area. For CKD devices, this must be a number from 1 to 9 that represents cylinders. For FBA devices, this must be a number from 1 to 2000 that represents 4 KB pages. You can use one of the following formulas to calculate the size.

If the warm start volume is a CKD DASD, use this formula to calculate number of cylinders required:

```
     maxfiles
n =  ------------
     1022 x ppc
```

If the warm start volume is an FBA DASD, use this formula to calculate number of 4 KB pages required:

```
     maxfiles
p =  ----------
       1022
```

n
: is the number of cylinders required for the warm start area.

p
: is the number of 4 KB pages required for the warm start area.

maxfiles
: is the maximum number of spool files that can exist at one time.

ppc
: is a device-dependent value selected from Table 39 on page 648. The value represents the number of 4 KB pages that will fit on one cylinder.

Example

For example, assume you chose a 3390 DASD as the device to be used for warm start data. Each 4 KB page of a CKD DASD allows 1022 spool file pointers. A 3390 DASD has 180 four-kilobyte (4 KB) pages per cylinder. Therefore each cylinder of a 3390 DASD can point to 183960 files. So a 9-cylinder warm start area (the maximum) allows a system-wide maximum of more than 1.6 million (1655640) spool files.

For information on how to specify the operands, see "SYSTEM_RESIDENCE Statement" on page 292.

Checkpoint Data

You specify the number of cylinders or 4 KB pages to allocate the checkpoint area. The number required for the checkpoint area is based on the number of CP-owned volumes in the configuration, the number of real spooling devices defined in the real I/O configuration, and the maximum number of users that will be logged on at the same time. Accounting, EREP, and symptom records are also stored in the checkpoint area. Normally, these records are retrieved by virtual machines identified in the system configuration file. But if those virtual machines are not actively retrieving the records, they will accumulate and take up space in the checkpoint area.

You can specify up to 9 cylinders (CKD DASD) or 2000 four-kilobyte (4 KB) pages (FBA DASD) for the checkpoint area. The warm start and checkpoint save area cannot begin at cylinder 0 (CKD DASD) or before page 16 (FBA DASD).

Use the CHECKPOINT operand of the SYSTEM_RESIDENCE statement in the system configuration file to specify the location and size of the area to be used for checkpoint (system restart) data:

- The starting location is specified as the cylinder or page number that designates the real starting location where the CP warm start information is to be saved. For CKD devices, this value is a nonzero decimal cylinder number ranging from 1 to 65511. For FBA devices, this value is a 4 KB page number beginning with page 16 or greater.
- The maximum size of the checkpoint area is specified as the number of cylinders or 4 KB pages to be allocated for the checkpoint information area. For CKD devices, this must be a number from 1 to 9 that represents cylinders. For FBA devices, this must be a number from 1 to 2000 that represents 4 KB pages. You can use one of the following formulas to calculate the size.

If the checkpoint volume is a CKD DASD, use this formula to calculate number of cylinders required:

```
n  =   ppc+4+((nvl+119)/120)+((nrs+60)/61)+(((nau*10)+39)/40)
       -----------------------------------------------------
                              ppc
```

If the checkpoint volume is an FBA device, use this formula to calculate number of 4 –KB pages required:

```
p  =   100+((nvl+119)/120)+((nrs+60)/61)+(((nau*10)+39)/40)
```

n
: is the number of cylinders required for the checkpoint area.

p
: is the number of 4 KB pages required for the checkpoint area.

ppc
: is a device-dependent value selected from Table 39 on page 648. The value represents the number of 4 KB pages that will fit on one cylinder.

nvl
: is the number of CP-owned volumes present in the configuration when the checkpoint process began.

nrs
: is the number of real spooling devices defined in the real I/O configuration.

nau
: is the number of users active on the system when the checkpoint process began.

Directory Space

To allocate a disk so that it contains space for the directory, use the Device Support Facilities program (ICKDSF). See the *Device Support Facilities User's Guide and Reference* book for additional information. The z/VM user directory cannot begin before page 16 on FBA DASD.

If you follow the installation procedures in the *z/VM: Installation Guide*, you should have enough disk space for two directories. This allows you to build a new directory whenever needed, without reformatting and reallocating space each time. If you wish to reallocate the area on which the directory resides, you may use the ICKDSF program, which can be invoked by the CP utility CPFMTXA.

A minimum of 4 cylinders (CKD DASD) or 600 pages (FBA DASD) should be allocated to the z/VM directory. This provides for two directories of 2 cylinders (CKD DASD) or 300 pages (FBA DASD) each. The space required for the directory depends on the number and size of the virtual machine configurations defined. If you have not allocated enough space for your directory, you will receive a message saying so.

You can estimate the minimum disk space required for your directory by using one of the following formulas:

For CKD DASD, use the formula below (round up any fraction to the next whole number) to calculate the minimum number of cylinders required to contain the object directory:

```
             tcbs
cylinders= -------- * 2
             ppc
```

For FBA DASD, use the formula below to calculate the minimum number of pages required to contain the object directory:

```
pages= tcbs * 2
```

cylinders
is the minimum number of cylinders of the CKD DASD type targeted to contain the object directory.

To calculate the minimum disk space required for an object directory, use the formula below for total control block space (*tcbs*) to calculate the number of pages. Then divide the result (*tcbs*) by the pages per cylinder value to obtain the number of cylinders for that type of DASD.

pages
is the minimum number of 4-KB pages of FBA DASD targeted to contain the object directory.

To calculate the minimum disk space required for an object directory, use the formula below for *tcbs* to calculate the number of pages.

ppc
is the number of pages per cylinder for the device type you intend to use for your object directory, as listed in Table 39 on page 648.

tcbs
is an estimate of the total number of pages required for directory control blocks, comprising the index pages, the profile specifications, the user specifications, and the identity specifications. The number can be found by using the following formula:

```
tcbs = 12 + (((512 + cpus x 16 + devices x 104 + iucvs x 32) / 4096) x users) + (identities
x 512)/4096
```

where

cpus
is the number of virtual CPUs defined for a typical virtual machine.

devices
is the number of virtual devices defined for a typical virtual machine, including those defined using SPOOL, CONSOLE, DEDICATE, MDISK, SPECIAL, and LINK statements. Include in the count for a user any devices that are defined in an imbedded profile or subconfiguration entry.

iucvs
is the number of IUCV statements defined for a typical virtual machine.

identities
is the number of virtual machines defined in the user directory using the IDENTITY statement.

users
is the number of virtual machines in the user directory, including those defined using USER, IDENTITY, SUBCONFIG, PROFILE, and POOL statements.

Example

For an installation with 5000 users, the numbers might look like:

users = 5000
cpus = 1
devices = 20
iucvs = 2

For these values, the number of pages of storage required would be:

```
tcbs = 12 + ((512 + 1 x 16 + 20 x 80 + 2 x 32) / 4096) * 5000

tcbs = 12 + 2676 = 2688

cylinders = 2688/180 x 2 = 15 x 2 = 30

pages = 2688 x 2 = 5336
```

For a 3390 at 180 pages per cylinder, the required space would be 30 cylinders. For an FBA device, the required space would be 5376 pages.

This is the minimum amount of space to be allocated. You should apply a growth factor to this amount to accommodate changes and additions to your system. Also, since this is only an approximation, if you make extensive use of directory statements not mentioned in the explanation of operands in the formula, you should apply a larger growth factor to the result.

Directory Size Constraints

The constraining factors on the number of users in the directory are as follows:

- The maximum size of the source directory that can be handled by the file editor. To maximize the effective size of the source directory, you can do the following:
  - Minimize the amount of virtual storage CMS is using by accessing as few disks as possible. This allows as much virtual storage as possible for the source directory.
  - Decrease the average number of records needed to describe a virtual machine by eliminating comments and by effective use of profiles.
  - Convert a very large monolithic source directory to cluster form.
- Disk space. The object directory must fit on one disk volume. With the size of disk supported by CP, this should not be a problem. The other limitations will come into play before this one is reached.

Paging Space

Attention: Underestimating your paging capacity may result in a system outage (PGT004 abend). Periodically use the QUERY ALLOC PAGE command to monitor paging space utilization and consider adding more paging space once your usage approaches 75% of the available space.

To determine the amount of paging space you will need to allocate on disk, you must consider the following characteristics of your installation:

1. Maximum total virtual machine size

- Consider the logon and maximum allowed storage sizes for each virtual machine. A virtual machine has the potential to instantiate all of the pages within its base address space plus any data spaces it is allowed to create.

2. Maximum total size of virtual disks in storage (VDISKs)
   - Consider both shared and private VDISKs and the number of pages that can be instantiated for each.
3. NSS/DCSS
   - Include the total number of shared pages within all NSS/DCSSs, plus the number of exclusive pages multiplied by the number of virtual machines loading them.
   - If any of your DCSSs are defined with the LOADNSHR option, you will need to allow for a separate copy of the DCSS for each expected non-shared usage of the DCSS.
4. System overhead
   - Include the number of CP object directory pages as reported by the DIRECTXA utility, plus an additional amount for future directory growth (possibly two times or more).
   - Allow approximately 1% of items 1, 2, and 3 for pageable PGMBKs.
   - The number of system virtual pages can vary based on your workload. Allow approximately 10% of your real storage size, with a maximum of 4G for system virtual pages.
5. System Growth
   - Any changes to the CP Directory to define more users, to increase virtual storage sizes for existing users, and to increase the number of allowable VDISKs must be monitored to ensure your existing paging capacity can handle the growth.
   - When operating within an SSI cluster, you must take into consideration the possibility for guests to relocate from other members of the cluster. Consideration should be given to the size of the guests to be relocated, including the size of VDISKs allocated to those guests.

Notes:

1. For CKD devices, to calculate the number of cylinders of that device type needed, divide the number of disk pages needed for paging space by the number of pages per cylinder for that device type, and round up.
2. For FBA devices to determine the number of blocks needed for paging space, multiply the number of pages by 8, since the number of 512-byte blocks in a page is 8.
3. To reduce paging slot allocations, you could consider specifying STORAGE AGELIST KEEPSLOT NO in the system configuration file, or on the SET AGELIST command, to cause paging slots to be unallocated when the corresponding page is read into memory. KEEPSLOT NO can reduce the number of slot allocations on paging volumes. However, it can also increase paging I/O during page-steal operations because every page needs to be written, even those that are unchanged since the last read from DASD.

Spooling Space

Spooling space is space on disk that CP requires to hold spool files on disk. When you allocate spool space, you should take these requirements into consideration:

- Normal spooling space
- The space required to write a dump
- The space required for a named saved system (NSS)
- The space required for system data files, such as:
  - Saved segments
  - UCR data files
  - Trace data files
  - Image libraries
  - NLS files.

Allocating Space for CP Hard Abend Dumps

It is recommended that you specifically allocate space for CP hard abend dumps, as described in this section. Table 42 on page 655 provides a recommended estimate of the dump space most customers will need. The estimates in Table 42 on page 655 are conservative; that is, most dumps will take up less space than what is indicated. For more specific details on estimating dump space, see "Additional Information about Dump Space Allocation" on page 656.

Use the DUMP option on the CP_OWNED system configuration file statement to reserve certain spool volumes to be used for dump space only. You should periodically check the amount of dump space allocated on your running system and make adjustments as necessary. For more information, including a list of CP commands you can use to query and modify dump space, see "Best Practices for Allocating Dump Space" on page 656.

Table 42. Amount of Dump Space for One Dump Based on Real Storage and Type of DASD

Real Storage Size	Percent of Real Storage that will be Dumped	4-KB Pages of Spool Space	3390 Cylinders	FBA 512-byte Blocks
256 MB	35 to 45%	23,000 to 29,500	127 to 164	184,000 to 236,000
512 MB	22 to 34%	28,900 to 44,600	160 to 248	231,000 to 357,000
1 GB	12 to 25%	31,500 to 65,600	175 to 364	252,000 to 525,000
2 GB	7 to 18%	36,700 to 94,400	204 to 524	294,000 to 755,000
4 GB	5 to 13%	52,500 to 137,000	291 to 757	420,000 to 1,100,000
8 GB	4 to 9%	83,900 to 189,000	466 to 1,050	672,000 to 1,510,000
16 GB	3 to 8%	126,000 to 336,000	699 to 1,870	1,010,000 to 2,690,000
32 GB	2 to 6%	168,000 to 504,000	932 to 2,800	1,350,000 to 4,030,000
64 GB	2 to 6%	336,000 to 1,010,000	1,870 to 5,600	2,690,000 to 8,060,000
128 GB	2 to 5%	672,000 to 1,680,000	3,730 to 9,330	5,370,000 to 13,500,000
256 GB	2 to 5%	1,350,000 to 3,360,000	7,460 to 18,700	10,800,000 to 26,900,000
512 GB	1.5 to 4%	2,020,000 to 5,370,000	11,200 to 29,900	16,200,000 to 43,000,000
1 TB	1.5 to 4%	4,030,000 to 10,800,000	22,400 to 59,700	32,300,000 to 85,900,000
2 TB	1.5 to 4%	8,060,000 to 21,500,000	44,800 to 120,000	64,500,000 to 172,000,000
4 TB	1.5 to 4%	16,120,000 to 43,000,000	89,600 to 240,000	129,000,000 to 344,000,000

If the amount of real storage you have is not listed in the table, use the value that is closest to your real storage amount and extrapolate the dump space you will need.

Dumps from various z/VM systems were analyzed and used to generate the estimates in Table 42 on page 655. Use the low percentage if you are estimating for a system with low memory utilization. Use the high percentage if you are estimating for a system with high memory utilization.

Information in this table is based on the following:

- Every 1 MB of data that must be dumped requires 256 pages of spool space.
- There are 180 four-kilobyte (4-KB) pages per 3390 cylinder.
- One 4-KB page is 8 FBA 512-byte blocks.

To illustrate how the values in the table were calculated, suppose your system has low memory utilization and has 100 GB of real storage.

```
100 GB x 1024 (MB per GB) x 2% x 256 (pages per MB) = 524288 pages

524288 pages / 180 (pages per 3390 cylinder) = 2913 cylinders

524288 pages x 8 (blocks per page) = 4194304 FBA 512-byte blocks
```

Best Practices for Allocating Dump Space

The following are best practices regarding dump space allocation:

- Use the DUMP option on the CP_OWNED system configuration file statement to reserve certain spool volumes to be used for dump space only. Also, fully allocate these DUMP volumes as SPOL space. These practices reduce spooling space fragmentation and will enable dump space allocation to find large clusters of available space.
- You should allocate space for more than one dump. Space for three dumps is recommended -- space for the next dump and space for two dumps that you may not have yet processed with the CP DUMPLOAD or DUMPLD2 utilities. It is recommended you purge dumps from spool after loading them to disk and successfully transmitting them to IBM Support, or when you are sure they are no longer needed.

Additional Hints and Tips

- When assigning devices for dump space, remember to consider that the speed of the device directly influences dump time.
- The following CP commands can be used to query and/or modify dump space:
  - The QUERY DUMP command provides information regarding the amount of spool (dump) space required for a dump of the system, given the status of the system at the time of the query. This command tells you how many pages are reserved on each spool volume. Note that a dump can span multiple spool volumes.
  - A privilege class B user can display the current dump setting with the QUERY DUMP command. A privilege class D user can display the current spool space status with the QUERY ALLOC SPOOL command. A privilege class D user can determine if any dump SPOOL files (spool class D) exist by issuing "CP QUERY RDR ALL CLASS D SHORTDATE".
  - A class B user can change the dump setting with the SET DUMP command. The SET DUMP command also allows the class B user to specify a preferred order of multiple disk volumes to be used for allocation of the dump space.
  - The SNAPDUMP and SET DUMP commands allow you to optionally include the CP frame table and user PGMBKs in a dump. Your calculations for dump size should assume these values are included when calculating the amount of dump space needed.

  For more information on these CP commands, see *z/VM: CP Commands and Utilities Reference*.

Additional Information about Dump Space Allocation

Table 42 on page 655 shows, for most z/VM systems, the amount of dump space that is required. However, because workloads can vary from system to system, the amount of spool space required for a dump also varies. If your installation requires a greater level of precision in allocating dump space, this section provides additional details to take into consideration.

CP hard abend dumps and CP snapdumps share spool space with other spool files. Dumps will be sent to spool space as long as enough spool space is available to contain the dump. The amount of spool space required for a dump is directly related to the amount of real storage used by the following:

1. The CP nucleus. The size of the CP nucleus is fairly constant but grows with each release of z/VM. Specifying the HCPLDR PAGEB operand, usually done only on test systems, can significantly increase the size of the CP nucleus. To find the size of the CP nucleus, look in the CPLOAD MAP file for the address of @LOADEND. For example, for z/VM 7.1, it is approximately X'E20000' if the HCPLDR

PAGEB operand is not specified. This value means that X'E20' four-kilobyte (4 KB) pages (or 3616 four-kilobyte pages) are used by the CP nucleus. That is about 14.1 MB.

2. The CP frame table. The CP frame table is directly proportional to the real storage size of the machine in which CP is executing. Its size is 1/128th the size of real storage. As the size of real storage increases, the size of the CP frame table becomes increasingly important to the size of the dump.

   Note: In systems with multiple noncontiguous storage extents, the CP frame table must map the highest real storage address. The CP frame table will contain entries for storage that is not configured. In this environment, the CP frame table is not proportionate to the amount of storage as it is for environments with contiguous online storage.
3. System execution space data structure. For systems with less than 4 GB, the system execution space page management table will account for a significant portion of the dump. Its size is 1/128th the size of real storage up to 2 GB. It is constant at 4096 pages for real storage sizes above 2 GB.
4. Calculation for PGMBKs. One control block to consider is the page management block (PGMBK). Many instances of this two-page control block can contribute to the size of the dump. A PGMBK exists for every megabyte of virtual storage with some contents. Therefore, while many of these PGMBKs are pageable, and can be in paging space, many of them are resident in memory and can contribute significantly to the size of the dump. This can vary widely depending on the amount of virtual storage in use by virtual machines and related virtual spaces. The more virtual storage in use on the system, the more PGMBKs in storage and in dump.
5. CP control blocks and structures based on your system's workload. During system initialization, CP tries to set aside enough spool space to contain a dump. This space changes dynamically as the system operates. As users log on, more CP control blocks are used and the pages required for dump space increases. Different types of production workloads could require significantly different CP control blocks and therefore have different requirements for dump space.

Named Saved System

To save a copy of an operating system running in a virtual machine, space must be reserved on a CP-owned volume for the named saved system. Named saved systems require one page of spool space for each page saved, plus an additional information page or pages. CP uses the information pages to save the virtual machine's register contents, PSW, and storage keys. The number of information pages required depends on the number of virtual machine pages being saved. The following formula shows this relationship:

```
information pages = 1 + virtual machine pages / 4096
```

Round up the result of this calculation to the next whole number.

For example, to save a 32 MB virtual machine, you need 8195 pages on disk (8192 virtual machine pages plus 3 information pages).

Spool Space Example

The following table shows you how much spooling space in files per pack you need to specify for CKD DASD based on the type of DASD you have and the average spool file size on your system. Three average spool file sizes are given: small (2 KB), medium (50 KB), and large (100 KB).

Device	Small (2 KB)	Medium (50 KB)	Large (100 KB)
3390-1	100,170	15,411	8,014
3390-2	200,340	30,822	16,028*
3390-3	300,510	46,233	24,042
3390-9	901,530	138,696	72,122

* Large work load (100 KB) and using a 3390-2

```
  2226  X   12  =  26712  X 15  =  400680 /  25  =   16,028  files/pack
   |         |                |                |
 Number of  Pages/Track     Tracks/Cyl.      Pages/File
 Cylinders
```

The table below shows you how much spooling space in files per pack you need to specify for FBA DASD based on the type of DASD you have and the work load running on your system. Three work load sizes are given: small (2 KB), medium (50 KB), and large (100 KB).

Device	Small (2 KB)	Medium (50 KB)	Large (100 KB)
9336-20	104,554	16,085**	8,364

** Medium work load (50 KB) and using a 9336-20

```
 1,672,881  /  8   = 209,110 - 2  =  209,108  / 13  =  16,085   files/pack
     |          |                |                |
Number of  Blocks/Page      Reserved Pages      Pages/File
Blocks
```

Switching Operating Modes for 3390 Devices

Attention: All mode switches require the affected device to be formatted, so you should move your data off the device before beginning this procedure.

To switch operating modes for a 3390 device, you will need to vary the device offline. If the device is defined to z/VM or is to be dynamically sensed, you may need to perform a SET RDEVICE so that CP recognizes the new device type. Otherwise, the integrity of the control blocks associated with the device is jeopardized.

If you have the devices defined in the system configuration file:

When the device is defined as a *3380* it is necessary to update the RDEVICE statement to define the device as a *3390*. Defined in this manner, z/VM will recognize the mode switch when the device is varied online. This should be done at a convenient time, such that the next IPL of the system will have the necessary definitions. Although it is not a requirement, for consistency, the device type can also be specified as 3390 in the I/O Configuration Dataset (IOCDS).

If you are switching from 3380 track compatibility mode to 3390 mode, and the device type is not defined as a *3390*, z/VM does not allow the device to be varied back online without first performing a SET RDEVICE to change the device type. If the device type is not also changed in the system configuration file, the device will remain offline and unusable after the next IPL.

If the devices are dynamically sensed:

No changes are required in the system configuration file. The SET RDEVICE command can be used to change the device definition for the current mode of the 3390 device. During IPL, the 3390 device will be brought online in its current mode of operation.

Remember that the 3390 Model 9 does not support 3380 track compatibility mode, so do not perform this procedure with this device.

Mode Switch Procedure

Use the following procedure for switching 3390 operating modes:

1. Prevent all applications and users from using the device. For example, if the device is being used by the system for minidisks, all users must detach those minidisks, and the device must be detached from the system. If the device is being used by one guest, that guest must detach the device.
2. Move the data off the affected device. The procedure of switching modes destroys the data on the device and changes the track format.

If you wish to restore the data to the same device after switching modes, you should not use service programs that require identical track formats (such as DASD Dump Restore). For information on moving data, see *Using IBM 3390 Direct Access Storage in a VM Environment*, GC26-4575.

3. Vary the device offline to all processors that have access to it, except the one from which you perform the mode switch.
4. Attach the device to the virtual machine of the person who is performing the mode switch; typically, this person is the system programmer. The virtual machine to which the device will be attached must have Class F authority.
5. Use the *SETMODE* parameter of the ICKDSF INSTALL command to change the mode. For information about using ICKDSF for this operation, see *Using IBM 3390 Direct Access Storage in a VM Environment*, GC26-4575.
6. Reformat the affected device when all mode switches are made, being sure that you have moved the data off the device before you begin.

   You should use ICKDSF for formatting and allocating disks rather than CPFMTXA, which has been used for those purposes in the past. For information on using this option, see the *Device Support Facilities User's Guide and Reference*, GC35-0033.

   Any CMS minidisks on the device also must be formatted using CMS FORMAT.
7. Detach the device from the virtual machine of the person who performed the mode switch.
8. If the mode switch is:

   - For a device defined in the system configuration file:
     - **From** 3380 track compatibility mode **to** 3390 mode and the device type is defined as a 3380, you must redefine the device as a 3390 by first varying the device offline and then entering ***either*** of the following commands:

       ```
       set rdevice rdev type dasd
       set rdevice rdev clear
       ```

       Then vary the device back online. You must also change the definition of this device in the system configuration file.
     - **From** 3380 track compatibility mode **to** 3390 mode and the device type is defined as a 3390, vary the device offline and then back online.
     - **From** 3390 mode **to** 3380 track compatibility mode, vary the device offline and then back online.
   - For a dynamically sensed device:
     - **From** 3380 track compatibility mode **to** 3390 mode or **from** 3390 mode **to** 3380 track compatibility mode, you must redefine the device by varying the device offline and entering ***either*** of the following commands:

       ```
       set rdevice rdev type dasd
       set rdevice rdev clear
       ```

       Then vary the device back online.

   For information on reformatting the volume for a particular use (CP and CMS minidisks), and moving CP-owned volumes and CMS minidisks, see the *Using IBM 3390 Direct Access Storage in a VM Environment*, GC26-4575.

   Remember that the device type you use when restoring data is different from the one you used to back up the data in Step "2" on page 658.
9. After the data is restored, attach the device again to the system or guest and allow applications to continue.

Migration Considerations for the 3390 Model 9

The z/VM support of 3390 Model 9 will not affect migration, but there are data migration considerations if you plan to move data from other devices to a 3390 Model 9.

The 3390 Model 9 is formatted as a CKD device and is supported for formatting by ICKDSF and the CPFMTXA utility. The allocation map format on a 3390 Model 9 device will be an extent-based allocation map as opposed to the cylinder-based allocation map currently used for existing CKD DASD. This is because the existing 4K cylinder-based allocation record is not large enough to account for the number of cylinders on a 3390 Model 9 device.

Because the DDR command is capable of doing a straight copy from an existing 3390 device to a 3390 Model 9 device, this may result in a Model 9 device having a cylinder-based allocation map record. The 3390 Model 9 device will still be usable by z/VM, but because of the limitation of the cylinder-based allocation map record, only the first 4K cylinders of the device will be capable of being used.

The reverse situation is also possible where a 3390 Model 1, 2 or 3 device could have an extent-based allocation map. This, however, will not be allowed by CP. If a 3390 contains an extent-based allocation map and an operator attempts to attach this 3390 Model 1, 2 or 3 as a CP volume, a warning message will be issued.

If either type of data movement is performed, the output disk should be reallocated using the ICKDSF CPVOLUME ALLOCATE command if the device is intended to be used as a CP volume. ICKDSF will convert a cylinder-based allocation map record on a 3390 Model 9 device into the extent-based allocation record that is necessary to use the entire device. An extent-based allocation record found on a 3390 model 1, 2 or 3 can also be converted by ICKDSF into a cylinder-based allocation map record.

The following coexistence considerations should also be noted:

- 3390 Model 9 devices can coexist in the same string with other 3390 devices, but cannot coexist in the same string with other 3380 devices.
- 3390 Model 9 B-units (B9C) cannot be intermixed in a string with other models of the 3390.

Adding DASD Space to a Running System

You can dynamically add CP-owned volumes to your running z/VM system. In order to do so, you must have CP_OWNED statements with slots specified as RESERVED in your system configuration file. You should always have RESERVED slots for this purpose. Up to 255 CP-owned slots can be defined, and you can specify all unused slots as RESERVED with a single statement:

```
        CP_OWNED       Slot 255       RESERVED
```

Adding new volumes in RESERVED slots does not disturb the relative entry number (slot number) that is used to locate existing spool files. This means you can warm start the system after the new volumes have been added.

For information about the CP_OWNED statement, see "CP_OWNED Statement" on page 73.

Attention: If you delete any volume that contains spool space from the CP-owned volume list, or move any volume that contains spool space to a different slot in the CP-owned volume list, a clean start is required. A clean start causes the deletion of spool files and system data files, including named saved systems and saved segments. Files that need to be preserved over such a change must be dumped to tape using the SPXTAPE DUMP command.

After the clean start, SPXTAPE LOAD must be used to restore the spool files and system data files from tape. This ensures that spool files and system data files that existed on the system before deletion or movement of volumes containing spool space are restored correctly.

If you are running an SSI cluster, issue the QUERY SSI command to confirm that it is in STABLE mode before beginning this procedure. If it is not in STABLE mode, *do not* begin this procedure until the cluster has returned to STABLE mode. In addition, complete this entire procedure before IPLing any other SSI cluster members. If you decide to abort the process after you have started it, you should undo what was done in the steps that you successfully completed.

Use the following steps to add a DASD volume in a RESERVED slot:

1. Issue the QUERY CPOWNED command to determine which slots are available (defined as RESERVED). Select the slot where the volume will be added.

   If you are running an SSI cluster, it is good practice to group slots of the same allocation type from the same SSI member together.
2. Locate an available DASD volume.
3. ATTACH the volume to your z/VM user ID.
4. Use the CPFMTXA utility to format the entire DASD volume, and assign a label (for example, NEWSPL).

   Attention: Volume labels must be unique across the SSI cluster.

   Allocate the volume as follows:

   ```
   PERM                      0 0
   (type of CP space)        1 END

   Type of CP space - PAGE, SPOL, TDISK, etc.
   ```

   If you are running an SSI cluster, you must also use the OWNER option in CPFMTXA to assign an SSI member as owner of the volume. The OWNER specified may be any defined SSI member that is part of the SSI cluster. In other words, OWNER does not have to be the SSI member on which CPFMTXA was executed.
5. DETACH the volume from your z/VM user ID.
6. Update your system configuration file (usually located on MAINT's CF1 disk) by adding or updating a CP_OWNED statement to assign the new volume to the selected RESERVED slot. This ensures that the new volume will be available to the system at the next IPL. For example:

   ```
   CP_OWNED          Slot 25          RESERVED
              is changed to:
   CP_OWNED          Slot 25          NEWSPL
   ```

   It is recommended that you use the RDEV operand on the CP_OWNED statement. If you have multiple volumes with the same label, the RDEV operand insures that you are using the correct volume.
7. Run the CPSYNTAX utility against the updated system configuration file to check for errors.
8. Issue the DEFINE CPOWNED command to define the new volume NEWSPL to your active system:

   ```
   DEFINE CPOWNED Slot 25 NEWSPL
   ```

   This command will be propagated to all active members of the SSI cluster.
9. If you are running in an SSI cluster, make sure that the volume is online and not already attached to all active SSI cluster members.
10. Issue the ATTACH command from the system on which the previous steps were performed:

    ```
    ATTACH rdev SYSTEM
    ```

    CP will begin using the space allocated on the new volume.

    Note: After the ATTACH command is issued in an SSI cluster, on the OPERATOR's console of the other SSI members, response "DASD rdev ATTACHED TO SYSTEM NEWSPL BY SYSTEM" is displayed. When a volume with SPOOL extents is defined as CP-owned on a member of the cluster and the ATTACH command is issued, the attach process will automatically attach the volume to the other active members where the device is online and not already attached. Also, if the volume is not in the CP-owned list on another active member where the device is being attached, the attach process will automatically add the volume to the CP-owned list on that member. As an extra check, issue the

QUERY CPOWNED command on all active members of the cluster to make sure that the new volume is shown in the correct slot. If it is not, review the steps above to make sure they were done correctly.

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

Chapter 24. How the CP-Owned List Is Processed

When a z/VM system is initialized, the list of CP-owned DASD volumes for that system, as specified on CP_OWNED statements in the system configuration file, is processed to determine:

- Which directory (DRCT), paging (PAGE), spooling (SPOL), and temporary disk (TDSK) allocations and checkpoint and warm start areas on each volume to bring online
- Whether the volume is owned by that system or shared with another system

After initialization, the DEFINE CPOWNED and ATTACH commands can be used to add a volume to the CP-owned list for the current session, and such a volume is processed in the same manner when attached.

On an Extended Address Volume (EAV), DRCT, and SPOL allocations must be below cylinder 65520, and PAGE allocations must be below cylinder 1182006. Checkpoint and warm start areas must be below cylinder 65520. TDSK allocations can be anywhere on an EAV.

Note: If the OWN or SHARED operand is specified on a CP_OWNED system configuration statement or a DEFINE CPOWNED command, the operand is ignored.

In a z/VM Single System Image (SSI) cluster, each CP-owned volume is marked with ownership information to ensure that CP on one member of the SSI cluster cannot define allocations and areas on a volume owned by another member. The ownership information consists of the name of the SSI cluster and the system name of the member, if any, that owns the volume. Ownership information is recorded on the volume by using the OWNER operand of the CPFMTXA utility. For information about how to specify the ownership information, see the CPFMTXA utility in *z/VM: CP Commands and Utilities Reference*. For additional information about SSI clusters, see Chapter 29, “Setting Up z/VM Single System Image Clusters,” on page 715.

In non-SSI configurations, ownership information may be recorded on CP-owned volumes but is not required. However, when a non-SSI z/VM system is installed, ownership information consisting of the system name and no cluster name is recorded on each CP-owned volume.

Standard DASD copy utilities will include the ownership information in the copy.

The processing of the CP-owned list includes any ownership information recorded on the volumes. The results depend on whether the system is a member of an SSI cluster. If no allocations or areas can be brought online for a CP-owned volume due to its ownership information, CP takes one of the following actions:

- If there is PERM space on the volume, the volume is attached to the system with the PERM space available for MDISK usage. Message HCP6631I is issued.
- If there is no PERM space on the volume, the volume is not attached to system, and remains a free device. Message HCP6629I is issued.

Note: For an EAV, PERM space can be located anywhere on the volume.

Marking ownership information on user volumes is optional. If you wish to mark your user volumes, use the CPFMTXA utility to format cylinder 0, allocate them as PERM 0-END, and mark their ownership appropriately. If the volume is to be used in an SSI cluster, it should be marked either with the SSI cluster name and no system name (in the case of a volume shared among cluster members) or with the SSI cluster name and the system name of a member (in the case of a volume holding member-specific (local) minidisks). If the volume is to be used in a non-SSI configuration, it should be marked with no cluster name (NOSSI) and the name of the owning system.

Processing for a Member of an SSI Cluster

When the CP-owned list is processed for a member of an SSI cluster, the volume allocations and areas are brought online as shown in Table 43 on page 664.

Note: If the system has been IPLed in REPAIR mode, this table does not apply. See "Processing for a System IPLed in REPAIR Mode" on page 665.

Table 43. CP-Owned Volume Allocations and Areas Brought Online for a Member of an SSI Cluster

SSI Cluster Name on Volume	System Name on Volume	SPOL Extents (Owner or Shared)	DRCT, PAGE, and TDSK Extents and Checkpoint and Warm Start Areas (Nonshared)
None	None	No	No
None	Name of this member	Yes (owner, single-member cluster only)	Yes
None	Not the name of this member	No	No
Name of this cluster	None	No	No
Name of this cluster	Name of this member	Yes (owner)	Yes
Name of this cluster	Name of another member	Yes (shared)	No
Name of this cluster	Not the name of a member (probable configuration error)	No	No
Not the name of this cluster	Any value	No	No

The following rules are enforced for the members of an SSI cluster:

- Spooling (SPOL) extents

  Spooling space can be owned by only one member and must be shared by the other members. If the cluster name and system name recorded on the volume match the values specified in the system configuration file for the member being initialized, the SPOL extents are brought online as owned (this member owns the spooling space). If another member in this SSI cluster owns the volume, the SPOL extents will be brought online as shared (this member is sharing the spooling space). Otherwise, the SPOL extents will not be brought online.

  A shared spooling volume must have the same slot number in each member of the cluster.
- Paging (PAGE) and temporary disk (TDSK) extents

  Multiple members cannot simultaneously use the PAGE or TDSK extents on a volume. Only the owning member can bring the PAGE and TDSK extents online.
- Checkpoint and warm start areas

  A member's checkpoint and warm start areas cannot be on the same volume as another member's checkpoint and warm start areas. If a member attempts to use a checkpoint or warm start area on a volume that it does not own, the member will enter wait state 6626.

Processing for a System That Is Not a Member of an SSI Cluster

When the CP-owned list is processed for a system that is not a member of an SSI cluster, the volume allocations and areas are brought online as shown in Table 44 on page 665.

Note: If the system has been IPLed in REPAIR mode, this table does not apply. See "Processing for a System IPLed in REPAIR Mode" on page 665.

Table 44. CP-Owned Volume Allocations and Areas Brought Online for a System That Is Not a Member of an SSI Cluster

SSI Cluster Name on Volume	System Name on Volume	SPOL Extents (Owner)	DRCT, PAGE, and TDSK Extents and Checkpoint and Warm Start Areas (Nonshared)
None	None	Yes	Yes
None	Name of this system	Yes	Yes
None	Not the name of this system	No	No
Any value	Any value	No	No

Processing for a System IPLed in REPAIR Mode

Using the Stand-Alone Program Loader (SAPL), you can IPL a z/VM system with the REPAIR parameter to bypass certain processing. For more information about the REPAIR parameter, see Passing IPL Parameters in *z/VM: System Operation*.

When the CP-owned list is processed for a system IPLed in REPAIR mode, the volume allocations and areas brought online are determined by the system name recorded on the volume and whether the system configuration file contains an SSI statement. Any SSI cluster name recorded on the volume is ignored.

If the system configuration file contains an SSI statement (valid or invalid), the volume allocations and areas are brought online as shown in Table 45 on page 665.

Table 45. CP-Owned Volume Allocations and Areas Brought Online for an SSI Cluster Member IPLed in REPAIR Mode

Cluster Name on Volume	System Name on Volume	SPOL Extents (Owner)	DRCT, PAGE, and TDSK Extents and Checkpoint and Warm Start Areas (Nonshared)
Ignored	None	No	No
Ignored	Name of this system	Yes	Yes
Ignored	Not the name of this system	No	No

If the system configuration file does not contain an SSI statement, the volume allocations and areas are brought online as shown in Table 46 on page 665.

Table 46. CP-Owned Volume Allocations and Areas Brought Online for a Non-SSI System IPLed in REPAIR Mode

SSI Cluster Name on Volume	System ID on Volume	SPOL Extents (Owner)	DRCT, PAGE, and TDSK Extents and Checkpoint and Warm Start Areas (Nonshared)
Ignored	None	Yes	Yes
Ignored	Name of this system	Yes	Yes

Table 46. CP-Owned Volume Allocations and Areas Brought Online for a Non-SSI System IPLed in REPAIR Mode (continued)

SSI Cluster Name on Volume	System ID on Volume	SPOL Extents (Owner)	DRCT, PAGE, and TDSK Extents and Checkpoint and Warm Start Areas (Nonshared)
Ignored	Not the name of this system	No	No

Chapter 25. DASD Sharing

Under z/VM, you can share information on a DASD volume among:

- Multiple virtual machines.
- One virtual machine and operating systems running on other processors.
- Multiple virtual machines and operating systems running on other processors.
- Multiple systems in a z/VM single system image (SSI) cluster. For more information on this, see Chapter 29, “Setting Up z/VM Single System Image Clusters,” on page 715.
- Multiple non-SSI systems using cross-system link (XLINK). For more information on this, see Appendix C, “Using Cross-System Link (XLINK),” on page 843.

Note that the guest operating system or application must provide the protection by using reserve/release. CP only grants access to the DASD and handles virtual reserve/release for MDISK and LINK definitions.

This chapter tells you how to share DASD under various circumstances:

- Among multiple virtual machines by using reserve/release
- Without using virtual reserve/release
- Using the CMS Shared File System
- Between a virtual machine and other systems
- Among multiple virtual machines and other systems

This chapter also includes information on the following topics:

- Restrictions for using reserve/release
- Using the Multi-Path Lock Facility (MPLF)
- Using cached DASD as high-speed storage devices
- Using IBM Parallel Access Volumes
- Using IBM HyperParallel Access Volumes
- Using Persistent FlashCopy
- Space-Efficient Volumes

Sharing DASD among Multiple Virtual Machines by Using Virtual Reserve/Release

If you need to share data among virtual machines running under VM but do not need to share it with operating systems running on other processors, the following method of sharing DASD gives you the best results for non-CMS guests. This method is called virtual reserve/release.

Sharing data between virtual machines involves the use of minidisks. A minidisk is a virtual DASD for a specific virtual machine. This virtual DASD is mapped to an extent of cylinders on a real DASD. CP can manage DASD sharing in full-pack minidisks, including DEVNO minidisks, and on smaller minidisks.

Note: Virtual reserve/release is also supported for virtual disks in storage, which are minidisks allocated from host storage rather than mapped to real DASD.

To share a minidisk among virtual machines, you must do the following:

1. Code the MDISK directory statement in the virtual machine definition. For example, the following MDISK directory statement for a virtual machine (VM1):

   ```
   MDISK 197 3390 001 100 WORKPK MWV VM1PASS
   ```

   - Defines a minidisk at VM1's 197 virtual device number

- Specifies that this minidisk is to be on the 3390 DASD with the volume ID WORKPK
- Specifies that this minidisk starts at cylinder 001 and encompasses 100 cylinders. Ensure that no other MDISK directory statement is defined to use any of these same cylinders. DASD RESERVE/RELEASE is not intended to control sharing of overlapping minidisks.
- Includes an access mode of MW (multi-write) with a V (indicating that this minidisk can be shared between virtual machines) and a password of VM1PASS.

2. Code the LINK directory statement in another virtual machine's directory. The LINK directory statement and the CP LINK command each allow virtual machines to access another user's minidisk. Suppose a second virtual machine (VM2) has this LINK directory statement in its directory:

   ```
   LINK VM1 197 197 MW
   ```

   This statement links VM2 to VM1's minidisk. MW indicates that VM2 requests multi-write access to the minidisk. (The MW access mode on the MDISK directory statement permits VM2 to obtain write access to this minidisk at the same time that VM1 has write access to it.)

   If VM2 does not have this LINK statement in its directory, it can link to VM1's minidisk with the CP LINK command. For example, entering:

   ```
   link vm1 197 197 mw vm1pass
   ```

   will perform this link. Or you can use the CMS VMLINK command to link and access the disk. VMLINK finds a free virtual device number and access mode letter:

   ```
   vmlink vm1 197 <= = mw> (noname pw vm1pass
   ```

   With the LINK and VMLINK commands, VM2 needed to specify the appropriate password (VM1PASS in this example) to be granted access. Using minidisk passwords helps you to protect shared data, because only those users who know the correct password can link to the minidisk with the CP LINK command and the CMS VMLINK command.

   Note: All DASD that do not respond to a sense ID request must be defined in the system configuration file using RDEVICE statements. All DASD that hold minidisks must be attached to SYSTEM, either at system initialization by appearing in the system configuration file using CP_OWNED or USER_VOLUME_LIST statements, or later by issuing the command ATTACH *rdev* TO SYSTEM.

   For more information about these system configuration file statements, see "RDEVICE Statement" on page 227, "CP_OWNED Statement" on page 73, and "USER_VOLUME_LIST Statement" on page 316.

3. Specify that the DASD will not be shared with another operating system. The default setting of the SHARED operand of the RDEVICE system configuration file statement (SHARED NO) takes care of this for you. A NO setting means this volume cannot be shared with an operating system running on another processor. You can also use the CP SET SHARED command to accomplish this. For example, if you enter:

   ```
   set shared off for 197
   ```

   the device at real device number 197 cannot be shared with other operating systems. However, you should enter SET SHARED OFF only after careful consideration. See *z/VM: CP Commands and Utilities Reference* for more information about the SET SHARED OFF command.

   Note: This method is for sharing a minidisk between virtual systems on the same real system—not for sharing with another real system. Therefore, set the SHARED operand to NO (the default). Setting the SHARED operand to YES applies only to sharing full-pack minidisks between multiple real systems. Sharing is managed by CP simulation of the guest reserve/release request through the MDISK block. Note that to invoke this support, the access mode must include V (for example, MWV for shared multiwrite mode). For information on sharing a dedicated device with another system running native on another processor, see "Sharing DASD between One Virtual Machine and Other Systems Using Real Reserve/Release" on page 670. For information on sharing a full-pack minidisk among multiple virtual systems and other systems running native, see "Sharing DASD among Multiple Virtual Machines and Other Systems Using Concurrent Virtual and Real Reserve/Release" on page 672.

Figure 17 on page 669 shows three virtual machines sharing a minidisk defined on a DASD through virtual reserve/release.

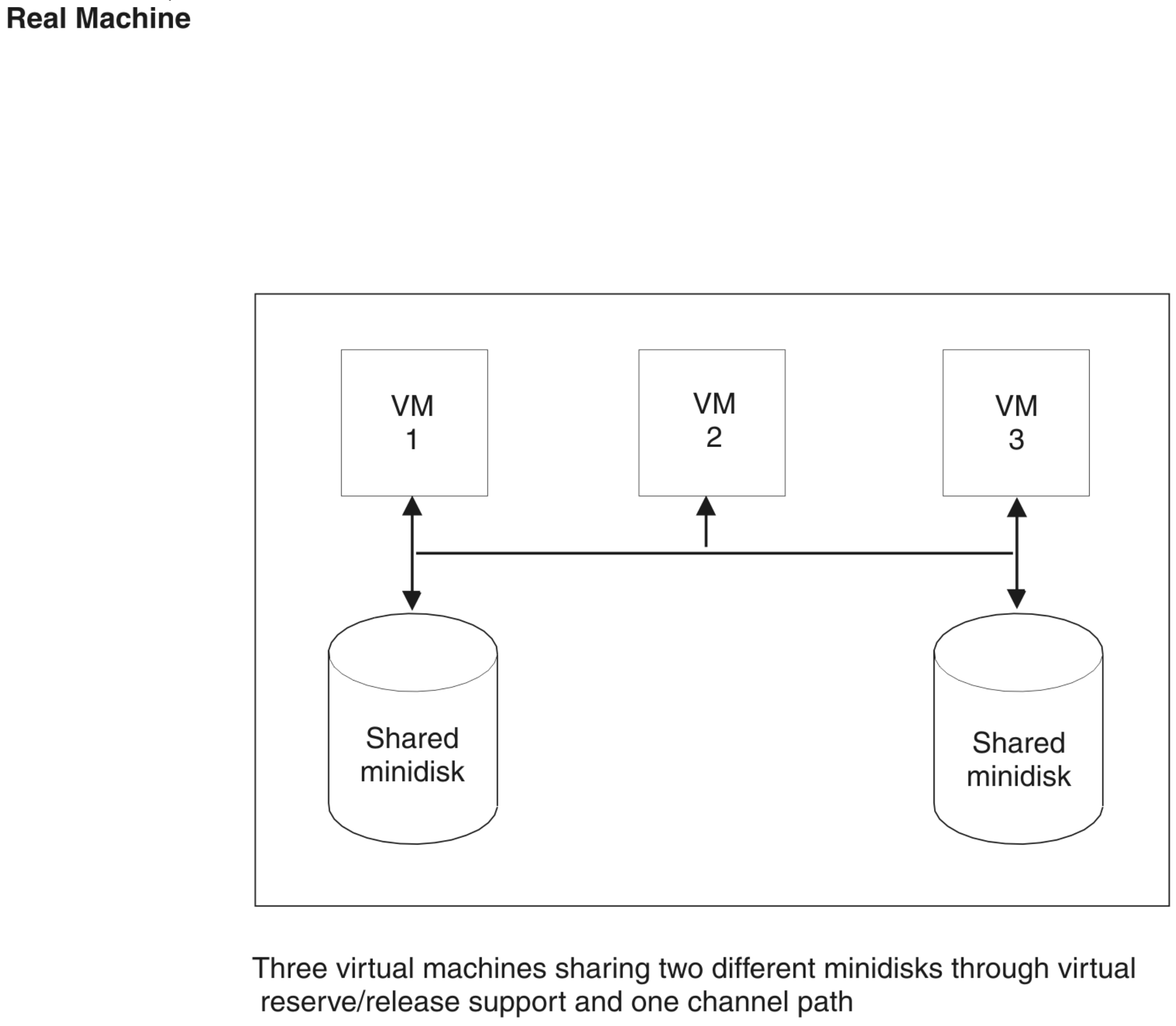

Real Machine
VM 1
VM 2
VM 3
Shared minidisk
Shared minidisk
Three virtual machines sharing two different minidisks through virtual reserve/release support and one channel path

Figure 17. Virtual Reserve/Release

When to Use Virtual Reserve/Release

In general, you should use virtual reserve/release when you need to share DASD among virtual machines running under VM but do not need to share the DASD with other systems and the guest operating system is not CMS. If the operating system is CMS, use the Shared File System. Note that the operating system running in a virtual machine must support reserve/release CCWs for you to use virtual reserve/release.

Note: The *BLOCKIO IUCV system service and DIAGNOSE code X'250' do not honor DASD reserves managed by VM's virtual reserve/release function. Therefore, do not use these I/O interfaces if a minidisk is shared by multiple guests on the same VM image where another guest is expecting to use reserve/release I/O to serialize its data access. Also, *BLOCKIO and DIAGNOSE code X'250' do not use reserve/release. Therefore, do not use these interfaces for any DASD (CP-attached or full-pack minidisk) that is shared with other LPARs where another LPAR expects to use reserve/release to serialize its data.

Sharing DASD without Using Virtual Reserve/Release

If the guest operating systems do not support reserve/release CCWs, you can still share DASD between virtual machines. However, no write protection is provided using this method. To do this, define a minidisk for one virtual machine and have other virtual machines link to that minidisk. The process is similar to that described for virtual reserve/release with the following exceptions:

1. You do not have to specify a V as part of the access mode on the MDISK directory statement.
2. You do not have to be concerned with the SHARED setting.

The major difference between this method of sharing DASD and virtual reserve/release is best illustrated by the case where you have included an access mode of MW (multiple write) on the MDISK definition. With virtual reserve/release, although many virtual machines can obtain write access to the minidisk, only one virtual machine can write to it at any one time if it uses virtual reserve release. If you do not use virtual reserve/release, two virtual machines can both obtain write access to the minidisk at the same time. To ensure data integrity, you should use virtual reserve/release when you need to give multiple virtual machines write access to a minidisk.

Note: Sharing DASD without using virtual reserve/release is not recommended. You could read incomplete data or have an abend.

The SET WRKALLEG command supports the MVS Sysplex Environment. SET WRKALLEG allows MVS guests running in a single virtual machine to share minidisks without using reserve and release CCWs. For more information, see the SET command in *z/VM: CP Commands and Utilities Reference*.

Sharing DASD Using the CMS Shared File System

The CMS Shared File System allows users to organize their files into groups called directories and selectively share those files and directories with other users. To do this, a collection of minidisks is assigned to a single virtual machine called a file pool server machine. The collection of minidisks is called a file pool. A file pool contains the files for many users, not just one. The file pool server virtual machine manages all the files in the file pool.

For additional information on the Shared File System, see *z/VM: CMS File Pool Planning, Administration, and Operation* and the *z/VM: CMS User's Guide*.

Sharing DASD between One Virtual Machine and Other Systems Using Real Reserve/Release

This method of sharing DASD with other systems, called real reserve/release, involves the use of dedicated devices. A dedicated device is exclusively owned by a single virtual machine. For example, suppose you want VM1 to have sole ownership of the 3390 DASD with volume ID SYSPK. Assume that this 3390 was defined at real device number 872 in the RDEVICE system configuration file statement. The DEDICATE directory statement gives a virtual machine exclusive use of a device. For example, the statement:

```
DEDICATE 872 872
```

in VM1's directory gives VM1 ownership of the device at real device number 872, and defines the DASD at VM1's 872 virtual device number. (It is generally a good idea to keep real and virtual device numbers the same.)

You can also use the ATTACH command to dedicate a device to a virtual machine. The command:

```
attach 872 to vm1 as 872
```

is equivalent to the previous DEDICATE directory statement.

Then define DASD that are accessible from more than one processor as shared in the system configuration file. (Define the DASD as shareable to CP by coding SHARED YES in the system configuration file.) Hardware and software are then prepared for the sharing environment and reserve/release commands are handled by the control unit.

VM1 can now share this dedicated DASD with another system, as shown in Figure 18 on page 671.

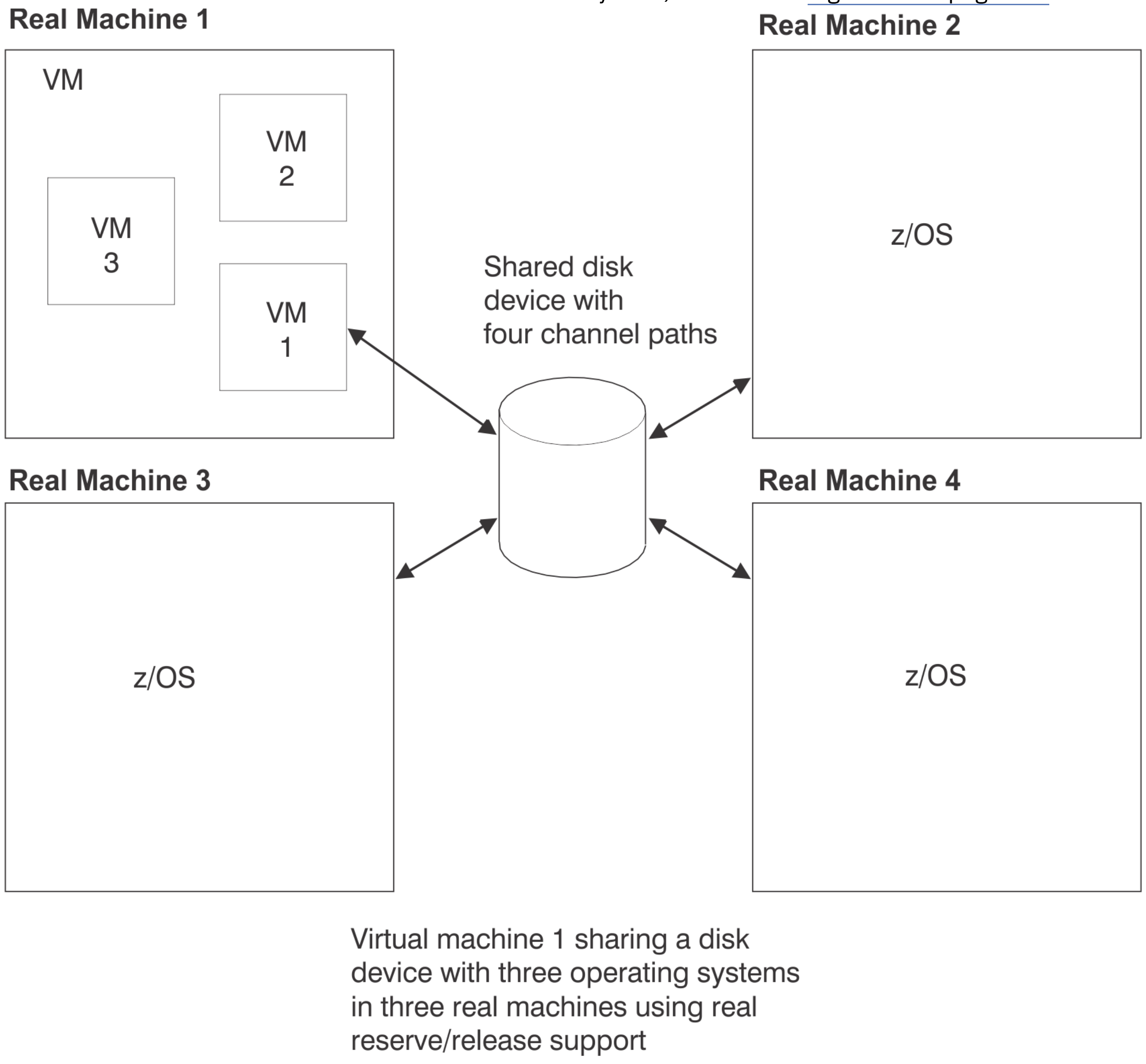

Real Machine 1
VM
VM 2
VM 3
VM 1
Shared disk device with four channel paths
Real Machine 2
z/OS
Real Machine 3
z/OS
Real Machine 4
z/OS
Virtual machine 1 sharing a disk device with three operating systems in three real machines using real reserve/release support

Figure 18. Real Reserve/Release

When to Use Real Reserve/Release

In general, you should use real reserve/release when you need to share a DASD among one virtual machine and other systems. Note, a dedicated DASD cannot be shared among virtual machines but can be shared among one virtual machine and other systems.

Note: The *BLOCKIO IUCV system service and DIAGNOSE code X'250' do not honor DASD reserves managed by VM's virtual reserve/release function. Therefore, do not use these I/O interfaces if a minidisk is shared by multiple guests on the same VM image where another guest is expecting to use reserve/release I/O to serialize its data access. Also, *BLOCKIO and DIAGNOSE code X'250' do not use reserve/release. Therefore, do not use these interfaces for any DASD (CP-attached or full-pack minidisk) that is shared with other LPARs where another LPAR expects to use reserve/release to serialize its data.

Sharing DASD among Multiple Virtual Machines and Other Systems Using Concurrent Virtual and Real Reserve/Release

The third method of sharing DASD, concurrent virtual and real reserve/release, combines virtual reserve/release and real reserve/release. It allows DASD to be shared among multiple virtual machines and operating systems running on other processors. Figure 19 on page 672 shows VM sharing a DASD between two virtual machines and a native system.

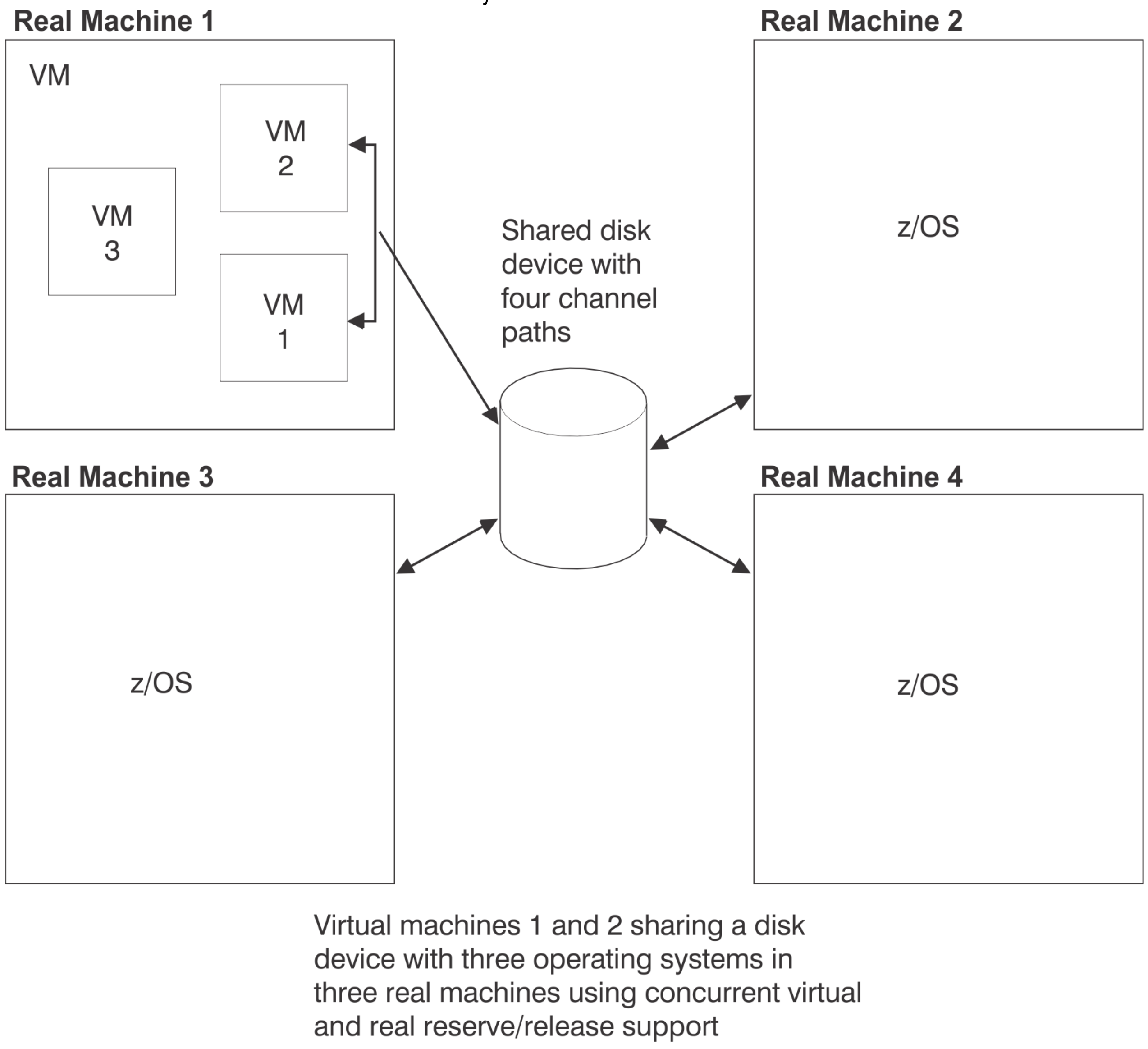

Real Machine 1
VM
VM 2
VM 3
VM 1
Shared disk device with four channel paths
Real Machine 2
z/OS
Real Machine 3
z/OS
Real Machine 4
z/OS
Virtual machines 1 and 2 sharing a disk device with three operating systems in three real machines using concurrent virtual and real reserve/release support

Figure 19. Concurrent Virtual and Real Reserve/Release

Concurrent virtual and real reserve/release involves the use of full-pack minidisks. A full-pack minidisk is a single minidisk allocated on an entire DASD volume. A full-pack minidisk encompasses all the primary (or addressable) tracks of the volume. In the example under "Sharing DASD among Multiple Virtual Machines by Using Virtual Reserve/Release" on page 667, VM1 defined a minidisk on a portion of the volume called WORKPK. You need to do the following to use concurrent virtual and real reserve/release:

1. Define the DASD as a shareable full-pack minidisk. The following MDISK directory statement defines a full-pack minidisk:

```
MDISK 199 3390 000 END SYSPK MWV FULLP1
```

This statement allocates all addressable cylinders of the 3390 called SYSPK as minidisk space, thus making SYSPK a full-pack minidisk. (This method is the preferred way of defining a full-pack minidisk because the number of cylinders does not need to be known when writing the MDISK statement.) The V at the end of the statement indicates that this DASD can be shared between virtual machines. To access the DASD, another virtual machine must use the LINK directory statement or the LINK command. The procedure is the same as described for virtual reserve/release under "Sharing DASD among Multiple Virtual Machines by Using Virtual Reserve/Release" on page 667.

Alternatively, one could use a DEVNO statement as in this example:

```
MDISK 199 3390 DEVNO 1234 MWV FULLP1
```

Multiple DEVNO or MDISK 0-END statements will work for the same real volume, but it is preferable to have a single owning MDISK statement with other guests on the same VM system using LINK to the owning MDISK. This applies only to full-pack volume minidisks. Data integrity issues will arise if multiple MDISK statements are used with non-full-pack minidisks. Non-full-pack minidisks must use a single owning MDISK statements with other guests using LINK statements.

2. Define the DASD as shareable to CP by coding SHARED YES in the system configuration file. As an alternative, you can use the CP SET SHARED ON command. For example, if you enter:

   ```
   set shared on for 199
   ```

   CP recognizes the full-pack minidisk at real device number 199 as a shared DASD.

When to Use Concurrent Virtual and Real Reserve/Release

In general, you should use concurrent virtual and real reserve/release when you need to share DASD among many virtual machines and other systems. Do not use this method when you need to share DASD only among virtual machines, because the CP overhead is much greater than if you use virtual reserve/release.

Note: The *BLOCKIO IUCV system service and DIAGNOSE code X'250' do not honor DASD reserves managed by VM's virtual reserve/release function. Therefore, do not use these I/O interfaces if a minidisk is shared by multiple guests on the same VM image where another guest is expecting to use reserve/release I/O to serialize its data access. Also, *BLOCKIO and DIAGNOSE code X'250' do not use reserve/release. Therefore, do not use these interfaces for any DASD (CP-attached or full-pack minidisk) that is shared with other LPARs where another LPAR expects to use reserve/release to serialize its data.

Restrictions for Using Reserve/Release

To use real reserve/release or concurrent virtual and real reserve/release, all of the following must support reserve/release CCWs:

- The operating system running under VM
- The system with which the guest will be sharing DASD
- The shared DASD

Virtual reserve/release must be enabled for any guest virtual machine reserve to be accepted, even if only one guest is using the device. Virtual reserve/release allows reserve/release CCWs to be issued by the guest. Real reserve/release allows CCWs to reach the device. CP simulates reserve/release if virtual reserve/release is enabled and real reserve/release is not. CP issues the reserve/release CCWs if both are enabled.

Note, concurrent virtual and real reserve/release support should be used only when DASD must be shared among many virtual machines and operating systems on other real machines. When DASD must be shared only among virtual machines, virtual reserve/release support should be used because it requires much less CP overhead.

For information on how link conflicts are handled when virtual reserve/release is authorized, see the LINK command in *z/VM: CP Commands and Utilities Reference*.

With one exception, the unconditional reserve CCW should not be used by any system sharing a DASD due to potential data integrity problems. The one exception is when it is known that the system holding the reserve is down and will be re-ipled.

Reserve/Release Summary

Table 47 on page 674 summarizes the differences between different settings of SET SHARED and the V attribute of the MDISK directory statement.

Table 47. Reserve/Release Summary

SET SHARED	V Attribute on MDISK	Description
ON	No V attribute	The minidisk cannot be shared between a virtual machine guest and a native system. There is no virtual reserve/ release.
ON	V attribute	Both real and virtual reserve/release are available. The minidisk can be shared between virtual machine guests and a native system.
OFF	V attribute	Only virtual reserve/release is available; no real reserves are issued. The minidisk cannot be shared with a native system.
OFF	No V attribute	No reserve/release support is available at all.

In an SSI cluster, virtual reserve/release can be used on only one system at a time for a given minidisk.

Sharing DASD using the Multi-Path Lock Facility

DASD data can be shared between native systems, between VM guests, or between native systems and VM guests through the use of the Multi-Path Lock Facility (MPLF) on IBM DASD subsystems. The MPLF provides locking serialization for data on the DASD, as the operating systems or programs use the MPLF channel commands to control the locks. The Transaction Processing Facility (TPF) is an example of an operating system that can exploit the MPLF.

VM allows guests to use the MPLF for data that resides on dedicated devices or on full-pack minidisks.

For dedicated devices, the real MPLF support is authorized by the LKFAC option of the directory statement. Authorizing a guest for LKFAC automatically enables all of that guest's dedicated devices to use the real MPLF. The guest must then issue the appropriate MPLF channel programs to exploit the function.

For full-pack minidisks, the real MPLF support is also authorized by the LKFAC option of the directory statement. In addition, the SET LKFACR ON command must be issued to enable each full-pack minidisk to use the real MPLF. Again, once the device is enabled, the guest must then issue the appropriate MPLF channel programs to exploit the function.

Unlike dedicated devices where guest authorization enables all dedicated guest devices, full-pack minidisks are enabled on a device-by-device basis. This allows one guest to use the real MPLF for some full-pack minidisks while at the same time using VM's simulated MPLF for other full-pack minidisks in the virtual configuration.

VM's simulation of the MPLF can be used to share data between VM guests when the real hardware is unavailable. Simulation is available for both full-pack and non-full-pack minidisks. Simulation is based on the original IBM D/T3990 model 3 MPLF RPQ, while real MPLF support is based on the function in the original IBM D/T3990 model 6 controller.

Like the real MPLF, simulated MPLF requires the user to be authorized by the LKFAC option of the OPTION directory statement. In addition, the user must issue the SET LKFAC command to create the simulated

MPLF environment. Then the guest must issue the appropriate MPLF channel programs to utilize the function.

The SET LKFAC command assumes that all full-pack minidisks and all non-full-pack minidisks in the configuration are part of the simulated MPLF. However, no action is taken until the first MPLF-related I/O request is issued to a device. At that time, the device and the 31 associated device addresses are considered to be 'active' in the simulated configuration. For example, I/O to device 191 would cause devices 180 through 19F to be part of one simulated partition (simulating the 32 device addresses on a real DASD subsystem).

Even with the global nature of simulated MPLF, real MPLF can coexist. For example, if device 180 had been enabled for real MPLF (SET LKFACR ON) before the simulated MPLF was enabled or 'active', then the real MPLF would override the simulation for device 180.

Note: Real MPLF and Simulated MPLF are mutually exclusive. It is recommended that the choice be made for each device range before the guest is IPL'd. If, after the guest is IPL'd, you decide to change a device from simulated to real or real to simulated, then the change will cause a SYSTEM RESET and the guest will have to be re-ipl'd. It is also recommended that all the devices within one 32-address range use the same type of MPLF—either all using real MPLF, or all using simulated MPLF.

Cached DASD

Virtual machines can use cached DASD as high-speed storage devices. The term cached DASD refers to a DASD with a control unit that contains a cache. A cache is a high-performance, random-access, electronic storage device. Frequently used pages are kept in the cache where the processor can quickly access them without going to the DASD itself. See *z/VM: General Information*, for a list of supported cached DASD control units and a description of which DASD can be connected to them.

A cached DASD control unit provides the high-speed cache as a connection between the processor channels and certain DASD. Figure 20 on page 676 shows a storage control unit logically attached to a string of 3390 DASD.

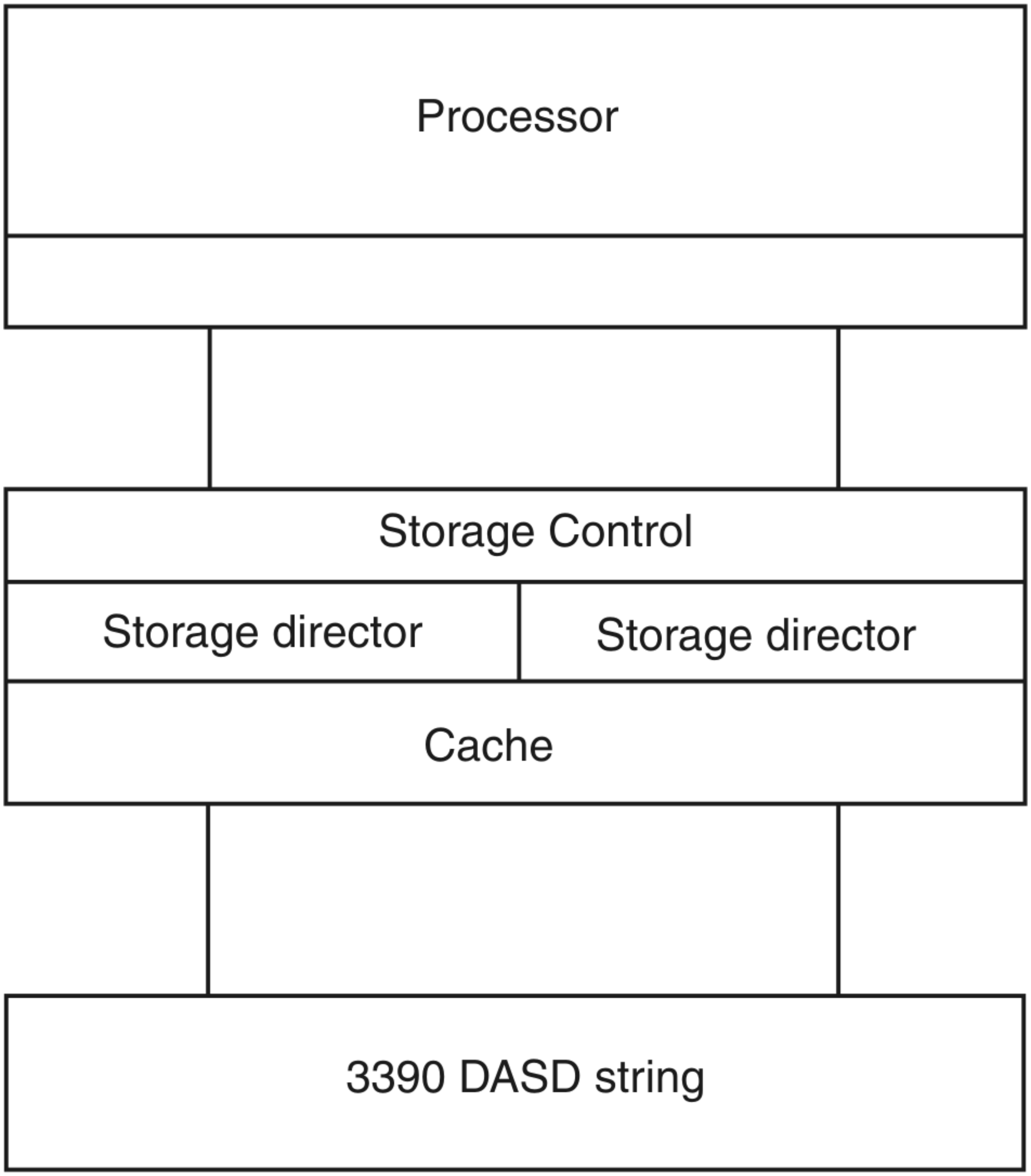


Figure 20. 3390 Cached DASD

I/O to and from the cache is performed electronically. This is faster than normal DASD access, which involves seeking and rotational delays.

Note: CP limits the channel command support to the basic device support. Therefore, CP does not support the speed-matching buffer.

Cache Control

There are three levels of cache control for storage controllers with 3390 DASD: the subsystem level, the device level, and the extent level. To enable caching at a given level means to enable a DASD to accept

CCWs that activate or deactivate caching at that level: the Define Extent CCW for extent-level caching and the Set Subsystem Mode CCW for device- and subsystem-level caching.

Cache control is hierarchical. If subsystem caching is not enabled, then device- and extent-level caching can be enabled for DASD using that storage control, but no caching can be performed. If device-level caching is not enabled, then extent-level caching can be enabled for that DASD, but it cannot be performed.

Cache Control at the Subsystem Level

Caching at the subsystem level is controlled with the DASDOPT directory statement, the ATTACH command, and the SET CACHE command.

The DASDOPT statement with the SYSCTL operand, the ATTACH command with the SYSCTL operand, and the SET CACHE command with the SUBSYSTEM operand enable devices using the storage control to accept CCWs that activate and deactivate caching at the subsystem level.

The SET CACHE SUBSYSTEM command can also turn caching off for devices using the storage control.

Cache Control at the Device Level

Device-level caching is controlled with the DASDOPT directory statement, the ATTACH command, and the SET CACHE command.

The DASDOPT statement with the DEVCTL operand, the ATTACH command with the DEVCTL operand, and the SET CACHE command with the DEVICE option enable a device to accept CCWs that activate and deactivate caching for a device.

When two users have full-pack minidisks on the same volume, both enabled for caching, their choices to activate or deactivate caching can conflict. The conflict can be moderated by using reserve/release (see "Sharing DASD among Multiple Virtual Machines and Other Systems Using Concurrent Virtual and Real Reserve/Release" on page 672).

The SET CACHE DEVICE command can also turn off caching that was enabled by the DASDOPT DEVCTL directory statement or the ATTACH DEVCTL command.

Cache Control at the Extent Level

Caching at the extent level is controlled through the MINIOPT directory statement. The MINIOPT CACHE statement enables a device to accept CCWs that activate and deactivate extent-level caching for a device. MINIOPT CACHE is the default.

Extent-level caching is not available for full-pack minidisks. MINIOPT NOCACHE turns off extent-level caching. Setting NOTRAN on can invalidate MINIOPT CACHE.

Defining a Minidisk on a Cached DASD

You can define a minidisk on a 3390 DASD attached to an IBM DASD subsystem.

The previous sections of this chapter told you how to define a minidisk on DASD other than cached DASD. To define a minidisk on a 3390 DASD attached to a storage control unit (with a cache), you must first define your 3390s by coding an RDEVICE system configuration file statement. For example, you would code the RDEVICE system configuration file statement as:

```
RDEVICE   197-19A   TYPE DASD   SHARED YES
```

This statement defines real devices 197, 198, 199, and 19A.

Next, you must decide whether you want to define a full-pack minidisk. Minidisks other than full-packs are discussed first.

Non-Full-pack Minidisk: To define a 3390 as a minidisk for a virtual machine, code an MDISK directory statement (and, optionally, a MINIOPT directory statement) with this format:

```
MDISK 198 3390 0001 0500 XARES MWV ALL
MINIOPT CACHE
```

Notice that not all of the cylinders on this DASD have been allocated; therefore, this is not a full-pack minidisk. The MINIOPT directory statement indicates this 3390 DASD has access to the cache of a storage control unit.

After you define a minidisk on a cached DASD, other virtual machines can link to that minidisk in the same way that they would link to a minidisk on a normal DASD. This procedure is explained in "Sharing DASD among Multiple Virtual Machines by Using Virtual Reserve/Release" on page 667.

Full-Pack Minidisk: To define a 3390 Model 3 as a full-pack minidisk, you need in your virtual machine definition an MDISK directory statement and, optionally, a DASDOPT directory statement. For example:

```
MDISK 197 3390 0 END XARES MWV ALL
DASDOPT DEVCTL
```

or

```
MDISK 197 3390 0000 3339 XARES MWV ALL
DASDOPT DEVCTL
```

Because all addressable cylinders of this 3390 (Model 3) have been allocated, this is a full-pack minidisk. The DASDOPT directory statement and its operands (DEVCTL, SYSCTL, and NOCTL) determine which types of CCWs the device can accept. If you do not include a DASDOPT directory statement after the MDISK directory statement, the default is DEVCTL. For more information on the DASDOPT directory statement, see "DASDOPT Directory Statement" on page 495.

Defining a Cached DASD as a Dedicated Device

You can give your virtual machine exclusive use of a cached DASD with the DEDICATE directory statement. In your virtual machine's directory, code a DEDICATE directory statement and optionally a DASDOPT directory statement. To dedicate a cached DASD at real device number 199 to virtual device number 199, include these statements in your directory:

```
DEDICATE 199 199
DASDOPT SYSCTL
```

If you do not include the DASDOPT directory statement, the default is DEVCTL.

Using IBM Parallel Access Volumes

z/VM provides support for the IBM Parallel Access Volumes (PAV) feature of IBM DASD subsystems. IBM DASD PAV volumes must be defined to z/VM as a 3390 Model 2, 3, or 9 DASD on a 3990 Model 3 or 6, 2105, 2107, or 1750 Storage Controller. 3380 track-compatibility mode for the 3390 Model 2 or 3 DASD is also supported.

Before the introduction of PAV, the terms "volume" and "subchannel" were frequently used interchangeably when discussing DASD architectures. With the introduction of PAV, a "volume" is more narrowly defined as a named collection of DASD cylinders that contain data, and "subchannels" are envisioned as controlling virtual disk access mechanisms for the DASD volume. Each subchannel has a unique subchannel number and device number. With PAV, a real DASD volume is accessed through a base subchannel and one or more alias subchannels. Although sometimes misleading, the base subchannel's device number identifier is used in many contexts to refer to the "volume" as well. This is a holdover from the widely accepted practice of referring to volumes with a six-character volume serial number and a "subchannel device number" interchangeably when a one-to-one correspondence existed. Today, a PAV volume has one volume serial number and multiple subchannels.

The PAV hardware feature allows you to configure one or more logical DASD volumes, each with a base and one or more alias subchannels. Figure 21 on page 679 illustrates the relationship between base and alias subchannels for a volume. The base subchannel represents the real (physical) DASD volume space. Alias subchannels (X and Y in this example) map to the same physical volume space as accessed by the base. The alias subchannels do not have their own data space; they are "shadows" of the base. This architecture allows concurrent accesses to a volume through its base and associated alias subchannels.

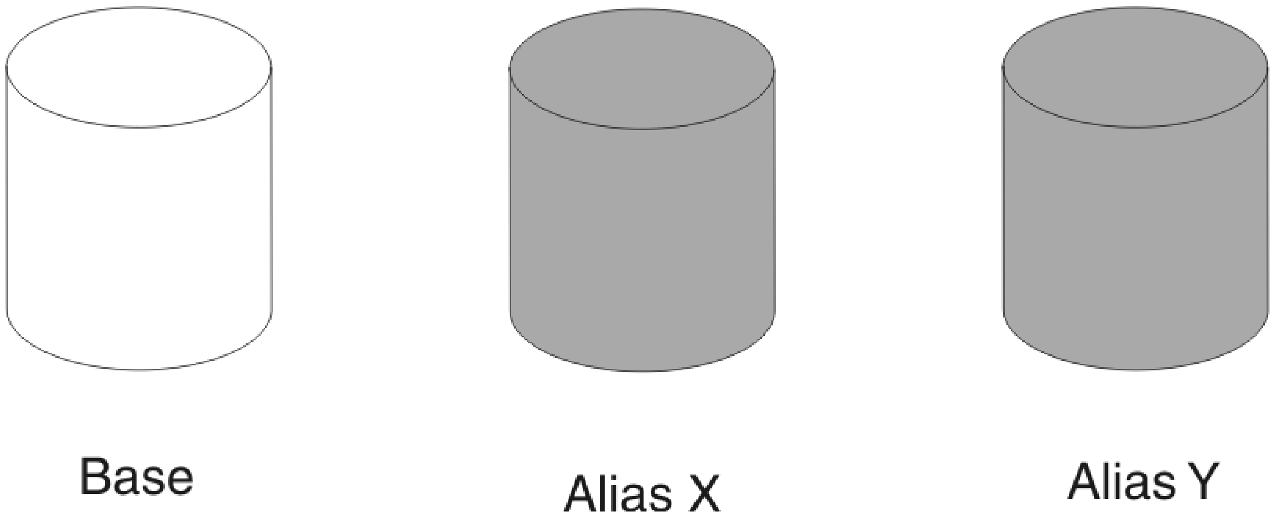


Figure 21. Base and Alias Parallel Access Volumes

Base subchannels are defined in IOCP as UNIT=3990, 2105, 2107, or 1750 on the CNTLUNIT statement and UNIT=3390B (or 3380B) on the IODEVICE statement. Alias subchannels are defined as UNIT=3990, 2105, 2107, or 1750 on the CNTLUNIT statement and UNIT=3390A (or 3380A) on the IODEVICE statement. Each base or alias subchannel can be assigned any available VM real device number. Use the IBM DASD subsystem configuration console to initially define which subchannels are base subchannels, which subchannels are alias subchannels, and which alias subchannels are associated with each base volume. Use the CP QUERY PAV command to view the current allocation of base and alias subchannels.

Base and alias subchannels provide nearly identical functions for the volume. One exception is that "volume-wide" commands such as the Reserve and Release channel commands can only be issued to a base subchannel, but the resulting status applies to all of the alias subchannels as well.

Certain virtual PAV operations require the consistent use of the same real base or alias subchannel. To facilitate this, each virtual PAV base and alias has an "assigned" real device subchannel that can be displayed with the "QUERY VIRTUAL *vdev* DETAILS" and "QUERY VIRTUAL PAV" commands. The assignment is automatic and cannot be changed. One example of this would be the execution of the Read Configuration Data command. The scheduling of I/O to an assigned device is automatically handled by z/VM during its analysis of the virtual channel program. Because each virtual PAV base or alias must have a uniquely assigned real PAV base or alias subchannel, you cannot have more virtual aliases than real aliases for a volume.

The Define Extent channel command specifies if a channel program can read and/or write data from or to a volume. For each volume, read operations are permitted concurrently over multiple base or alias subchannels. However, write operations are serialized on the volume when the cylinder ranges specified in the Define Extent channel command overlap with another active CCW chain in any other subchannel for the volume.

A dedicated PAV volume cannot have its alias subchannels spread among multiple guests. The use of shared minidisks provides that functionality.

Using PAV Dedicated DASDs

To use PAV dedicated devices, the guest must contain support for managing and serializing the volume's data across the subchannels – an "exploiting" operating system. z/VM acts only as the "pipe" between the guest and the hardware. Once the necessary base and associated alias subchannels are attached to the guest, the guest must manage their use.

In a dedicated environment, the performance benefits of PAV are entirely up to the operating system running in the virtual machine. z/VM will not make any attempt to optimize or alter the I/O flowing through the base and alias subchannels. Note that real volumes cannot be dedicated to multiple guests. Figure 22 on page 680 shows a typical example of a guest virtual machine using two dedicated PAV volumes with one base subchannel and two alias subchannels for each volume:

```
  USER GUEST1

DEDICATE E100 1300
DEDICATE E140 1301
DEDICATE E141 1302
DEDICATE E101 1303
DEDICATE E142 1304
DEDICATE E143 1305
```

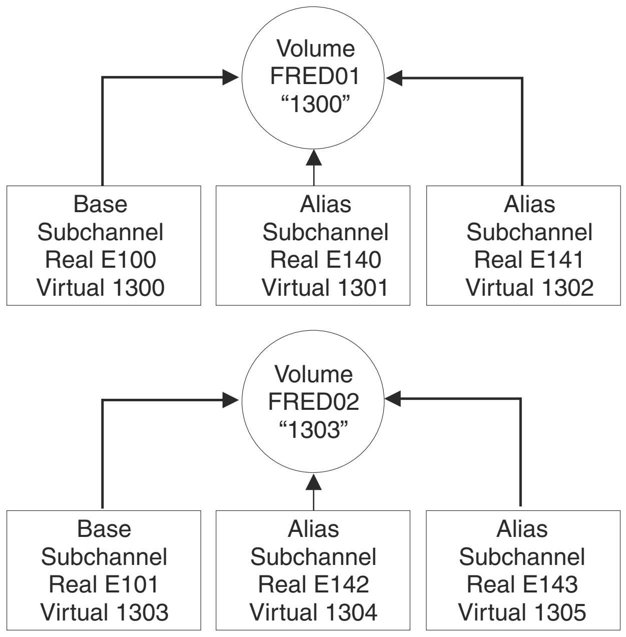

Volume
FRED01
“1300”
Base Subchannel Real E100 Virtual 1300
Alias Subchannel Real E140 Virtual 1301
Alias Subchannel Real E141 Virtual 1302
Volume
FRED02
“1303”
Base Subchannel Real E101 Virtual 1303
Alias Subchannel Real E142 Virtual 1304
Alias Subchannel Real E143 Virtual 1305

Figure 22. A PAV Dedicated Configuration: An Example

Using PAV Minidisks

In the context of PAV, the support for both full-pack and non-full-pack minidisks behaves in the same manner.

A guest virtual machine can define one or more minidisk volumes that exist on a real PAV volume. The real PAV volume has a real base and one or more real PAV alias subchannels. Each minidisk volume has one virtual PAV base and zero or more virtual PAV alias subchannels. These can be displayed with the QUERY VIRTUAL PAV command. For each minidisk volume, the number of virtual PAV aliases for a guest cannot exceed the number of real PAV aliases defined in the hardware for the underlying real volume.

All I/O operations that are directed to a minidisk volume through either its virtual base or alias subchannels are optimized by z/VM's automatic selection of an appropriate real PAV base or alias subchannel for the underlying real volume. In other words, the scheduling of I/O to a virtual PAV base or alias subchannel will be dynamically scheduled and multiplexed on any real PAV base or alias subchannel that is defined in the hardware. This gives z/VM the flexibility to choose a real PAV base or alias subchannel that is not in use at the time. For example, if users GUEST1 and GUEST2 simultaneously issue an I/O request to two different minidisk volumes that are defined on the same real underlying volume, via their respective virtual base subchannels, the result would be that one I/O would be executed on the real base subchannel and the other, simultaneously, would be executed on a real alias subchannel.

The Minidisk Cache and the SET MDCACHE command are supported for PAV minidisk volumes.

Using PAV Minidisks with Exploiting Operating Systems

An *exploiting* operating system is one that is capable of controlling the PAV architecture and is configured to control the features of PAV. Such an operating system understands how to control and utilize virtual PAV aliases. Examples might be z/VM, z/OS, or Linux.

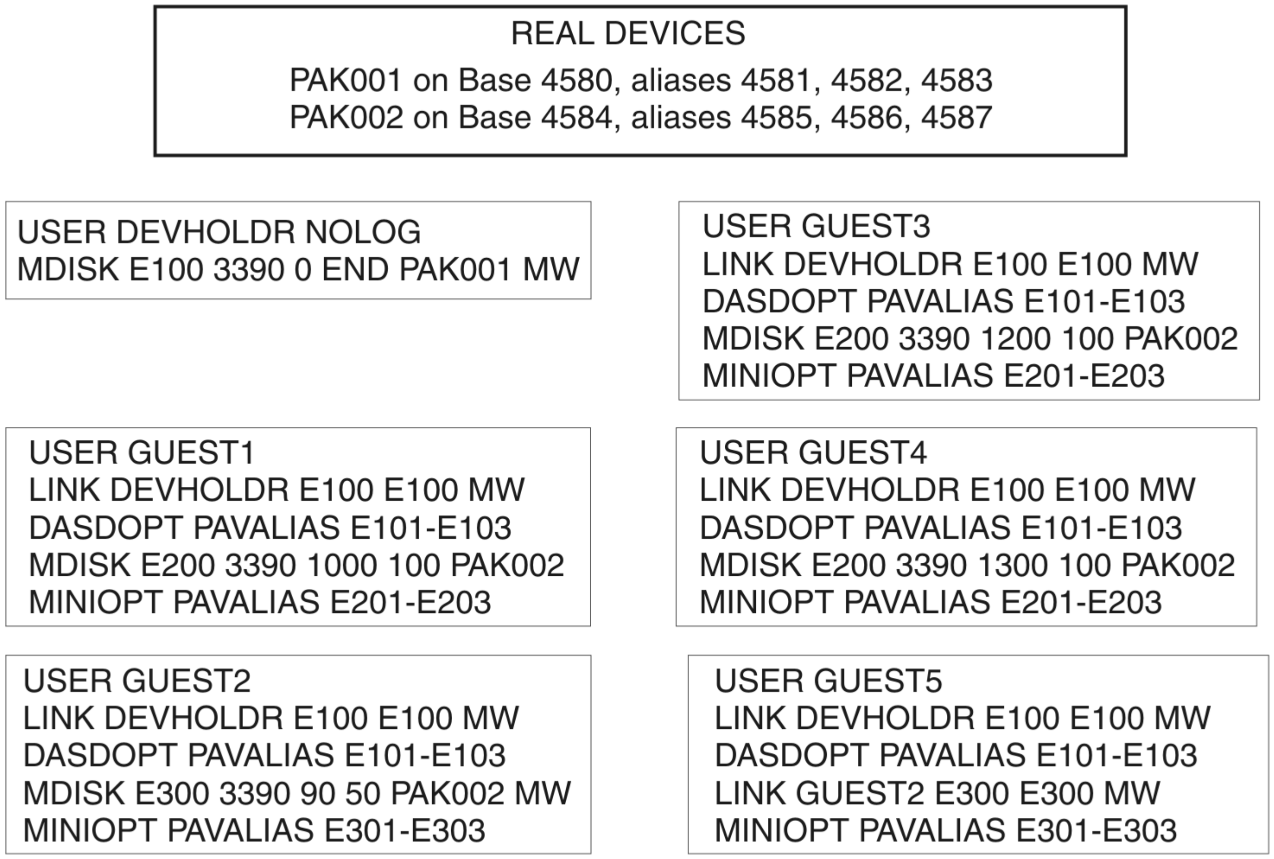
To define a full-pack minidisk for an exploiting operating system at virtual E100 with virtual aliases at E101, E102, and E103, you can code the following statements in the user directory:

```
MDISK E100 3390 0 END PAK001
DASDOPT PAVALIAS E101-E103
      or
LINK GUEST1 E100 E100 MW
DASDOPT PAVALIAS E101-E103
```

To define a non-full-pack minidisk for an exploiting operating system at virtual E100 with virtual aliases at E101, E102, and E103, you can code the following statements in the user directory:

```
MDISK E100 3390 100 200 PAK002
MINIOPT PAVALIAS E101-E103
      or
LINK GUEST1 E100 E100 MW
MINIOPT PAVALIAS E101-E103
```

The following is a typical example of several guest virtual machines that exploit PAV volumes. One volume is a full-pack minidisk that is shared among the five guests (E100, PAK001), and there are four non-full-pack minidisk volumes (E200s and E300) that share the same underlying real PAV volume (PAK002). Note that there are more PAV minidisk volumes (5 MDISKs) than real volumes (2). z/VM will multiplex I/O operations on the real base and alias subchannels for each:



```
REAL DEVICES
PAK001 on Base 4580, aliases 4581, 4582, 4583
PAK002 on Base 4584, aliases 4585, 4586, 4587
```

```
USER DEVHOLDR NOLOG
MDISK E100 3390 0 END PAK001 MW
```

```
USER GUEST3
LINK DEVHOLDR E100 E100 MW
DASDOPT PAVALIAS E101-E103
MDISK E200 3390 1200 100 PAK002
MINIOPT PAVALIAS E201-E203
```

```
USER GUEST1
LINK DEVHOLDR E100 E100 MW
DASDOPT PAVALIAS E101-E103
MDISK E200 3390 1000 100 PAK002
MINIOPT PAVALIAS E201-E203
```

```
USER GUEST4
LINK DEVHOLDR E100 E100 MW
DASDOPT PAVALIAS E101-E103
MDISK E200 3390 1300 100 PAK002
MINIOPT PAVALIAS E201-E203
```

```
USER GUEST2
LINK DEVHOLDR E100 E100 MW
DASDOPT PAVALIAS E101-E103
MDISK E300 3390 90 50 PAK002 MW
MINIOPT PAVALIAS E301-E303
```

```
USER GUEST5
LINK DEVHOLDR E100 E100 MW
DASDOPT PAVALIAS E101-E103
LINK GUEST2 E300 E300 MW
MINIOPT PAVALIAS E301-E303
```

Figure 23. Example: PAV Minidisk Configuration for Exploiting Guests

Using PAV Minidisks with Non-Exploiting Operating Systems

A *non-exploiting* operating system is one that is not configured to control the features of PAV or has no knowledge of the PAV architecture. Although the guest operating system won't use its minidisk volumes in PAV mode, z/VM will still provide PAV performance optimization across multiple non-exploiting guests. Performance gains can be realized only when full-pack minidisks are shared among guests with multiple LINK statements or when multiple non-full-pack minidisk volumes reside on a real PAV volume. Performance gains are achieved by transparently multiplexing the I/O operations requested on each guest minidisk volume over the appropriate real PAV base and alias subchannels. CMS, TPF, and VSE are examples of non-exploiting operating systems. Linux and z/OS can also be considered non-exploiting, depending on how they are configured.

To define a full-pack minidisk for a non-exploiting operating system at virtual E100, you can code the following statements in the user directory:

```
MDISK E100 3390 0 END PAK001
      or
LINK GUEST1 E100 E100 MW
```

To define a non-full-pack minidisk for a non-exploiting operating system at virtual E100, you can code the following statements in the user directory:

```
MDISK E100 3390 100 200 PAK002
      or
LINK GUEST1 E100 E100 MW
```

The following is a typical example of several non-exploiting guest virtual machines using PAV for enhanced performance:

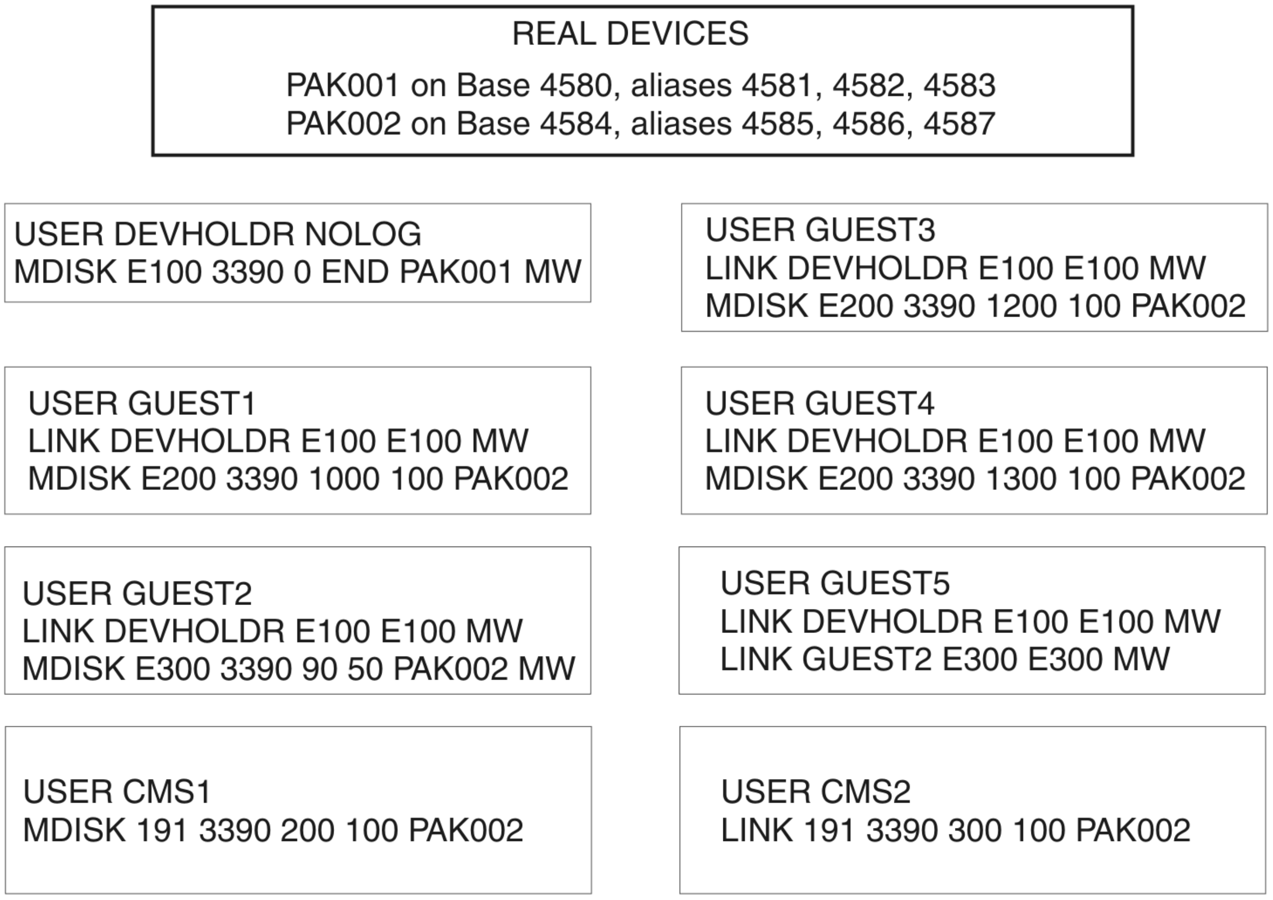

REAL DEVICES
PAK001 on Base 4580, aliases 4581, 4582, 4583
PAK002 on Base 4584, aliases 4585, 4586, 4587
USER DEVHOLDR NOLOG
MDISK E100 3390 0 END PAK001 MW
USER GUEST3
LINK DEVHOLDR E100 E100 MW
MDISK E200 3390 1200 100 PAK002
USER GUEST1
LINK DEVHOLDR E100 E100 MW
MDISK E200 3390 1000 100 PAK002
USER GUEST4
LINK DEVHOLDR E100 E100 MW
MDISK E200 3390 1300 100 PAK002
USER GUEST2
LINK DEVHOLDR E100 E100 MW
MDISK E300 3390 90 50 PAK002 MW
USER GUEST5
LINK DEVHOLDR E100 E100 MW
LINK GUEST2 E300 E300 MW
USER CMS1
MDISK 191 3390 200 100 PAK002
USER CMS2
LINK 191 3390 300 100 PAK002

Figure 24. Example: PAV Minidisk Configuration for Non-Exploiting Guests

Note that in the above configuration, there are five links to real volume PAK001. For these five virtual PAV base subchannels, there is one real PAV base and three real PAV alias subchannels that will be used to perform the I/O. z/VM will concurrently multiplex the I/O from the GUEST1-GUEST5 E100 virtual bases onto the real 4580, 4581, 4582, and 4583 subchannels as they are available. Using this strategy, it is possible to have many guest virtual machines sharing a real DASD volume with z/VM dynamically handling the selection of real PAV base and alias subchannels.

I/O operations to the minidisks defined on PAK002 will likewise be optimized by z/VM's dynamic selection of real PAV base and alias subchannels 4584, 4585, 4586, and 4587.

Note: CMS is a non-exploiting operating system, and therefore the use the DASDOPT PAVALIAS and MINIOPT PAVALIAS user directory statements is not recommended. Additionally, the use of the Class G DEFINE PAVALIAS command by CMS users should be discouraged. It is possible to write a CMS application that can take advantage of PAV volumes, but CMS itself is not PAV-aware. When multiple CMS volumes are defined on a real PAV volume, I/O operations by CMS can be concurrently scheduled on any real PAV base or alias subchannel by z/VM. The CMS user does not need to take any action for this to occur.

z/VM Restrictions on Using PAV

1. A virtual alias subchannel cannot be IPLed.
2. You should not use PAV alias volumes as z/VM installation volumes (for example, do not use for the 520RES volume).
3. z/VM Paging and SPOOLing operations do not take advantage of PAV. It is recommended that PAGE and SPOOL areas be placed on DASD devices dedicated to this purpose.
4. A real alias subchannel can be attached to a guest or SYSTEM only after its associated real base subchannel has been attached to the same guest or SYSTEM.
5. A real base subchannel can be detached from a guest or SYSTEM only if all of its associated alias subchannels are already free.
6. A real alias subchannel will not come online to z/VM without an associated real base subchannel. Also, a real base subchannel must have at least one associated real alias subchannel for z/VM (for example, the QUERY PAV command) to recognize the device as a Parallel Access Subchannel.
7. A real base subchannel cannot be changed or deleted with the SET RDEVICE, DELETE RDEVICE, DELETE DEVICE, or MODIFY DEVICE command unless all associated real alias subchannels have been deleted with the DELETE RDEVICE command..
8. CMS does not support virtual alias subchannels. Under CMS, you can for example issue a Class G "DEFINE PAVALIAS 291 FOR BASE 191" command followed by an "ACCESS 291 B". If you use "FILELIST * * B", you will see the contents of the 191 disk, but CMS does not understand that 191-A and 291-B are actually the same volume and corruption will occur if changes are made to either 191 or 291. This is similar to the damage that can be caused by issuing "LINK * 191 291 MW".
9. Virtual PAV devices should not be used if any system sharing the DASD uses the unconditional reserve CCW. This includes both dedicated and DEFINE PAVALIAS devices.

Using IBM HyperParallel Access Volumes

Read the above section, "Using IBM Parallel Access Volumes" on page 678, before reading this section.

z/VM provides support for the IBM HyperParallel Access Volumes (HyperPAV) feature of IBM DASD subsystems. For specific support requirements, see the DASD support table in *z/VM: General Information*.

Traditional PAV support operates on statically assigning one or more PAV alias subchannels to a specific PAV base device. The DASD Administrator is able to manually reassign PAV aliases from one PAV base to another using the DASD subsystem's configuration menus and certain software can "dynamically" reassign PAV aliases as well. When there are many PAV bases and aliases, it is possible to begin to exhaust the supply of subchannels that are available. This potential for exhausting the supply of available subchannels and easier system operation has led to the creation of HyperPAV.

A Logical Subsystem (LSS) can operate in one of the Non-PAV, PAV, or HyperPAV modes. When an LSS is in HyperPAV mode, there is one pool of HyperPAV bases and aliases that are shared within the LSS. Any HyperPAV alias in a pool can service I/O requests for any HyperPAV base in the same pool. Thus, the PAV concept of an alias being assigned to a particular base is no longer appropriate. Instead, the HyperPAV base-alias association exists only for the duration of each I/O operation on a HyperPAV alias. This pooling of HyperPAV bases and aliases can greatly reduce the number of aliases that can be required. With HyperPAV, only the number of aliases required to obtain a desired I/O performance objective for the LSS are required — performance tuning is now moved from the volume level to the LSS level.

Within the HyperPAV content, the concept of a "volume" becomes a bit more "natural" in the sense that a HyperPAV alias subchannel no longer has a fixed association with a particular volume. With PAV, an alias subchannel was associated with a particular volume, and this led to some confusion. In the HyperPAV world, there is no such association.

HyperPAV devices are defined within a Storage Controller when the proper Licensed Internal Codes (LICs) are installed and enabled. The LSS is configured as a PAV environment, and when the HyperPAV feature is enabled by z/VM, the static PAV aliases are converted to HyperPAV aliases and they are joined together to form the pool for the LSS. z/VM can be configured to operate each LSS in Non-PAV, PAV, or HyperPAV mode by using the new CU DASD statement in its configuration file and/or the new SET CU command.

PAV base subchannels are defined in IOCP as UNIT=3990, 2105, or 2107 on the CNTLUNIT statement and UNIT=3390B (or 3380B) on the IODEVICE statement. Alias subchannels are defined as UNIT=3990, 2105, or 2107 on the CNTLUNIT statement and UNIT=3390A (or 3380A) on the IODEVICE statement. Each base or alias subchannel can be assigned any available z/VM real device number. Use the IBM DASD subsystem configuration console to initially define which subchannels are base subchannels, which subchannels are alias subchannels, and which alias subchannels are associated with each base volume. Use the CP QUERY PAV command to view the current allocation of base and alias subchannels.

Base and alias subchannels provide nearly identical functions for a volume. One exception is that "volume-wide" commands such as the Reserve and Release channel commands can only be issued to a base subchannel, but the resulting status applies to the associated alias subchannels as well.

Certain virtual HyperPAV operations require the consistent use of the same real base or alias subchannel. To facilitate this, each virtual HyperPAV base and alias has an "assigned" real device subchannel that can be displayed with the QUERY VIRTUAL *vdev* DETAILS and QUERY VIRTUAL PAV commands. The assignment is automatic and cannot be changed. One example of this would be the execution of the Read Configuration Data command. The scheduling of I/O to an assigned device is automatically handled by z/VM during its analysis of the virtual channel program. Because each virtual HyperPAV base or alias must have a uniquely assigned real HyperPAV base or alias subchannel, you cannot have more virtual HyperPAV aliases than real HyperPAV aliases for an LSS.

The Define Extent channel command specifies if a channel program can read and/or write data from or to a volume. For each volume, read operations are permitted concurrently over multiple base or alias subchannels. However, write operations are serialized on the volume when the cylinder ranges specified in the Define Extent channel command overlap with another active CCW chain in any other subchannel for the volume.

A dedicated HyperPAV base volume or alias can only be assigned to one guest. I/O operations initiated through a HyperPAV alias can only be directed to base volumes that are ATTACHED or LINKED to the issuing virtual machine.

HyperPAV Pools

HyperPAV support includes the concept of a pool. A pool consists of a collection of HyperPAV bases and the alias subchannels that can refer to them. A typical PAV configuration can be defined as in the figure below:

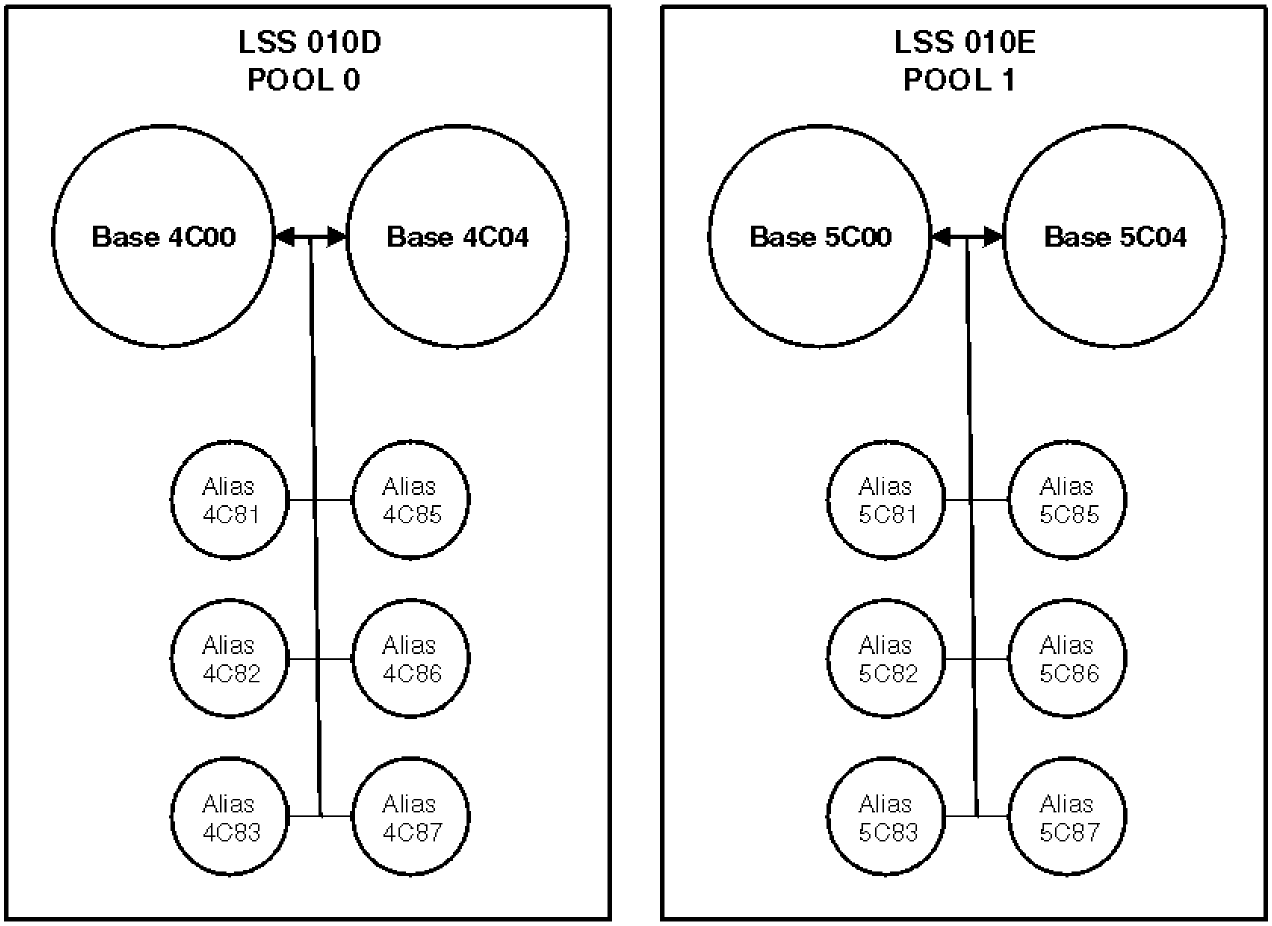


Figure 25. Example: DASD Logical Subsystems and Pools

A pool can contain up to 254 HyperPAV alias devices, and there is a limit of 16,000 pools in a z/VM configuration.

There is a one-to-one correspondence of pools and LSSs. Base disks are assigned to a specific pool and aliases within the same pool can be used to access the base.

Using HyperPAV Dedicated DASDs

Dedicated HyperPAV base devices operate in the traditional z/VM manner. To use dedicated HyperPAV alias devices, the guest must contain support for managing and serializing the volume's data across the subchannels - an "exploiting" operating system. z/VM acts only as the "pipe" between the guest and the hardware. After the necessary base subchannel and associated alias subchannels are attached to the guest, the guest must manage their use. Dedicated HyperPAV alias I/O operations are restricted by z/VM to be able to access only HyperPAV base devices that are attached to the guest.

In a dedicated environment, the performance benefits of HyperPAV are entirely up to the operating system running in the virtual machine. z/VM will not make any attempt to optimize or alter the I/O flowing through the base and alias subchannels.

Using HyperPAV Minidisks

In the context of HyperPAV, the real I/O scheduling algorithms for full-pack and non-full-pack minidisks behave in the same manner.

A guest virtual machine can define one or more minidisk volumes that exist on a real HyperPAV volume. The real HyperPAV volume has a real base and is associated with a pool that also contains zero or more HyperPAV aliases.

All HyperPAV I/O operations that are directed to a minidisk volume are optimized by the z/VM automatic selection of an appropriate real HyperPAV base or alias subchannel for the underlying real volume. In other words, the scheduling of I/O to a virtual device will be dynamically scheduled and multiplexed on any real HyperPAV base or alias subchannel that is defined in the hardware. This gives z/VM the flexibility to choose a real HyperPAV base or alias subchannel that is not in use at the time. For example, if users GUEST1 and GUEST2 simultaneously issue an I/O request to two different minidisk volumes that are defined on the same real underlying volume, via their respective virtual subchannels, the result would be that one I/O would be executed on the real base subchannel and the other, simultaneously, would be executed on a real alias subchannel.

Using HyperPAV for Paging

As with minidisk I/O, HyperPAV alias devices can be used to assist in the execution of CP Paging Subsystem I/O directed at HyperPAV base volumes in the same pool. This support is enabled through the FEATURES ENABLE PAGING_ALIAS system configuration file statement or the CP command SET PAGING ALIAS ON. When enabled, HyperPAV paging I/O operations are optimized by the zVM I/O scheduler to execute the I/O on the HyperPAV base volume (if it is currently idle) or on an available system-attached HyperPAV alias volume in the same pool (if available). Otherwise, the I/O is queued at the base HyperPAV device for subsequent execution.

Note: The Paging Subsystem performs I/O to any volume that contains PAGE, SPOL, or DRCT allocations, in addition to mapped-minidisk pool I/O (as established using the MAPMDISK macro).

HyperPAV Minidisks and HyperPAV for Paging

When some HyperPAV bases are used by the CP Paging Subsystem and some are used for guest I/O to minidisks in the same pool and system-attached HyperPAV aliases exist in that pool, the VM I/O scheduler will use the HyperPAV aliases to assist in the execution of I/O for both types of HyperPAV bases on a first-come, first-served (FCFS) basis. When all system-attached HyperPAV aliases in a pool are currently busy executing I/O and then one becomes available, that FCFS algorithm can be fine-tuned using the SET CU command to give alias share priority to one type over the other, if so desired. Monitor data can be used to determine whether SET CU is needed to change the alias share.

Using HyperPAV Minidisks with Exploiting Operating Systems

An exploiting operating system is one that is capable of controlling the HyperPAV architecture and is configured to control the features of HyperPAV. Such an operating system understands how to control and utilize virtual HyperPAV aliases. Examples might be z/VM and z/OS.

Virtual HyperPAV base devices can only be defined as full-pack or 1-END minidisks on real HyperPAV base devices. Associated virtual HyperPAV alias devices can be subsequently defined using the DEFINE HYPERPAVALIAS command (in the user directory or after the user is logged on). Virtual HyperPAV devices can be displayed using the QUERY VIRTUAL PAV command. For each LSS, the number of virtual HyperPAV aliases for a guest cannot exceed the number of real HyperPAV aliases defined in the hardware for the underlying real LSS.

The SET MDCACHE command is not valid for an alias HyperPAV minidisk volume. Cache settings are only applicable for base HyperPAV minidisk volumes.

To define a full-pack minidisk for an exploiting operating system at virtual E100 with virtual aliases at E101, E102, and E103, you can code the following statements in the user directory:

```
COMMAND DEFINE HYPERPAVALIAS E101 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E102 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E103 FOR BASE E100
MDISK E100 3390 0 END PAK001
```

or

```
COMMAND DEFINE HYPERPAVALIAS E101 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E102 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E103 FOR BASE E100
LINK GUEST1 E100 E100 MW
```

To define a 1-END minidisk for an exploiting operating system at virtual F100 with virtual aliases at F101, F102, and F103, you can code the following statements in the user directory:

```

COMMAND DEFINE HYPERPAVALIAS F101 FOR BASE F100
COMMAND DEFINE HYPERPAVALIAS F102 FOR BASE F100
COMMAND DEFINE HYPERPAVALIAS F103 FOR BASE F100
MDISK F100 3390 1 END PAK001
```

Figure 26 on page 688 shows a typical example of several virtual machines that exploit HyperPAV volumes. Both volumes are full-pack minidisks that are shared among the five guests (E100, PAK001 and E200, PAK002). Note that there are more HyperPAV minidisk volumes (five MDISKs) than real volumes (two). z/VM will multiplex I/O operations on the real base subchannel and alias subchannels for each volume:

```
                    REAL DEVICES
      PAK001 on Base 4580, aliases 4581, 4582, 4583
      PAK002 on Base 4584, aliases 4585, 4586, 4587
```

```
USER DEVHOLDR NOLOG
MDISK E100 3390 0 END PAK001 MW
MDISK E200 3390 0 END PAK002 MW
```

```
USER GUEST1
COMMAND DEFINE HYPERPAVALIAS E101 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E102 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E103 FOR BASE E100
LINK DEVHOLDR E100 E100 MW
COMMAND DEFINE HYPERPAVALIAS E201 FOR BASE E200
COMMAND DEFINE HYPERPAVALIAS E202 FOR BASE E200
COMMAND DEFINE HYPERPAVALIAS E203 FOR BASE E200
LINK DEVHOLDR E200 E200 MW
```

```
USER GUEST2
COMMAND DEFINE HYPERPAVALIAS E101 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E102 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E103 FOR BASE E100
LINK DEVHOLDR E100 E100 MW
COMMAND DEFINE HYPERPAVALIAS E201 FOR BASE E200
COMMAND DEFINE HYPERPAVALIAS E202 FOR BASE E200
COMMAND DEFINE HYPERPAVALIAS E203 FOR BASE E200
LINK DEVHOLDR E200 E200 MW
```

```
USER GUEST3
COMMAND DEFINE HYPERPAVALIAS E101 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E102 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E103 FOR BASE E100
LINK DEVHOLDR E100 E100 MW
COMMAND DEFINE HYPERPAVALIAS E201 FOR BASE E200
COMMAND DEFINE HYPERPAVALIAS E202 FOR BASE E200
COMMAND DEFINE HYPERPAVALIAS E203 FOR BASE E200
LINK DEVHOLDR E200 E200 MW
```

```
USER GUEST4
COMMAND DEFINE HYPERPAVALIAS E101 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E102 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E103 FOR BASE E100
LINK DEVHOLDR E100 E100 MW
COMMAND DEFINE HYPERPAVALIAS E201 FOR BASE E200
COMMAND DEFINE HYPERPAVALIAS E202 FOR BASE E200
COMMAND DEFINE HYPERPAVALIAS E203 FOR BASE E200
LINK DEVHOLDR E200 E200 MW
```

```
USER GUEST5
COMMAND DEFINE HYPERPAVALIAS E101 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E102 FOR BASE E100
COMMAND DEFINE HYPERPAVALIAS E103 FOR BASE E100
LINK DEVHOLDR E100 E100 MW
COMMAND DEFINE HYPERPAVALIAS E201 FOR BASE E200
COMMAND DEFINE HYPERPAVALIAS E202 FOR BASE E200
COMMAND DEFINE HYPERPAVALIAS E203 FOR BASE E200
LINK DEVHOLDR E200 E200 MW
```

Figure 26. A HyperPAV Minidisk Configuration for Exploiting Guests: An Example

Note that in Figure 26 on page 688, there is no reference to pool numbers. The purpose of the FOR BASE *nnnn* option on the DEFINE HYPERPAVALIAS command is to make sure the virtual alias is assigned to an appropriate real alias in the same pool as the base. If, in Figure 26 on page 688, PAK001 and PAK002 are in the same real LSS, all of the devices are assigned to the same pool and the E1*nn* and E2*nn* aliases can be used to issue I/O requested to PAK001, PAK002, or both. If PAK001 and PAK002 are in different logical subsystems, the E1*nn* aliases are in a unique pool and can only access PAK001 and the E2*nn* aliases can only access PAK002.

Using HyperPAV Minidisks with Non-Exploiting Operating Systems

A non-exploiting operating system is one that is not configured to control the features of HyperPAV or has no knowledge of the HyperPAV architecture. Although the guest operating system will not use its minidisk volumes in HyperPAV mode, z/VM will still provide HyperPAV performance optimization across multiple non-exploiting guests. Performance gains can be realized only when full-pack or 1-END minidisks are shared among guests with multiple LINK statements or when multiple non-full-pack minidisk volumes reside on a real HyperPAV volume. Performance gains are achieved by transparently multiplexing the I/O operations requested on each guest minidisk volume over the appropriate real HyperPAV base and alias subchannels. CMS, TPF, and VSE are examples of non-exploiting operating systems. z/OS can also be considered non-exploiting, depending on how it is configured.

To define a full-pack minidisk for a non-exploiting operating system at virtual E100, you can code the following statements in the user directory:

```
MDISK E100 3390 0 END PAK001
     or
LINK GUEST1 E100 E100 MW
```

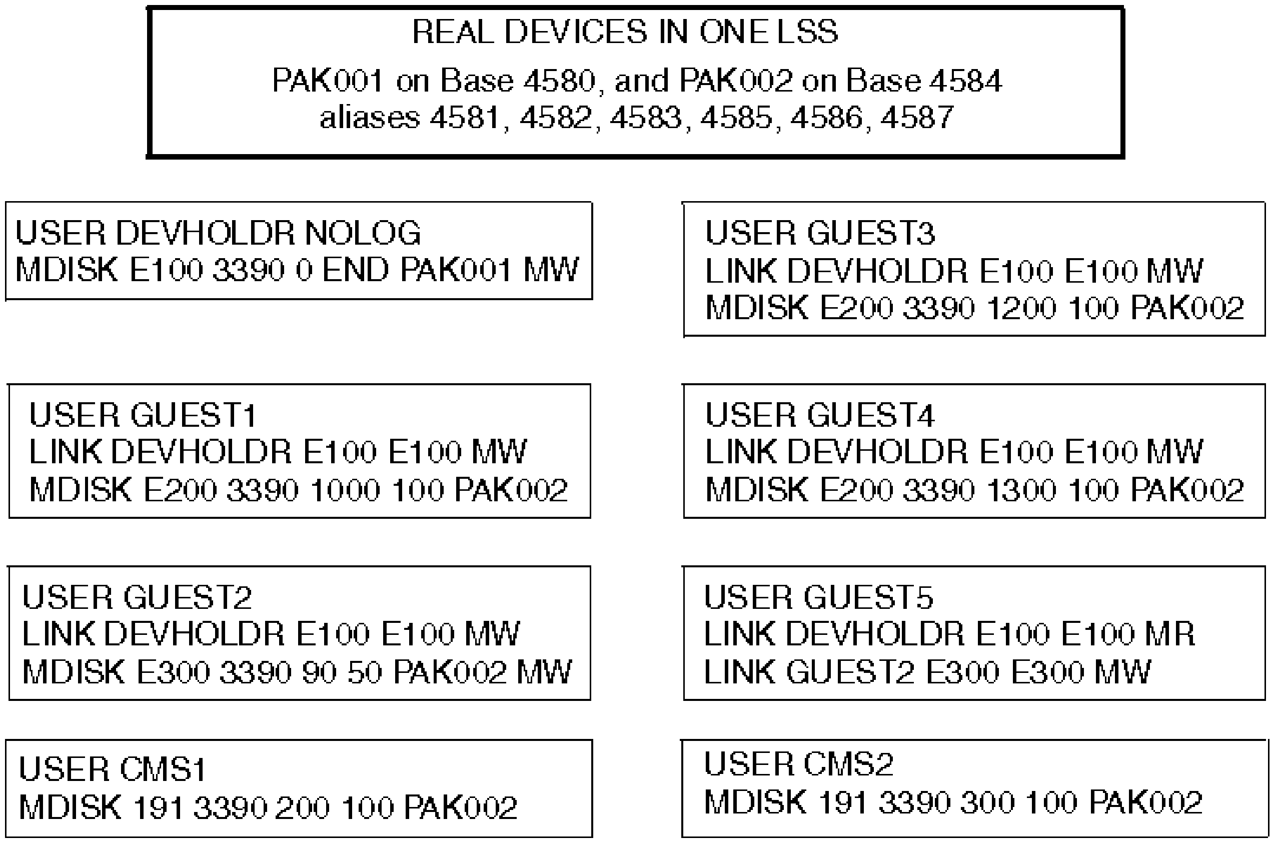
To define a 1-END minidisk for a non-exploiting operating system at virtual F100, you can code the following statements in the user directory:

```
MDISK F100 3390 0 END PAK001
     or
LINK GUEST1 F100 F100 MW
```

To define a non-full-pack minidisk for a non-exploiting operating system at virtual E100, you can code the following statements in the user directory:

```
MDISK E100 3390 100 200 PAK002
     or
LINK GUEST1 E100 3100 MW
```

The following is a typical example of several non-exploiting guest virtual machines using HyperPAV for enhanced performance:



```
REAL DEVICES IN ONE LSS
PAK001 on Base 4580, and PAK002 on Base 4584
aliases 4581, 4582, 4583, 4585, 4586, 4587
```

```
USER DEVHOLDR NOLOG
MDISK E100 3390 0 END PAK001 MW
```

```
USER GUEST3
LINK DEVHOLDR E100 E100 MW
MDISK E200 3390 1200 100 PAK002
```

```
USER GUEST1
LINK DEVHOLDR E100 E100 MW
MDISK E200 3390 1000 100 PAK002
```

```
USER GUEST4
LINK DEVHOLDR E100 E100 MW
MDISK E200 3390 1300 100 PAK002
```

```
USER GUEST2
LINK DEVHOLDR E100 E100 MW
MDISK E300 3390 90 50 PAK002 MW
```

```
USER GUEST5
LINK DEVHOLDR E100 E100 MR
LINK GUEST2 E300 E300 MW
```

```
USER CMS1
MDISK 191 3390 200 100 PAK002
```

```
USER CMS2
MDISK 191 3390 300 100 PAK002
```

Figure 27. Example: HyperPAV Minidisk Configuration for Non-Exploiting Guests

Note that in the above configuration, there are five links to real volume PAK001. For these five virtual HyperPAV base subchannels, there is one real HyperPAV base and six real HyperPAV alias subchannels that will be used to perform the I/O. z/VM will concurrently multiplex the I/O from the GUEST1-GUEST5 E100 virtual bases onto the real 4580, 4581, 4582, 4583, 4585, 4586, and 4587 subchannels as they are available. Using this strategy, it is possible to have many guest virtual machines sharing a real DASD volume with z/VM dynamically handling the selection of real HyperPAV base and alias subchannels.

I/O operations to the minidisks defined on PAK002 will likewise be optimized by z/VM's dynamic selection of real HyperPAV base and alias subchannels 4584, 4581, 4582, 4583, 4585, 4586, and 4587.

Due to their dynamic nature, HyperPAV aliases have additional CCW controls that are not present for PAV aliases. For example, under CMS it is possible (but not recommended) to use the DDR program to access a PAV alias since there is a fixed association of the PAV alias to a specific PAV base. Since DDR does not understand (that is, is non-exploiting) how to control a HyperPAV alias, any attempt to access a HyperPAV alias will result in an I/O error because it is unclear which base is intended.

Note: CMS is a non-exploiting operating system, and therefore the use of the class G DEFINE HYPERPAVALIAS command is not recommended. CMS itself is not HyperPAV-aware. When multiple CMS volumes are defined on a real HyperPAV volume, I/O operations by CMS can be concurrently scheduled on any real HyperPAV base or alias subchannel by z/VM. The CMS user does not need to take any action for this to occur.

z/VM Restrictions on Using HyperPAV

A virtual alias subchannel cannot be IPLed.

You should not use HyperPAV alias volumes as z/VM installation volumes (for example, do not use for the 520RES volume).

Virtual HyperPAV devices can only be defined for full-pack or 1-END minidisks.

For a HyperPAV exploiting guest, it is recommended to avoid defining a mixture of dedicated HyperPAV alias devices and full-pack or 1-END HyperPAV alias devices for the same underlying real LSS. Dedicated HyperPAV aliases can only be associated with dedicated HyperPAV bases and full-pack or 1-END HyperPAV aliases can only be associated with full-pack or 1-END HyperPAV aliases. If a mixture is defined for an LSS, the full compliment of alias devices cannot be exploited for each base device.

CMS does not support virtual HyperPAV alias subchannels.

DIAGNOSE codes X'18', X'20', X'A4', X'250', and the *BLOCKIO System Service do not support HyperPAV alias devices. I/O issued to a HyperPAV alias via one of these interfaces will be rejected because a means for specification of the associated base device is not provided.

Virtual HyperPAV devices should not be used if any system sharing the DASD uses the unconditional reserve CCW. This includes both dedicated and DEFINE HYPERPAVALIAS devices.

Using Persistent FlashCopy

The IBM System Storage DASD Subsystem has introduced the concept of a Persistent FlashCopy Relationship, available with Licensed Internal Code. This feature enables the creation of instant point-in-time copies of dedicated devices, full-pack minidisks, and ordinary minidisks.

The various features of the FLASHCOPY ESTABLISH and related commands provide several options for data backup and test data replication scenarios.

When a persistent relationship between a source and one to twelve targets is created with the FLASHCOPY ESTABLISH command, a copy of the source is *logically* created on the target and the command completes within a few seconds, regardless of the amount of data involved. When the FLASHCOPY ESTABLISH command has completed, the DASD subsystem has remembered the current state of the source and target extents and can begin physically copying the data from the source to the target(s) with a background copy process that is executed within the DASD subsystem. The decision to start the copying and at what speed to do it are based on available resources within the DASD subsystem.

How and Where Does the Hardware Read and Write to the Source and Target(s) During a Persistent Relationship?

There are four possible ways to access the source and target data tracks during a persistent FlashCopy relationship:

- Read from Source
  - Read source track and return it.
- Write to Source
  - If the source track has not yet been copied to the target (by either a prior write-to-source or background copy), read the original source track and write it onto the target.
  - After the source track has been copied to the target, replace the original source track with the new data.
- Read from Target
  - If the target track has been copied (by write-to-source or background copy) or modified (by write-to-target), read the target track and return it.
  - Otherwise, read the source track and return it.
- Write to Target
  - Mark the source track as no longer needing to be copied.
  - Write the data to the target track.

One key feature of this logic is background copy processing. When a persistent relationship is established with the FLASHCOPY ESTABLISH command, the normal action for the DASD subsystem is to begin copying all of the source tracks to all of the targets by executing a background copy process within the

DASD subsystem. After some period of time, the source tracks will have been copied to the target(s) and the *logical* copy is now a *physical* copy as well. However, for applications such as instant point-in-time backups and test data replication scenarios, this is not what we want. In these cases, what we want to do is to preserve the original version of the data. This is accomplished with the NOCOPY option of the FLASHCOPY ESTABLISH command which prevents the initiation of the background copying process.

Instant Point-in-Time Backups (disk-to-disk)

In this scenario, we want to create a complete instantaneous copy of a source disk onto a target disk. This is accomplished by executing:

```
FLASHCOPY ESTABLISH SOURCE vdev1 TARGET vdev2 NOSETARGET
```

Immediately after this command has completed, we have a logical copy (*vdev2*) that can be read or written without affecting the source (*vdev1*) and vice versa. You do not have to wait for the background copy to complete before you can access the copy on *vdev2*. The physical copying will be performed by the background copy process in the DASD subsystem, and its completion will be indicated by the remaining field showing a zero in the QUERY FLASHCOPY HARDWARE's response.

```
q flashcopy hardware 5101
              ---------SOURCE--------- ---------TARGET---------
SEQUENCE FLGS RDEV VOLSER CCCCCCCCC/HH RDEV VOLSER CCCCCCCCC/HH REMAINING/TOTAL
4A20BAC2 8800 5100 PACK01       100/00 5101 PACK02       100/00 0/150
4A20BAC2 8800 5100 PACK01       120/00 5101 PACK02       120/00 309/450
4A20BAC2 8800 5100 PACK01       150/00 5101 PACK02       170/00 0/150
```

Also, the FLASHCOPY WITHDRAW command will not execute successfully until the remaining count is zero:

```
FLASHCOPY WITHDRAW TARGET vdev2
HCPNFC2469E WITHDRAW failed because 309 tracks remain to be copied to vdev2.
Ready(2469);
```

The NOSETARGET option specifies that the target device cannot be a space-efficient device.

Instant Point-in-Time Backups (disk-to-tape)

In this scenario, we want to create an instantaneous copy of a source disk onto a tape. The advantage of this is the DASD subsystem will only copy source tracks that have been modified onto the target, thus reducing the load on the DASD subsystem. Using this feature while performing many concurrent disk-to-tape copies will help prevent storage controller overload errors. This is accomplished by executing:

```
FLASHCOPY ESTABLISH SOURCE vdev1 TARGET vdev2 NOCOPY
```

The NOCOPY option requests that the background copy operation not be initiated. Therefore, we have a logical copy that can be read or written, but not a physical copy.

Typically at this point the target (*vdev2*) would be copied to tape with your favorite disk-to-tape backup program, such as DDR or the CMS TAPE command.

After the tape backup has completed, you can destroy the target by issuing:

```
FLASHCOPY WITHDRAW TARGET vdev2 FORCE
```

The FORCE option specifies that you want to destroy the target — specifying FORCE will render the contents of the target unpredictable.

Test Data Replication

Another use of FlashCopy is to be able to create test data. An example is to make a copy of a production database for testing purposes. CHANGE RECORDING and RESYNC are only available on dedicated devices or full-pack minidisks.

To establish the copy, you would execute:

```
FLASHCOPY ESTABLISH SOURCE vdev1 TARGET vdev2 CHGRECORD NOCOPY
```

If you decide you want to synchronize the target with the current source, you would issue:

```
FLASHCOPY RESYNC SOURCE vdev1 TARGET vdev2
```

If you decide not to retain the resulting target DASD, you can issue:

```
FLASHCOPY WITHDRAW TARGET vdev2 FORCE
```

The target DASD will not be usable.

If you decide you want to retain the resulting target DASD, you can issue:

```
FLASHCOPY ESTABLISH SOURCE vdev1 TARGET vdev2 CHGRECORD NOCOPY NOSETARGET
FLASHCOPY BACKGNDCOPY SOURCE vdev1
FLASHCOPY WITHDRAW TARGET vdev2
```

Multiple LPAR FlashCopy Considerations

Persistent FlashCopy relationships can be created by z/VM or any other operating system running in any LPAR attached to the DASD subsystem. These relationships are visible to z/VM and persist until they are WITHDRAWN, even surviving system IPLs, DASD subsystem power-down/power-up, and DASD subsystem microcode loads.

CP-Owned Areas and FlashCopy

If a persistent FlashCopy relationship source extent is detected on a CP-owned volume during system IPL that is allocated as non-PERM space, warning message HCP2462I will be issued to the system operator's console.

If a persistent FlashCopy relationship target extent is detected on a CP-owned volume during system IPL that is allocated as non-PERM space, warning message HCP2463E will be issued to the system operator's console.

The ATTACH TO SYSTEM command will allow the attachment of volumes with persistent FlashCopy relationships if the relationships include non-PERM extents. However, messages HCP2462I and HCP2463E will be issued to the user issuing the command.

No further actions will be performed by z/VM on these extents, but care should be taken with these configurations to ensure that the FlashCopy relationship is not withdrawn or resynchronized while z/VM is using these extents, especially if the extent is the target of a FlashCopy relationship. Otherwise, system instability could occur as a result of the changes made to the volume by an external source.

The Class B QUERY FLASHCOPY HARDWARE command exists to facilitate the management of persistent FlashCopy relationships.

To remove all the target relationships on a volume you can use the ATTACH *rdev* TO * AS *vdev* command to attach the volume, or use DEFINE MDISK *vdev* 0 END *volser* (if the volume is attached to SYSTEM). Then, issue a FLASHCOPY WITHDRAW TARGET *vdev* command.

Space-Efficient Volumes

The IBM System Storage DASD Subsystem has introduced the concept of space-efficient volumes, available with Licensed Internal Code. This feature enables the partial provisioning of DASD devices by only allocating physical DASD subsystem resources to DASD cylinders that actually contain data. Commands such as QUERY DASD SPACE-EFFICIENT and QUERY CU DASD SELC exist to assist with the configuration and maintenance of the space-efficient pools and repositories. The QUERY DASD DETAILS command includes an indicator showing whether a device is an Extent Space Efficient (ESE) volume.

The RELSPACE command is used to return all extents of the specified VDEV on an ESE device to its associated space-efficient pool. The contents of those extents are also erased. A user can release the space of any virtual device to which they have write access. For more information, see *z/VM: CP Commands and Utilities Reference*.

Extent space-efficient volumes may be used for any CP area or guest disk use case. Guest (Linux) recognition of ESE capability and exploitation of space release functionality is provided for full pack minidisk, 1-END minidisk and dedicated DASD devices.

Track Space-Efficient Volumes with CP Areas Are Not Allowed

The original release of space-efficient volumes was implemented as track space-efficient (TSE) volumes. These volumes are only supported by z/VM as the targets for persistent FlashCopy relationships.

The use of track space-efficient volumes that contain PAGE, SPOOL, TDISK, Directory, Warmstart, Checkpoint, and PARM extents is not allowed because when a track space-efficient volume exhausts its allocated space, all I/O operations stop on the device. For PAGE and SPOOL, this would be very harmful to system stability.

When a CP-owned volume is detected on a track space-efficient DASD during system IPL, message HCP9050W is issued and the SYSTEM IPL stops.

HCP9050W
CP-OWNED volume is on a Space-Efficient DASD.

The ATTACH TO SYSTEM command will not allow track space-efficient volumes to be attached to the system. Message HCP2467E is issued.

HCP2467E
The CP-OWNED volume *volser* is allocated on Space-Efficient device *rdev*. Space-Efficient volumes are not allowed as CP-OWNED volumes.

These messages are not issued for extent space-efficient devices, as the process of formatting the devices for CP use causes the device to become fully provisioned. Therefore, in the CP Area use case, there is no risk of I/O operations being restricted due to a lack of available space in the space-efficient pool.

Space-Efficient Volumes and FlashCopy Backups

The IBM System Storage DASD Subsystem has introduced the concept of space-efficient persistent FlashCopy relationships, available with Licensed Internal Code. This feature enables the use of space-efficient DASD for FlashCopy targets.

Background copy processing is not available with track space-efficient targets because it would defeat the purpose of space-efficient DASD, which is to copy as little data as possible. By implication, disk-to-disk backups do not make sense and are not possible with track space-efficient targets. On the other hand, disk-to-tape backups work very well because the minimum necessary DASD cylinders are provisioned.

Because background copying is not available, the NOCOPY option is assumed. To establish a persistent FlashCopy relationship on a track space-efficient target, you can specify:

```
FLASHCOPY ESTABLISH SOURCE vdev1 TARGET vdev2
```

or

```
FLASHCOPY ESTABLISH SOURCE vdev1 TARGET vdev2 NOCOPY
```

When you have completed backing up *vdev2* to tape, you can destroy the target by issuing:

```
FLASHCOPY WITHDRAW TARGET vdev2 FORCE RELEASE
```

The FORCE option specifies that you are willing to destroy the target, and the RELEASE option tells the DASD subsystem to return the allocated tracks back to the repository for allocation to another device. If the optional RELEASE is not specified, the cylinders will continue to be provisioned to the target volume, which is probably not intended for space-efficient target devices.

The FLASHCOPY BACKGNDCOPY command is not available. This restriction does not apply for extent space-efficient volumes.

Space-Efficient Volumes and FlashCopy Test Data Replication

To establish the copy, you would issue this command:

```
FLASHCOPY ESTABLISH SOURCE vdev1 TARGET vdev2 CHGRECORD NOCOPY
```

This operates the same way as discussed in "Test Data Replication" on page 692, with the exception that you are not able to make changes to the target with the FLASHCOPY BACKGNDCOPY command and that the FLASHCOPY WITHDRAW FORCE option is required. The test data will be available until the persistent relationship is withdrawn by issuing:

```
FLASHCOPY WITHDRAW TARGET vdev2 FORCE RELEASE
```

The FORCE option specifies that you want to destroy the target. Note that specifying FORCE will render the contents of the target unpredictable.

Chapter 26. Defining and Managing SCSI FCP Disks

Overview of z/VM Support for SCSI Devices

z/VM supports SCSI FCP disk logical units (SCSI disks) for both system and guest use. SCSI disks can be used directly by a guest operating system when an FCP subchannel is dedicated to a guest. Such a guest must contain its own SCSI device driver – Linux on IBM Z is one such guest.

SCSI disks can also be used as emulated 9336 model 20 fixed-block-architecture (FBA) disks. CMS and CP rely almost exclusively on this emulated-FBA support for their SCSI usage. Specifically, this usage includes system paging, spooling, directory services, minidisks, and all other system functions and programming services that support FBA disks. Guests that support FBA disks (such as CMS, GCS, RSCS, Linux, and VSE) also can use SCSI disks through the emulated-FBA support, without requiring any specific SCSI support in the guests.

z/VM supports emulated FBA disks and their underlying SCSI disks up to 1 terabyte minus 1 page (2,147,483,640 512-byte blocks) in size, without regard to the capacity of real 9336 disks.

Note:

When CP uses FBA disks, space is allocated by 4096-byte pages. FBA DASD space allocated for use as directory, paging, and spooling must reside within the first 16,777,215 pages (64 GB minus 1 page) of the volume. Other types of CP allocations (TDSK, PERM, and PARM) can exist beyond the first 64 GB. CMS minidisks have absolute and practical capacity limitations. See the notes in CMS Restrictions in *z/VM: CMS Planning and Administration*.

An emulated FBA SCSI disk requires the following elements within its configuration definition:

FCP device number
A real device number for a subchannel associated with an FCP channel, providing access to the fibre-channel fabric.

Target worldwide port name (WWPN)
The unique worldwide port name associated with a target port on a SCSI controller.

Logical unit number (LUN)
The number of a specific logical unit (i.e. logical device) associated with the target port.

Refer to Figure 28 on page 698 for an illustration of these required elements.

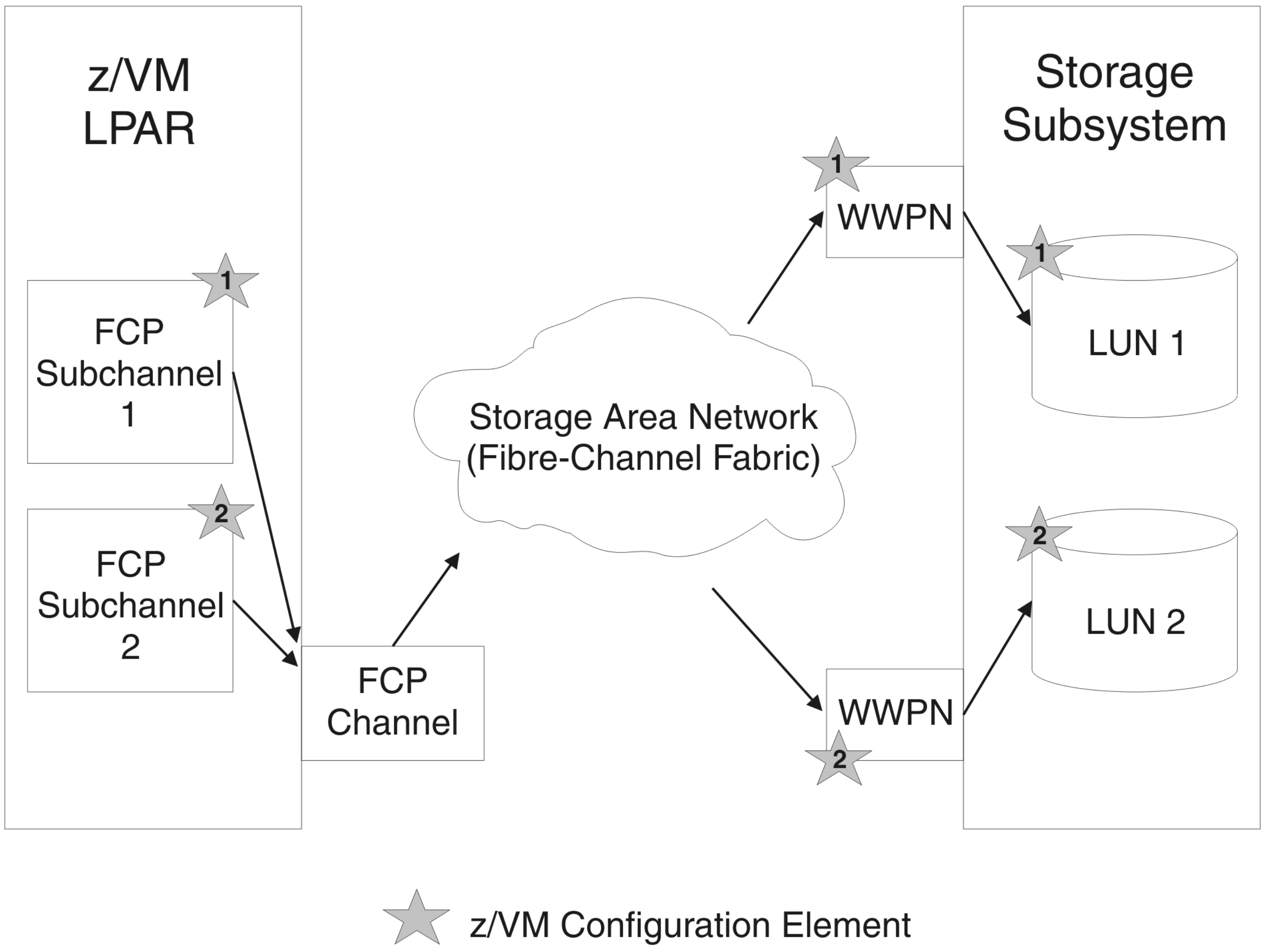


Figure 28. Required Elements for SCSI FCP definition

Figure 29 on page 699 illustrates the overall system architecture related to z/VM SCSI support. The figure shows how channel programs for emulated FBA disks are processed through an emulation layer and a SCSI driver into the fibre-channel fabric. The figure also shows how guests with their own SCSI support can use a dedicated FCP subchannel to access the fabric directly. Note that the elements described above in Figure 28 on page 698 are contained in the FCP Channel and Storage Area Network portions of Figure 29 on page 699.

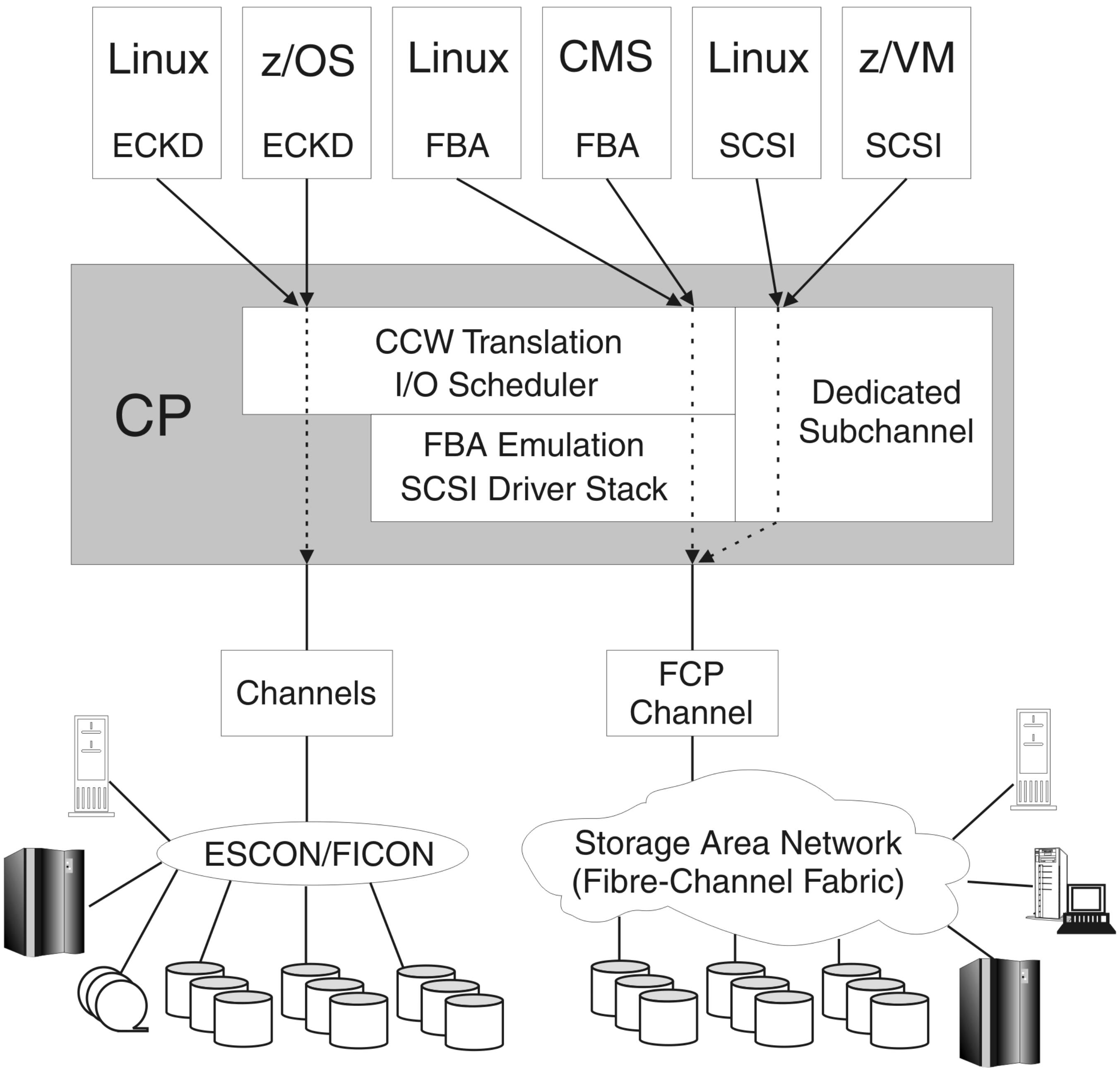


Figure 29. SCSI System Architecture

Defining SCSI Devices

Emulated FBA Disks on SCSI Disks

A SCSI device is defined to the z/VM system by specifying an EDEVICE statement in the system configuration file (see *z/VM: CP Planning and Administration*), by issuing a CP SET EDEVICE command (see *z/VM: CP Commands and Utilities Reference*), or by using z/VM's HCM and HCD support (see z/OS and z/VM: Hardware Configuration Manager User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/eequ100_v2r5.pdf) and *z/VM: I/O Configuration*). The EDEVICE statement, SET EDEVICE command, and the HCM and HCD programs all provide the parameters to identify the SCSI device (FCP device number, target WWPN, and LUN) to CP. The emulated FBA disk is assigned a real device number and becomes associated with a SCSI disk through an FCP device, which also has a real device number. The device number of an emulated FBA disk must not conflict with the device number of any real device in the z/VM system. That is, all emulated FBA disks and all real devices must have unique real device numbers, even though the emulated FBA disks are not real devices.

When defining emulated FBA disks to represent SCSI disks, there must be a one-to-one relationship between each FBA disk and its underlying SCSI disk. A path to a specific SCSI disk comprises the three elements described in "Overview of z/VM Support for SCSI Devices" on page 697: the FCP device number, target WWPN, and LUN. You can define multiple paths to the SCSI disk associated with an emulated FBA disk – these paths should be routed through different network components to achieve the greatest availability for the disk. In other words, the different paths must use different FCP channels, the channels should be connected to different switches, and the paths should end at different target ports of the SCSI controller. You might want to keep the paths completely separate by means of appropriate zoning within your fabric.

All paths defined for an emulated FBA disk should represent physical paths through the fabric to the same real SCSI disk. If you define multiple paths for an emulated FBA disk but the paths go to different SCSI disks, or if you define more than one emulated FBA disk with the same underlying real SCSI disk, or if you define multiple paths to the same SCSI disk using the same FCP channel, then unpredictable results and/or data-integrity problems could occur.

Once the emulated FBA disk is defined, it is managed on z/VM like a real 9336 FBA disk. CP commands such as VARY, ATTACH, and QUERY execute as if the emulated disk were a real FBA disk. This also applies to user-directory and system-configuration-file statements.

The following EDEVICE commands can be used to manage emulated FBA disks:

- To create, modify, or clear an emulated-device definition, use the SET EDEVICE command.
- To delete an emulated-device definition, use the DELETE EDEVICE command.
- To query an emulated-device definition, use the QUERY EDEVICE command.

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

The SCSIDISC utility can be used to dynamically discover all SCSI disks, and their associated paths, accessible by a virtual FCP device. EXPLORE FCP is used to test FCP subchannels and associated WWPN paths in a more persistent manner than SCSIDISC. See SCSIDISC and EXPLORE FCP in *z/VM: CP Commands and Utilities Reference*.

Note: The SET EDEVICE and DELETE EDEVICE commands are not allowed when HCM and HCD are controlling the z/VM software configuration.

Real SCSI Disks

The following commands can be used to manage an FCP device dedicated to a guest for direct access to SCSI disks through a SCSI device driver:

- ATTACH
- DETACH
- QUERY FCP
- SET LOADDEV
- QUERY LOADDEV

See *z/VM: CP Commands and Utilities Reference* for more information on these commands.

Restrictions and Caveats for Using SCSI Devices

The following restrictions apply to emulated SCSI devices:

- For CP volumes (formatted with CPFMTXA), space that is allocated by CP for use as directory, paging, and spooling must reside within the first 16,777,215 pages (64 GB minus one page) of the volume.
- Blocks 0-31 of a CP-formatted FBA volume contain the standard label, volume allocation map, and other reserved space. Do not write to blocks 0-31. User minidisks must start at block 32 or higher.

SALIPL writes the Stand-Alone Program Loader (SAPL) up to block 207. If the CP-formatted FBA volume was set up by using the SALIPL utility, blocks 0-207 must be reserved, and user minidisks must start at block 208 or higher.

- As with all FBA minidisks, Reserve and Release CCW commands that are issued to an emulated FBA disk are supported only among guests within the same z/VM system image. Reserve and Release CCW commands are rejected if a guest operating system has access to a dedicated FBA disk or to an emulated FBA full-pack minidisk that is defined as shared among multiple z/VM images.
- A single FCP channel can be shared by multiple logical partitions and multiple virtual machines. However, the operating systems in those logical partitions and virtual machines must not attempt to share a real SCSI device (as specified by a WWPN and LUN pair). All such operating systems appear to the SCSI controller as the same host system (or SCSI initiator). Problems with concurrent-write access to the disk and with error-recovery procedures can result. A real SCSI device can be shared (if it makes sense for the particular device) if each operating system has its own path through separate FCP channels to the SCSI device.
- Unpredictable results can occur if you assign different logical unit numbers (LUNs) to the same logical device within a SCSI controller. Shared access to a device can occur without being detected.

Using N_Port Identifier Virtualization (NPIV) for FCP Channels

NPIV is supported by IBM Z FICON channel cards in FCP mode. Additionally, NPIV requires support in the entry switch used to attach the channel to the SAN fabric. No support is required in any other switches or devices attached to the SAN.

z/VM exploits the adapter capability to define multiple virtual FCP channels, each with its own unique fibre channel port name and fibre channel identifier (FC_ID). By assigning distinct virtual port names to different guests, the guests can use the virtual FCP channels as if they were using dedicated physical FCP channels. Access controls based on the virtual port names may be applied in the SAN fabric using standard mechanisms like zoning in the switches and logical unit number (LUN) masking in the storage controllers, thereby providing access control at the FCP *subchannel* level.

Previously, all FCP subchannels shared the common WWPN burned into the associated FCP adapter. Within the SAN fabric, therefore, the actual I/O initiator (a specific subchannel) could not be determined because the initiator was always the WWPN of the FCP adapter. Access control could only be managed at the *adapter* level.

z/VM support of NPIV-enabled hardware is automatic and transparent. No special initialization or command is required. The QUERY command will display the hardware-assigned WWPN of NPIV-enabled FCP subchannels. See *z/VM: CP Commands and Utilities Reference* for more information on this command.

The procedure for enabling NPIV support on the FICON adapter can be found in the *Support Element Operations Guide* for your server. Refer to the associated product documentation to enable NPIV support on a switch.

Chapter 27. Defining and Managing NVMe Devices

Non-Volatile Memory Express (NVMe) devices that are connected via PCI Express (PCIe) adapters can be defined and managed as Fixed-Block Architecture (FBA) emulated devices (EDEVICEs). All host and guest FBA functions are supported except for those functions that require stand-alone support such as checkpoint and warm start.

Emulated FBA Disks on NVMe Devices

An NVMe device that is associated with the ID of a real PCIe function can be defined as one or more emulated FBA devices. An emulated device can be defined as a base device or an alias device.

An NVMe device is defined to the z/VM system by specifying an EDEVICE statement in the system configuration file or by issuing a CP SET EDEVICE command. The EDEVICE statement and the SET EDEVICE command identify a PCIe function that is associated with the NVMe device. The first RDEV that is associated with an NVMe PCIe function is defined as a HyperParallel Access Volume (HyperPAV) base. Subsequent RDEVs that are associated with the NVMe PCIe function are defined as HyperPAV bases or aliases and are used to perform I/O operations concurrently.

Note: Real FBA HyperParallel Access Volumes do not exist and no architecture describes their characteristics or behavior.

Restriction: Guests cannot have virtual or dedicated HyperPAV aliases for emulated FBA disks that are defined on NVMe devices.

After the emulated FBA disk is defined, it is managed on z/VM like a real 9336 FBA disk. CP commands such as VARY, ATTACH, and QUERY run as if the emulated disk were a real FBA disk. User directory statements and system configuration file statements also apply as if the emulated disk were a real FBA disk.

The following EDEVICE commands can be used to manage emulated FBA disks:

- To create, modify, or clear an emulated-device definition, use the SET EDEVICE command.
- To delete an emulated-device definition, use the DELETE EDEVICE command.
- To query an emulated-device definition, use the QUERY EDEVICE command.

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

The PCIFUNCTION commands (DEFINE, DELETE, DETACH, MODIFY, QUERY, RESET, SET, and VARY) can be used to manage the NVMe PCIe function.

See the following command topics in *z/VM: CP Commands and Utilities Referencez/VM: CP Commands and Utilities Reference*:

- DEFINE PCIFUNCTION
- DELETE PCIFUNCTION
- DETACH PCIFUNCTION
- MODIFY PCIFUNCTION
- QUERY PCIFUNCTION
- RESET PCIFUNCTION
- SET PCIFUNCTION
- VARY PCIFUNCTION command

For more information about the EDEVICE statement, see "EDEVICE Statement" on page 139.

For more information about PCIe functions, see Chapter 17, "Using PCIe Functions for z/VM Guests," on page 439.

For more information about Parallel Access Volumes (PAV) aliases and HyperParallel Access Volumes (HyperPAV), see the following topics:

- "Using IBM Parallel Access Volumes" on page 678 and "Using IBM HyperParallel Access Volumes" on page 683
- Parallel Access Volumes (PAV) and HyperPAV for guest I/O to minidisks and HyperPAV for the Paging Subsystem in *z/VM: Performance*

Storage Capacity of Emulated Base Devices on an NVMe Device

z/VM determines the storage capacity and maximum number of emulated base devices that can be defined on an NVMe device. z/VM divides the total storage capacity of an NVMe device into the least number of equal-capacity segments that are no greater than the emulated FBA capacity limitation. The QUERY EDEVICE DETAILS command displays the number of base EDEVICEs that would completely consume the capacity of an NVMe device. The SEGMENT line shows the number of the current device segment and the maximum allowed number of segments. The capacity of one segment determines the capacity of any defined base RDEV on the NVMe device.

Consider the example of a 12.8 TB NVMe device. The NVMe device nominal capacity is 12.8×10^{12} bytes, which is slightly less than 11.65×2^{40} bytes (slightly less than 11.62 TB in the z/VM TB convention). z/VM divides the NVMe device into 12 segments. The storage capacity of each segment is slightly less than 2^{40} bytes and does not exceed the capacity limitation for an emulated FBA device.

Note:

- The TB convention for an NVMe device differs from the TB convention for z/VM.
  - The TB convention for an NVMe device is 1 TB = 10^{12} bytes.
  - The TB convention for z/VM is 1 TB = 2^{40} bytes. (A newer, IEC terminology for 2^{40} bytes is *tebibyte* (TiB), but z/VM uses legacy terminology *terabyte* (TB)).
  - The capacity limitation for an emulated FBA device is 2^{40} bytes less one page (one page = 4096 bytes).

The ZPCIMON EXEC Utility

NVMe solid-state drives are subject to wear and have a limited lifetime. A device provides health information that can be used to gauge whether the device needs to be replaced. Monitoring device health is a customer responsibility. The ZPCIMON EXEC sample utility is provided to monitor the health of NVMe devices.

ZPCIMON monitors an individual NVMe EDEVICE by issuing a QUERY PCIFUNCTION DETAILS command every 15 minutes. The utility parses the health-related information in the command response. ZPCIMON reports the values of device health indicators on its console and issues a message when device health issues are detected. The exec reports device health issues by using the SET PCIFUNCTION REPORT command.

To run ZPCIMON, set up a dedicated CMS virtual machine to monitor health of a single device. Dedicate a separate CMS virtual machine for each device that you want to monitor. Copy the ZPCIMON SAMPEXEC file from the samples disk to a common tools disk. Use the CMS virtual machine's PROFILE EXEC or some similar mechanism to issue the ZPCIMON command and specify the NVMe device's PCIe function ID as the command's only operand. Autolog the guest after the NVMe EDEVICE is defined. You can modify the sample utility to provide capabilities such as recording the log information on disk, managing log file rotation, and informing appropriate personnel when a device health issue is detected.

For more information, see the following topics:

- AUTOLOG in *z/VM: CP Commands and Utilities Reference*
- "EDEVICE Statement" on page 139

Restrictions and Caveats for using NVMe Devices

The following restrictions and limitations apply to emulated FBA devices:

- When CP uses FBA disks, space is allocated by 4096-byte pages. FBA DASD space allocated for use as directory, paging, and spooling must reside within the first 16,777,215 pages (64 GB minus 1 page) of the volume. Other types of CP allocations (TDSK, PERM, and PARM) can exist beyond the first 64 GB. CMS minidisks have absolute and practical capacity limitations. See the notes in CMS Restrictions in *z/VM: CMS Planning and Administration*.
- Blocks 0-31 of a CP-formatted FBA volume contain the standard label, volume allocation map, and other reserved space. Do not write to blocks 0-31. User minidisks must start at block 32 or higher.

  SALIPL writes the Stand-Alone Program Loader (SAPL) up to block 207. If the CP-formatted FBA volume was set up by using the SALIPL utility, blocks 0-207 must be reserved, and user minidisks must start at block 208 or higher.
- FBA emulated disks on NVMe devices cannot be shared among multiple z/VM images.
- A combination of up to 127 base and alias devices can be defined for a single NVMe PCIe function.

The following caveats apply to emulated FBA devices:

- Because of the potentially large capacity of an NVMe device, it might be necessary to define two or more emulated devices to completely consume the capacity of a single NVMe device. The number of EDEVICEs that would completely consume the NVMe device capacity is shown in the response to a QUERY EDEVICE DETAILS command for a base EDEVICE. The SEGMENT line shows the number of the current device segment and the maximum allowed number of segments. When all segments are assigned, the complete NVMe device capacity is assigned.
- CP disk I/O can be classified into two types: paging I/O and minidisk I/O. There can be issues when two I/O types are directed to a particular HyperPAV-enabled logical control unit (LCU). Problems can occur when I/O types are mixed on a control unit and I/O rates are high enough to routinely exhaust the alias pool. It is possible for one type of I/O to starve another I/O type's exploitation of the LCU's aliases. The SET CU command is not valid for emulated devices and cannot be used to set the relative share of system-attached HyperPAV aliases that are used for minidisk I/O and paging I/O. To watch the alias pool, use Performance Toolkit FCX327 HPALIAS or equivalent.

See SET EDEVICE and SET CU in *z/VM: CP Commands and Utilities Reference* and FCX327, HyperPAV Alias Activity Screen – HPALIAS in *z/VM: Performance Toolkit Reference*. For performance considerations of using aliases, see Parallel Access Volumes (PAV) and HyperPAV for guest I/O to minidisks and HyperPAV for the Paging Subsystem in *z/VM: Performance*.

Chapter 28. Device Encryption Planning

Using Tape Encryption

Guest operating systems, such as CMS, that are not capable of enabling the hardware encryption available with the 3592 Model E05 tape drive are able to use new z/VM facilities that enable the encryption on behalf of the guest. Guest operating systems that do support tape encryption, such as z/OS with proper service, will be able to do so without interference from z/VM.

There are other components necessary besides z/VM and an encryption-capable 3592 Model E05 tape drive. The control unit needs to able to communicate with an encryption key manager (EKM) [1] and an associated key store(s). z/VM expects the EKM communication to take place via TCP/IP, referred to as an “out-of-band” connection. The EKM and associated key store(s) will maintain a list of Key Encrypting Key (KEK) Labels, where each KEK Label is up to 64 characters long (including spaces) and represents an RSA public/private key pair. Typically, the public key is used for encryption, while the private key is used for decryption.

The KEK Labels are written to the tape cartridge to facilitate the decryption process later on. A KEK Label can either be written directly or as a hash of the associated public key. The former will require the decryption process to have an identical KEK Label in the key store that represents the necessary keys, while the latter will allow for more flexibility in transporting encrypted tapes to other parties by allowing different KEK Labels to point to the same RSA key pair.

Data is sent to the tape drive as plain text, where a symmetric Data Key (DK) is generated by the EKM to perform a high-speed encryption of the data before it is written to the cartridge. The DK itself is encrypted with the public key of the specified KEK, creating what is known as an Externally Encrypted Data Key (EEDK). Two EEDKs are created out of specified KEK Labels and written to the tape cartridge. If only one KEK Label is specified when enabling the encryption environment, it is used to create both EEDKs. In order to decrypt the data, the recipient must hold the correct private key necessary to decrypt the EEDK, and thus be able to extract the DK that is used to decrypt the rest of the cartridge. It should be noted that the volume label for the cartridge will be written with a key that is known to the tape drive, permitting decryption of the label to take place without conversing with the EKM. This will allow cartridges to be identified and help determine what keys were used for the encryption, without compromising the security that protects the rest of the data.

The use of KEK Labels for the creation of EEDKs is only necessary when first creating an encrypted tape. Upon mounting a previously encrypted tape cartridge, the tape drive initiates communication with the EKM in hopes that the necessary keys are present to decrypt the EEDKs and thus the entire cartridge. If the required keys are not present, the cartridge cannot be decrypted and an encryption key failure will be presented upon a read operation. If the required keys are present in the EKM, then the cartridge will be decrypted and presented to the host as a regular tape.

If a newly mounted tape cartridge is moved past the load point in order to append additional data, that data will be written in the same format as the existing data on the cartridge. This helps provide consistency of the data set on the cartridge, as the cartridge will remain entirely encrypted or plaintext. Only when the tape is written to at the load point will the specified KEK Labels (or lack thereof) be used to change how the data exists on a cartridge.

z/VM Support

Note: The information provided below is only to be used in conjunction with those guest operating systems that are not able to enable the hardware encryption environment themselves. Attempts to combine them are likely to create hardware problems such as missing interrupts.

[1] See IBM System Storage Tape Enterprise Key Manager, Introduction, Planning and User Guide, GA76-0418.

z/VM maintains a list of key aliases, defined via the SET KEYALIAS command, that represent the EKM's KEK Labels to be used by z/VM. Each key alias is up to 32 characters long (including spaces) and contains the KEK Label as well as an encoding mechanism of how that KEK Label will be used to create an EEDK. A KEK Label can be used directly ("LABEL") or hashed ("HASH") when creating the EEDKs. The latter may be more convenient if the recipient of the encrypted cartridge has a different KEK Label string defined to represent the matching keys.

This information can be recalled with the QUERY KEYALIAS command for either a single alias, or for the entire list of known aliases.

```
set keyalias cow label keylabel 'moo moo moo'
set keyalias duck hash keylabel 'quack quack'
set keyalias 'old macdonald' label keylabel 'Had a Farm'

q keyalias
KEYALIAS: (L) COW
        = MOO MOO MOO
KEYALIAS: (H) DUCK
        = QUACK QUACK
KEYALIAS: (L) OLD MACDONALD
        = HAD A FARM
Ready; T=0.01/0.01 15:56:39
q keyalias 'old macdonald'
KEYALIAS: (L) OLD MACDONALD
        = HAD A FARM
Ready; T=0.01/0.01 15:57:19
```

Figure 30. SET/QUERY KEYALIAS

When dedicating a tape to a guest, the key aliases can be specified on the ATTACH command in order to select the KEK Labels to use when creating the EEDKs. If an alias is not recognized, an error will be presented and the ATTACH command will fail. If no key aliases are specified a set of default KEK Labels, defined in the EKM, will be used to create the EEDKs.

At the time of the ATTACH, the tape drive must be either unloaded or at beginning-of-tape in order to establish the encryption environment. Thus, a tape cartridge is guaranteed to have a consistent set of data that is either completely encrypted with a unique set of EEDKs, or completely unencrypted. Attempts to ATTACH a tape drive with encryption settings that do not meet these criteria will fail with an error message.

Guest operating systems that use the z/VM facilities to enable encryption are able to use the MULTIUSER option on ATTACH. However, all guests using the shared tape device need to use the same encryption settings, in order to provide a consistent environment for enabling the encryption.

The encryption settings specified on an ATTACH will persist until the drive is detached and unloaded. If a DETACH is issued with the LEAVE option, the encryption settings will remain in place along with the mounted cartridge. This will allow a subsequent ATTACH to be issued without any encryption settings, but allowing the encryption environment to be passed to a target user. In the case of the ATTACH of shared tape, the encryption settings persist until all instances are DETACHed, and the device is marked as free again.

```
att 181 * key
TAPE 0181 ATTACHED TO JOECOOL 0181
Ready; T=0.01/0.01 16:46:29

det 181
TAPE 0181 DETACHED
Ready; T=0.01/0.01 16:46:32

att 181 * multi key cow duck
TAPE 0181 ATTACHED TO JOECOOL 0181
Ready; T=0.01/0.01 16:46:35

att 181 * multi key cow pig
HCPATR1128E Device 0181 not attached; a mismatch in hardware encryption settings
            was detected.
Ready(01128); T=0.01/0.01 16:47:04
```

Figure 31. ATTACH

The SET RDEVICE FEATURE command can be used to enable the encryption environment without making changes to the ATTACH command. As with ATTACH this command must also be issued when the drive is unloaded or at beginning-of-tape, but also must be issued before it is dedicated to a guest. This will permit the encryption environment to be enabled through the use of library or tape managers that issue a regular, unmodified, ATTACH command. Up to two aliases can be specified per SET RDEVICE FEATURE command, as defined by the SET KEYALIAS command. If none are specified, the EKM's default keys will be used.

```
set rdev 7e2 feature key
HCPZRP6722I Characteristics of device 07E2 were set as requested.
1 RDEV(s) specified; 1 RDEV(s) changed; 0 RDEV(s) created
Ready; T=0.01/0.01 08:34:04

set rdev 7e2 feature key cow duck
HCPZRP6722I Characteristics of device 07E2 were set as requested.
1 RDEV(s) specified; 1 RDEV(s) changed; 0 RDEV(s) created
Ready; T=0.01/0.01 08:37:21
```

Figure 32. SET RDEVICE FEATURE

There is new information available with different QUERY responses that will provide encryption-related information to the issuer. This includes the Class B `QUERY TAPES DETAILS <rdev>`, and the Class G `QUERY VIRTUAL TAPES` and `QUERY VIRTUAL <rdev> DETAILS` commands.

All three variations will include text that indicates which drives are capable of encryption. The two commands with DETAILS options will provide additional information regarding the KEK Labels in use on the tape device. If any encryption settings were defined with the SET RDEVICE FEATURE command, those settings will be displayed under the heading "INACTIVE KEY LABELS." Once an ATTACH command is issued, those same settings will reside under the heading "ACTIVE KEY LABELS" if ATTACH was performed unmodified. Otherwise, the active heading will contain the encryption information specified on the ATTACH command. If either heading indicates "DEFAULT," the default keys defined in the EKM are being used.

The above statement is true when a tape is written to immediately after being mounted and attached with encryption information. If, however, the first operation to the mounted tape is a read, any EEDKs that were stored on the tape are decrypted and the resulting Data Key is used to read the contents of the tape or to append additional data. Issuing a QUERY command with a DETAILS option at this point will display the KEK Labels and encoding mechanism that were present on the tape cartridge under the "ACTIVE KEY LABEL(S)" heading, and the information specified through ATTACH will instead be listed under an "ATTACHED KEY LABEL(S)" heading. This distinction is only displayed when what is currently in use and what was specified on ATTACH are found to be different. If the mounted tape cartridge is not encrypted, the "ACTIVE KEY LABEL(S)" will indicate "NONE."

If you are careful, you can use this information to identify what KEK Labels are actually used when the EKM default keys are requested by ATTACH. Unfortunately, no indication is returned that a given KEK Label is marked as a default, so you might get different KEK Label information as a “default” key if the tape was not initially written after mounting with the default keys option.

```
q v tapes
TAPE 0181 ON DEV  07E2 3590 R/W SUBCHANNEL = 0008 ENCRYPTION CAPABLE
Ready; T=0.01/0.01 16:42:02

q v 181 details
TAPE 0181 ON DEV  07E2 3590 R/W SUBCHANNEL = 0008 ENCRYPTION CAPABLE
  ACTIVE KEY LABEL(S):
    (H) the first mighty key label
    (L) the second mighty key label
  INACTIVE KEY LABEL(S): DEFAULT

q tapes details 7e2
TAPE 07E2 SEQUENCE NUMBER E0010 LIBPORT 1 ENCRYPTION CAPABLE
  ACTIVE KEY LABEL(S):
    (H) the first mighty key label
    (L) the second mighty key label
  INACTIVE KEY LABEL(S): DEFAULT

q tapes details 7e3
TAPE 07E2 SEQUENCE NUMBER E0010 LIBPORT 1 ENCRYPTION CAPABLE
  INACTIVE KEY LABEL(S):
    (H) the first mighty key label
    (L) the second mighty key label
```

Figure 33. Variations of QUERY output

The z/VM DASD Dump Restore (DDR) utility also supports Tape Encryption for 3592 drives, but utilizes its own unique externals in order to run in the absence of an underlying z/VM system. The Input/Output control statement includes a new option, "KEY," that will cause the output device to be enabled for device encryption. This option is only valid on the output statement, as it has no meaning on an input device. A new HASH/LABEL control statement has been added to define the encoding mechanism used on the tape. These statements (up to two can be specified) determine what key labels are to be used for encrypting the tape, and how they will be used (as a label, or as a hash of the public key). These control statements are optional, and if not included, the default keys in the EKM will be utilized for encryption. The following example dumps all data from the volume labeled SYSRES onto the tape mounted on unit 181 and the data is encrypted using key e3rw33rssesyqypsftqqpx0539.

```
input 191 3390 sysres
label1 e3rw33rssesyqypsftqqpx0539
output 181 3592 (key
dump all
```

Figure 34. Encrypted data dump

Tape Rekey

An extension to the tape encryption support, known as a tape rekey operation, has been made available with newer levels of Control Unit microcode for the 3592 Model E05 tape drive. This operation will decrypt the EEDKs stored on an encrypted tape cartridge, and encrypt them with a different set of KEK Labels before storing them back on the tape. The result is a different set of access rights being given to a tape cartridge, without requiring a time-consuming tape-to-tape copy operation that would otherwise be required.

The CP SET TAPE command is used to initiate the rekey operation against an encrypted tape and accepts one or two key aliases as input. Obviously, the affected Control Unit needs to be able to access both the old and new KEK Labels in order for the rekey operation to complete successfully. However, it also needs to be issued to an encrypted tape cartridge that is mounted and positioned at Beginning of Tape. The

QUERY responses that contain detailed information about the KEK Labels will reflect this change after one has been made. It should be noted that in this case, the aforementioned KEK Labels that fall under the “ATTACHED KEY LABEL(S)” heading will likely not be used until a new tape is mounted in the drive.

The EEDKs are the only pieces that are actually changing on the tape, so the actual format of the tape does not matter to z/VM. Therefore, if you want, z/VM can rekey a tape on behalf of its guest.

Pervasive Encryption for z/VM

Pervasive Encryption refers to end-to-end cryptography introduced on the IBM z14. This generation of hardware introduces improved performance and a more global scope for encryption on the platform. Pervasive Encryption means you can enable encryption everywhere.

For z/VM, encryption in the hypervisor layer is handled by the z/VM Control Program (CP). Encryption is enabled for a particular CP subsystem using Pervasive Encryption.

With the PTF for APAR VM65993, the paging subsystem can perform encryption. This encryption protects guest data, specifically virtual machines' primary address space, data space, and virtual-disk-in-storage (VDISK) pages, paged out of memory and onto CP-owned paging volumes (or similar devices). This encryption is handled by an ephemeral key generated during the z/VM system IPL process. This key is wrapped after creation and never exists in clear memory.

Enabling and Disabling Encryption

Encryption can either be set in the system configuration file or set dynamically with a CP command. Each method has its benefits. For your z/VM system, using the SET ENCRYPT PAGING ON|OFF command might be more appropriate than modifying the system configuration file. For more information, see the SET ENCRYPT command description in *z/VM: CP Commands and Utilities Reference*.

Precautions for Using the ENCRYPT PAGING REQUIRED Option

When enabling encryption, the REQUIRED option of the ENCRYPT PAGING configuration statement should be used with caution. When you specify this option for a particular service, the encryption setting for that service cannot be changed without a system IPL. Also, a system a configuration file that specifies REQUIRED will cause CP not to IPL if the hardware support for encryption is not enabled. Instead, system IPL stops and a disabled wait state HCP1393W is loaded. No easy method is available for the system operator to bypass the REQUIRED option; this is done intentionally to prevent a system from running in a state that conflicts with your installation's stated security policy. Although the characteristics of the REQUIRED option ensure security for systems, this rigidity can cause complications in disaster recovery scenarios when a common system configuration is used to run a secure system on back-level hardware.

IBM recommends you take the following steps before specifying ENCRYPT PAGING REQUIRED in the system configuration file:

- Test the ENCRYPT PAGING ON option extensively before switching to using the ENCRYPT PAGING REQUIRED option. Configuration tolerance should be validated before locking the encryption service, and workload performance testing should be done to validate that the "cost" of encryption is acceptable. Your tests should include a workload that pushes storage overcommitment to your expected maximums. Also, you should validate the CPU cost by examining the appropriate monitor data. Testing should include production systems and disaster recovery sites.
- Consider setting ENCRYPT PAGING ON in the system configuration file for all systems, then automating REQUIRED as part of the system IPL. This allows the system to IPL, whether the required hardware is present or not, and locks encryption in place if the appropriate hardware is available. Because paging can occur early in the system IPL process, recommended points of automation include either of the following:
  - Specify SET ENCRYPT PAGING REQUIRED on a COMMAND statement in the system operator virtual machine (OPERATOR).
  - Specify SET ENCRYPT PAGING REQUIRED as a CP command executed in the PROFILE EXEC of the AUTOLOG1 virtual machine, prior to starting an External Security Manager.

- Prepare an alternate system configuration file for back-up or emergency purposes. An alternate configuration file on the same PARM volume can be specified with the FN= and FT= IPL parameters.
- In shared configuration files, use record qualifiers as needed to limit the ENCRYPT configuration statement to systems running on the required hardware.

For more information, see "ENCRYPT Statement" on page 152, or see the SET ENCRYPT command description in *z/VM: CP Commands and Utilities Reference*.

Tracking the Encryption Settings

Auditing of the hypervisor-level encryption state is important for compliance purposes. If security requirements demand 100% encryption of paging data, then demonstrating the state of the system, along with the tracking of changes, is vital to proving conformance to policy.

- The QUERY ENCRYPT command displays the current state of a given encryption service as well as the IPL setting. When encryption is enabled, the algorithm in use is displayed. For more information, see the description of the QUERY ENCRYPT command in *z/VM: CP Commands and Utilities Reference*.
- An informational message is issued to the system operator when the ENCRYPT setting for a CP service is a successfully changed. This message can be used for automation purposes by virtual machines or products with Observer access to the system operator. See the description of message HCP1394I in *z/VM: CP Messages and Codes*.
- An External Security Manager such as RACF for z/VM can audit any z/VM command, including QUERY and SET subcommands. Tracking successful instances of the SET ENCRYPT command provides a permanent record of any tampering with the SET ENCRYPT setting. The resulting logs can be incorporated into other security-relevant data. For more information, see *z/VM: RACF Security Server Auditor's Guide*.
- Monitor records are available to track the initial ENCRYPT state of the system, the current state, the user ID that issued the SET ENCRYPT command, and specific data such as paging encryption details and CPU utilization for encryption processes. For more information on updated monitor records, see *z/VM: Performance*.

Deciding Whether to Enable Encryption

Deciding whether to enable encryption depends on the overall security posture of your a system, as well as the mandates of your company's security policy. Pervasive Encryption on the IBM z14 represents a "full stack" picture, from the hardware up to the application and database level. If the workloads running in your z/VM environment do not require Encrypted Paging, or if only one or two guests require it, then enabling encryption in the hypervisor layer may not prudent. Encryption should be enabled for the guest, hypervisor, and/or hardware in a way which meets the compliance requirements governing the system.

Part 5. Single System Image Clusters Planning and Administration

Chapter 29. Setting Up z/VM Single System Image Clusters

A z/VM single system image (SSI) cluster is a multisystem environment in which the z/VM systems can be managed as a single resource pool and running virtual servers can be moved from one system to another.

This chapter describes:

- "A z/VM SSI Environment" on page 715
- "Major Attributes of a z/VM SSI Cluster" on page 716
- "Planning for a z/VM SSI Cluster" on page 728
- "Creating a z/VM SSI Cluster" on page 735
- "z/VM SSI Cluster Operation" on page 736
- "Implications for Vendor Products and Customer Applications" on page 741
- "System Configuration File for a z/VM SSI Cluster" on page 742

A z/VM SSI Environment

A z/VM SSI cluster consists of up to 8 z/VM systems (*members*) in an Inter-System Facility for Communications (ISFC) collection. Figure 35 on page 716 shows the basic structure of a cluster with 4 members. The cluster is self-managed by CP using ISFC messages that flow across channel-to-channel connections between the members. All members can access shared DASD volumes, the same Ethernet LAN segments, and the same storage area networks (SANs). For requirements and restrictions, see "Planning for a z/VM SSI Cluster" on page 728.

What a z/VM SSI Cluster Provides

A z/VM SSI cluster can provide several benefits:

- Facilitates the horizontal growth of z/VM workloads.
- Allows running virtual servers (guest virtual machines) to be moved from one member to another, a process known as *live guest relocation*.

  Note: This function is currently supported for Linux guests only.
- Allows z/VM and hardware maintenance to be less disruptive to workloads.
- Allows less disruptive workload balancing.
- Eases the deployment and maintenance of multiple z/VM images.

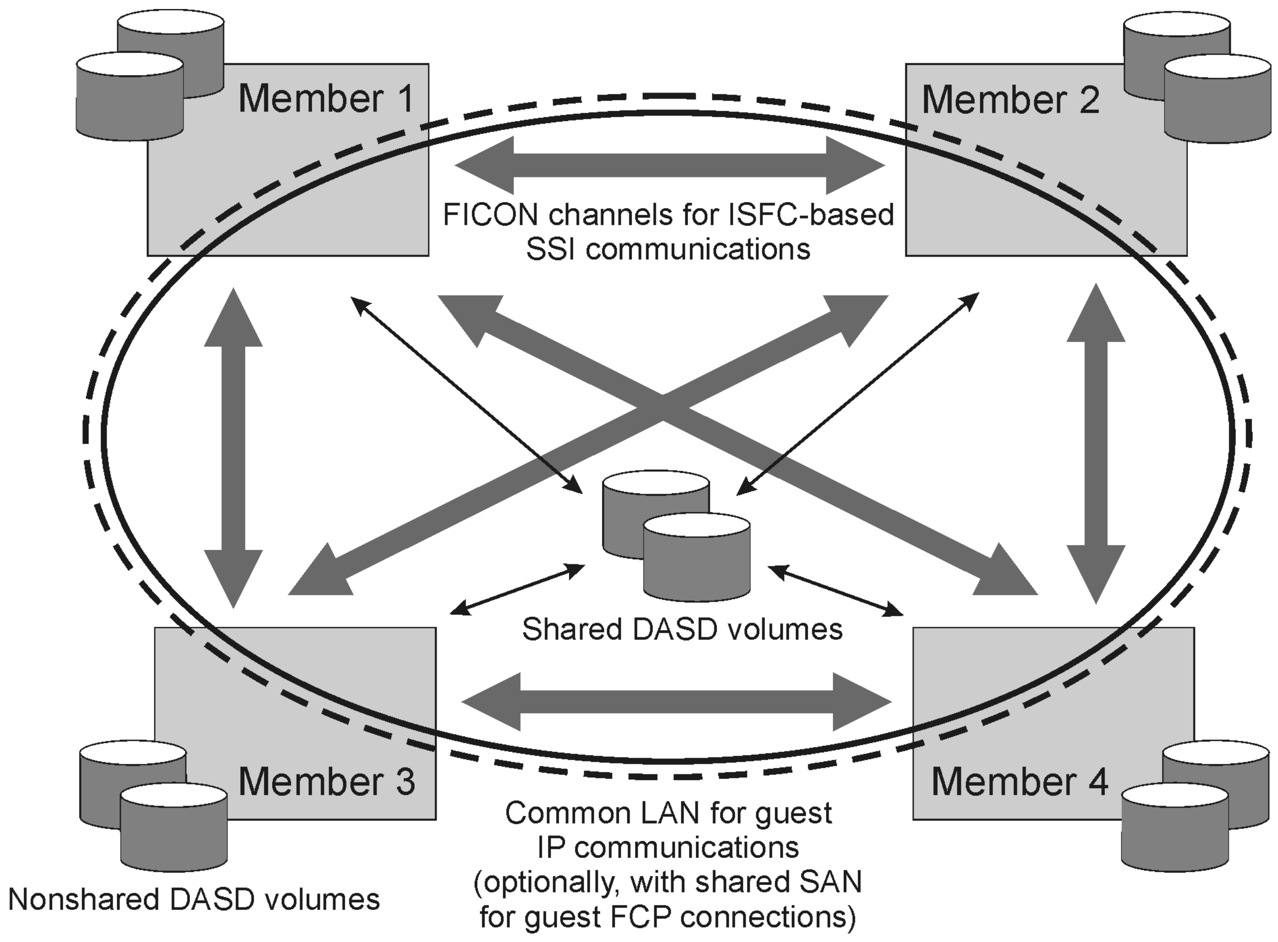

Member 1
Member 2
FICON channels for ISFC-based SSI communications
Shared DASD volumes
Member 3
Member 4
Common LAN for guest IP communications (optionally, with shared SAN for guest FCP connections)
Nonshared DASD volumes

Figure 35. A Four-Member z/VM SSI Cluster

Major Attributes of a z/VM SSI Cluster

The major attributes of an SSI cluster are:

- "Multisystem Installation" on page 716
- "Single Maintenance Stream" on page 717
- "Cross-System Highest Release Level Programs" on page 719
- "Common System Configuration File" on page 720
- "Persistent Data Record" on page 721
- "Ownership Checking of CP-Owned Volumes" on page 721
- "Virtual Machine Definition Management" on page 722
- "Cross-System Spool" on page 724
- "Cross-System SCIF, Observer, and CP Commands" on page 724
- "Cross-System Minidisk Management" on page 725
- "Real Device Management" on page 727
- "Virtual Networking Management" on page 727
- "Live Guest Relocation" on page 727

Multisystem Installation

When you install z/VM, you can select an installation procedure that creates an SSI cluster. You do not need to do multiple system installations. On the installation panels you supply information about the SSI cluster, such as the name of the cluster, the number of members and their system names, the name of the logical partition (LPAR) in which each member will be IPLed, and information about the DASD volumes and channel-to-channel connections. The installation program will install the z/VM images as an SSI cluster and provide a common system configuration file and a common source directory.

The process provides additional flexibility:

- You can create an SSI cluster with more members than you need right now. The installation program will install and customize the z/VM images and create the cluster structure for the "extra" members, and you can activate them when you need them.
- You can create a single-member SSI cluster as the first step in moving to a multimember SSI cluster. At a later time you can clone the single member to add other members to the cluster.
- You can select a non-SSI installation procedure and convert that z/VM system to the initial member of an SSI cluster at a later time. The SSI and non-SSI installations construct the system configuration file, the source directory, and the layout of DASD resources in the same way, to facilitate this conversion.

Single Maintenance Stream

Figure 36 on page 718 shows the basic organization of DASD volumes and minidisks in an SSI cluster (consisting of two members in this example):

- One set of cluster-wide volumes (shared)

  The common volume (default label VMCOM1) contains shared data files for the SSI cluster, such as the SSI persistent data record (PDR). This volume also contains the cluster-wide minidisks, which are owned by the PMAINT user ID. These minidisks include (but are not limited to):

Minidisk	**Purpose**
PMAINT CF0	Common system configuration file
PMAINT 2CC	Common source directory
PMAINT 41D	VMSES/E production inventory disk
PMAINT 551	SSI cluster common disk (contains common utilities, such as CPFMTXA, DIRECTXA, DIRMAP, and DISKMAP)

- One set of release volumes for each z/VM release in the cluster (shared)

  Service for release *vrm* (where *vrm* is a 3-digit string that identifies the z/VM version, release, and modification level) is loaded to minidisks on one or two release-specific volumes (default labels *vrm*RL1 and *vrm*RL2). The minidisks on these volumes are owned by a release-specific user ID, MAINT*vrm*. These minidisks include (but are not limited to):

Minidisk	**Purpose**
MAINT*vrm* 490	Test CMS system disk
MAINT*vrm* 493	Test system tools disk
MAINT*vrm* 51D	VMSES/E software inventory disk
MAINT*vrm* CF2	Test parm disk

- One set of system volumes for each member (nonshared and shared)

The system residence volume is member-specific and therefore nonshared. The default volume label is M0*m*RES, where *m* is the number of the member in the member list on the SSI configuration statement (for example, M01RES for member 1). This volume contains member-specific data such as the warm start and checkpoint areas, the object directory, and the standard system minidisks owned by the MAINT user ID, such as MAINT 190 and MAINT 193.

The CP-owned volumes for paging and temporary disks are also member-specific and nonshared. The default volume labels are M0*m*P*nn* and M0*m*T*nn*, respectively (for example, M01P01 and M01T01 for member 1).

The spool volumes owned by each member are shared with the other members. The default volume labels are M0*m*S*nn* (for example, M01S01 for member 1 and M02S01 for member 2).

PARM or PERM extents on other CP-owned volumes owned by each member are sharable.

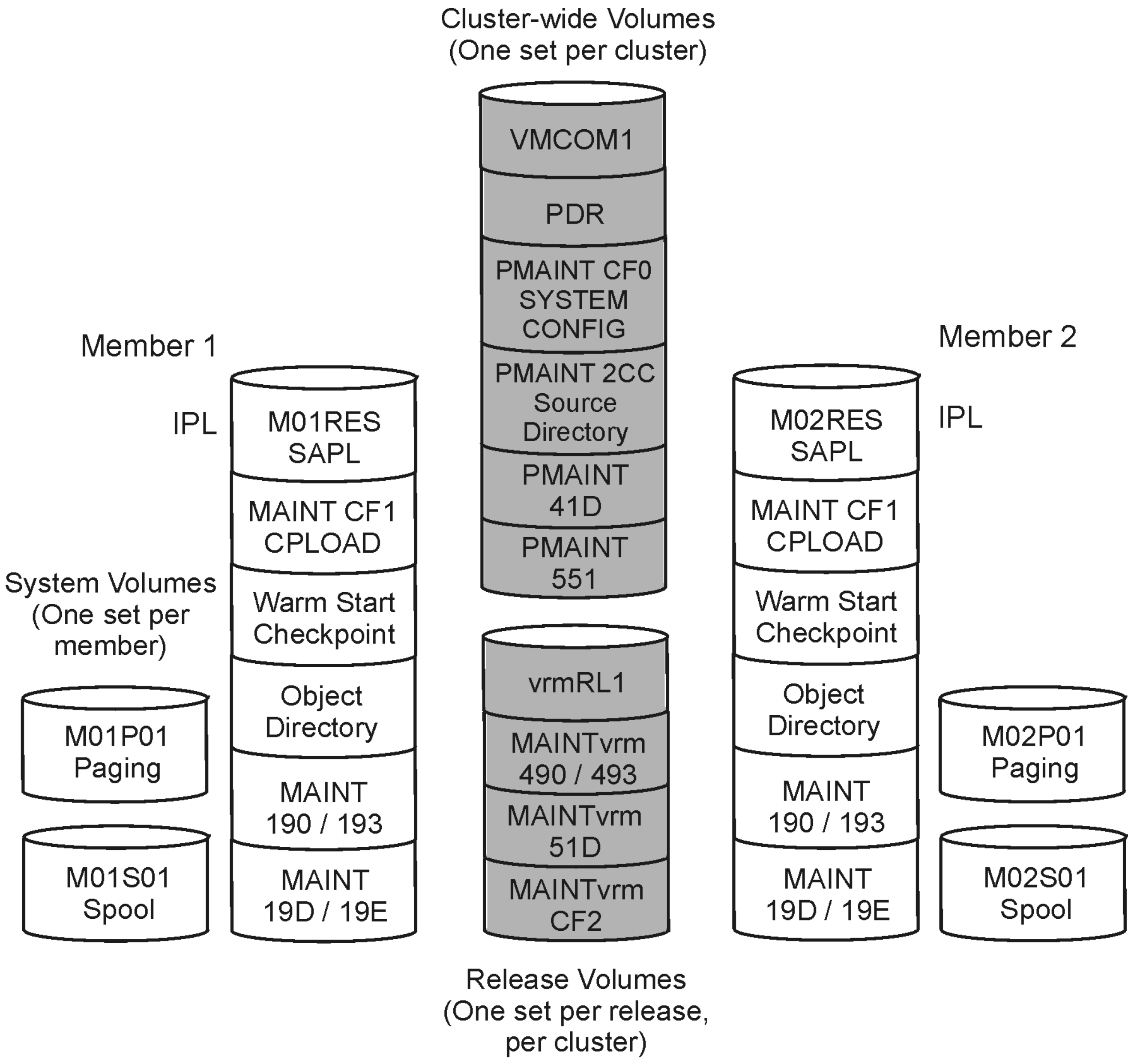


Figure 36. Organization of DASD Volumes and Minidisks in an SSI Cluster

For each z/VM release in an SSI cluster, there is:

- One maintenance user ID (MAINT*vrm*)
- One set of shared service inventory minidisks:
  - Local/Sample, APPLY, DELTA, and test build disks

– One minidisk for local modifications to CP

Each member of an SSI cluster has a separate set of production minidisks. This allows service to be put into production independently on each member.

As shown in Figure 37 on page 719, the process for applying service for release *vrm* in an SSI cluster consists of the following steps:

1. Log on to MAINT*vrm* on any member of the cluster (running any release).
2. Issue the SERVICE command to install the *vrm* Recommended Service Upgrade (RSU) or corrective service (COR) to the minidisks on volume *vrm*RL1.
3. Log on to MAINT*vrm* on a member running release *vrm* that you want to update.
4. Issue the PUT2PROD command to put the serviced products into production on the system residence volume for that member.
5. When you are ready to put the service into production on another member, repeat steps 3 and 4.

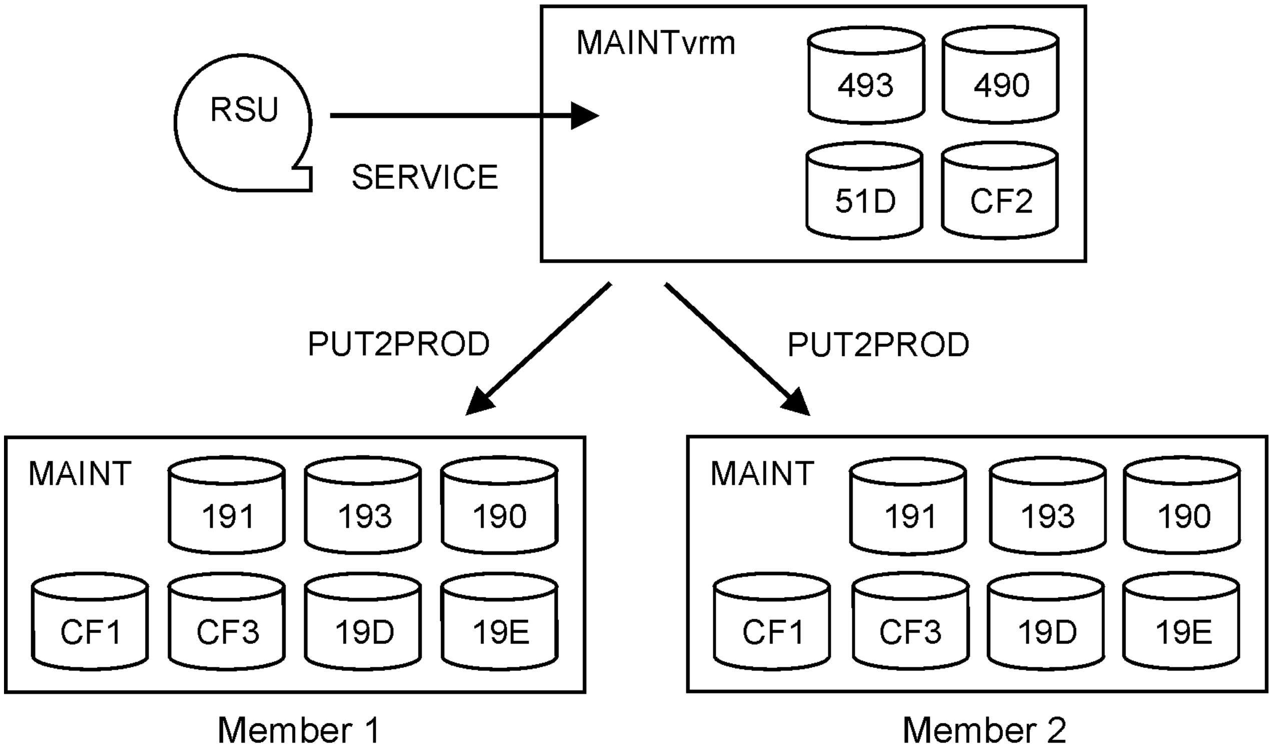


Figure 37. Applying Service to the Members of an SSI Cluster

Cross-System Highest Release Level Programs

In an SSI cluster, certain resources – for example, the user directory, the system configuration file, the permanent data record (PDR) and shared DASD devices – are shared and managed by all members in the cluster. If the members in an SSI cluster are running different release levels of z/VM, certain programs which manage shared resources are required to be at the highest release level that is running in the cluster. These programs must be on all members in the cluster regardless of the release level running on each member. These *highest release level programs* reside on the SSI system common disk (PMAINT 551, by default).

When a z/VM release which supersedes all other releases running on the members in an SSI cluster is installed on a member of the cluster, z/VM installation processing places these programs from the superseding release on the SSI system common disk, replacing all programs from the superseded release. When these programs are serviced on the highest release level running in the cluster, the programs are serviced as normal, being built and copied to the SSI system common disk. However, for all members running a lower-level, superseded z/VM release, these programs cannot be built during the service process because this would back-level the parts on the SSI system common disk. Instead, when these programs are serviced on any superseded release running in the cluster, the service process bypasses the building of these parts so that the parts from the highest release level installed in the cluster remain on the SSI system common disk.

Common System Configuration File

The system definition statements for all of the members of an SSI cluster are contained in a common system configuration file. The file resides on the common parm minidisk (PMAINT CF0) on the common DASD volume (VMCOM1).

Because some statements in the common system configuration file apply to all members of the SSI cluster, while others are member-specific, record qualifiers are used to identify the member-specific statements. Qualified BEGIN and END statements can be used to identify blocks of member-specific statements.

The file begins with a SYSTEM_IDENTIFIER statement for each member of the SSI cluster, as shown in Figure 38 on page 720. This statement defines the unique system name (system ID) for the z/VM system to be run in a specified logical partition (LPAR). The system name is used as the record qualifier parameter on other configuration statements that apply only to that member of the cluster. When a member system is initialized in an LPAR, CP locates the SYSTEM_IDENTIFIER statement with that LPAR name, then uses the defined system name to select the configuration statements to be processed for that member — all statements qualified with that system name and all statements with no qualifier.

```
    System_Identifier LPAR LP01 VMSYS01
    System_Identifier LPAR LP02 VMSYS02
    System_Identifier LPAR LP03 VMSYS03
    System_Identifier LPAR LP04 VMSYS04
```

Figure 38. Example of SYSTEM_IDENTIFIER Configuration Statements for an SSI Cluster

After the system names are defined, the SSI statement indicates this system configuration file defines an SSI cluster. As shown in Figure 39 on page 720, the SSI statement specifies the name of the cluster, the label of the DASD volume that contains the SSI persistent data record (PDR), and the system names of the members.

```
    SSI CLUSTERA PDR_Volume VMCOM1 ,
         Slot 1 VMSYS01,
         Slot 2 VMSYS02,
         Slot 3 VMSYS03,
         Slot 4 VMSYS04
```

Figure 39. Example of the SSI Configuration Statement

ACTIVATE ISLINK statements identify channel-to-channel adapters (CTCAs) for ISFC links between the members of the cluster. Each member must have at least one direct link to each of the other members. For example, the statements for member VMSYS01 shown in Figure 40 on page 720 identify devices for two links to member VMSYS02, two links to member VMSYS03, and two links to member VMSYS04. Each of the other members of the cluster would require a similar set of statements.

```
    VMSYS01: BEGIN
             Activate ISLINK rdev1 rdev2 Node VMSYS02
             Activate ISLINK rdev3 rdev4 Node VMSYS03
             Activate ISLINK rdev5 rdev6 Node VMSYS04
    VMSYS01: END
```

Figure 40. Example of ACTIVATE ISLINK Statements for an SSI Cluster

For an example of the file, see "System Configuration File for a z/VM SSI Cluster" on page 742.

Persistent Data Record

Information about the state of each member of an SSI cluster is maintained in the SSI persistent data record (PDR). The PDR provides a cross-system serialization point on disk, which assists in member state tracking. The PDR is used to provide a *heartbeat* mechanism, which ensures that a stalled or stopped member can be detected.

The FORMSSI utility is used to create or display the PDR. The PDR is created on cylinder 0 of the shared 3390 volume identified on the SSI configuration statement. The PDR can be located on the volume that also contains the common parm disk and the common source directory.

Each active member periodically:

- Updates the PDR with a heartbeat timestamp and sends the same heartbeat to the other members
- Monitors the heartbeat timestamps received from the other members and compares them with the timestamps in the PDR

Relocating the PDR

The SET SSI PDRVOLUME command can be used to relocate the PDR to a new DASD volume without a planned outage. The PDR on the old device is marked obsolete. After a successful PDR relocation, the SSI statement in the system configuration file must be updated to complete the switching of the PDR. Also, the new PDR volume must be added to the CP-owned or user volume list in the system configuration file. Although any attempt to use the old PDR volume will automatically switch to the new one, it is strongly recommended that you update the SSI statement immediately after a successful switch.

After the SSI statement is updated with the replacement PDR volume, the old PDR volume can be detached from the system if no other system area is in use on that volume, and the replacement PDR volume will no longer be detachable. If you are performing a DASD subsystem replacement, you might choose to shut down some members of the SSI cluster before relocating the PDR. (You must do so at least for the systems that do not support the PDRVOLUME operand of the SET SSI command.) If the SSI statement is not updated with the new volume after a successful switch, systems without this support will not understand the obsolete PDR and will load a 9052 wait state during system IPL.

In an SSI layout, the PDR is on the common volume (default label VMCOM1), which also contains the system configuration file and other shared data files. In many cases, the task of relocating the PDR is just one part of the job of moving the contents of the entire volume. For more information, see "Considerations for Migrating the SSI Common Volume" on page 740.

Ownership Checking of CP-Owned Volumes

To ensure that CP on one member of an SSI cluster will not allocate warm start, checkpoint, spool, paging, temporary disk, or directory data on a volume owned by another member, each CP-owned volume in an SSI cluster must be marked with ownership information. The ownership information consists of the name of the cluster and the system name of the member that owns the volume. Ownership information can be recorded on a DASD volume by using the OWNER operand of the CPFMTXA utility.

When the CP-owned list is processed for a member of an SSI cluster, the volume extents to be brought online are determined by the ownership information recorded on each volume. (The OWN and SHARED operands of the CP_OWNED system configuration statement and the DEFINE CPOWNED command are ignored.) For more information about how the CP-owned list is processed, and how the volume extents are determined by the ownership information, see Chapter 24, "How the CP-Owned List Is Processed," on page 663.

Ownership information for CP-owned volumes can be viewed by issuing the QUERY CPOWNED command. In the following examples, an SSI cluster named CLUSTERA has two members, VMSYS01 and VMSYS02.

Issuing the QUERY CPOWNED command from a user ID logged on to member VMSYS01 results in the following response:

```
SLOT  VOL-ID  RDEV  TYPE   STATUS                  SSIOWNER SYSOWNER
   1  M01RES  C4A0  OWN    ONLINE AND ATTACHED     CLUSTERA VMSYS01
   2  ------  ----  -----  RESERVED                -------- --------
   3  ------  ----  -----  RESERVED                -------- --------
   4  ------  ----  -----  RESERVED                -------- --------
   5  VMCOM1  C4FF  OWN    ONLINE AND ATTACHED     CLUSTERA --------
   6  ------  ----  -----  RESERVED                -------- --------
   7  ------  ----  -----  RESERVED                -------- --------
   8  ------  ----  -----  RESERVED                -------- --------
   9  ------  ----  -----  RESERVED                -------- --------
  10  M01S01  C4A8  OWN    ONLINE AND ATTACHED     CLUSTERA VMSYS01
  11  M02S01  C4B8  SHARE  ONLINE AND ATTACHED     CLUSTERA VMSYS02
  12  M01S02  C4A9  OWN    ONLINE AND ATTACHED     CLUSTERA VMSYS01
  13  M02S02  C4B9  SHARE  ONLINE AND ATTACHED     CLUSTERA VMSYS02
  14  M01S03  C4AA  DUMP   ONLINE AND ATTACHED     CLUSTERA VMSYS01
  15  M02S03  C4BA  DUMP   ONLINE AND ATTACHED     CLUSTERA VMSYS02
  16  ------  ----  -----  RESERVED                -------- --------
⋮

 253  ------  ----  -----  RESERVED                -------- --------
 254  M01P02  C4C1  OWN    ONLINE AND ATTACHED     CLUSTERA VMSYS01
 255  M01P01  C4C0  OWN    ONLINE AND ATTACHED     CLUSTERA VMSYS01
READY;
```

Issuing the QUERY CPOWNED command from a user ID logged on to member VMSYS02 results in the following response:

```
SLOT  VOL-ID  RDEV  TYPE   STATUS                  SSIOWNER SYSOWNER
   1  M02RES  C4B0  OWN    ONLINE AND ATTACHED     CLUSTERA VMSYS02
   2  ------  ----  -----  RESERVED                -------- --------
   3  ------  ----  -----  RESERVED                -------- --------
   4  ------  ----  -----  RESERVED                -------- --------
   5  VMCOM1  C4FF  OWN    ONLINE AND ATTACHED     CLUSTERA --------
   6  ------  ----  -----  RESERVED                -------- --------
   7  ------  ----  -----  RESERVED                -------- --------
   8  ------  ----  -----  RESERVED                -------- --------
   9  ------  ----  -----  RESERVED                -------- --------
  10  M01S01  C4A8  SHARE  ONLINE AND ATTACHED     CLUSTERA VMSYS01
  11  M02S01  C4B8  OWN    ONLINE AND ATTACHED     CLUSTERA VMSYS02
  12  M01S02  C4A9  SHARE  ONLINE AND ATTACHED     CLUSTERA VMSYS01
  13  M02S02  C4B9  OWN    ONLINE AND ATTACHED     CLUSTERA VMSYS02
  14  M01S03  C4AA  DUMP   ONLINE AND ATTACHED     CLUSTERA VMSYS01
  15  M02S03  C4BA  DUMP   ONLINE AND ATTACHED     CLUSTERA VMSYS02
  16  ------  ----  -----  RESERVED                -------- --------
⋮

 253  ------  ----  -----  RESERVED                -------- --------
 254  M02P02  C4D1  OWN    ONLINE AND ATTACHED     CLUSTERA VMSYS02
 255  M02P01  C4D0  OWN    ONLINE AND ATTACHED     CLUSTERA VMSYS02
READY;
```

Virtual Machine Definition Management

All of the virtual machines in an SSI cluster are defined in a single source directory, from which the object directory for each member is created. There are two types of virtual machine definitions:

Single-configuration virtual machine definition

A user ID defined by a single-configuration virtual machine definition (the traditional type of definition) can be logged on to any member of the SSI cluster, one member at a time. The definition begins with a USER statement and consists of the user entry and any included profile entry.

Use this type of virtual machine definition for general workload (users and applications), platforms for running guest operating systems, and service virtual machines (SVMs) and servers that require only one logon in the SSI cluster (that is, those which provide cluster-wide services).

Multiconfiguration virtual machine definition

A user ID defined by a multiconfiguration virtual machine definition can be logged on concurrently to multiple members of the SSI cluster. The *instances* of the user ID on the members have common attributes but can also be configured to access different resources.

The definition begins with an IDENTITY statement and consists of the identity entry, any included profile entry, and the associated subconfiguration entries. Authorization information (such as privilege classes) and authentication information (such as a password) must be the same for all instances of the user ID, and therefore must be included in the identity entry. Resources available to all instances of the user ID are defined in the identity entry. The identity entry can also include a BUILD statement for each cluster member that points to a corresponding subconfiguration entry. Each subconfiguration entry contains member-specific information, such as MDISK statements for minidisks that are defined on a different volume for each member. When the user ID logs on to a member of the cluster, the statements from the identity entry, any included profile entry, and the appropriate subconfiguration entry are merged to create the instance of the user ID on that member.

Use this type of virtual machine definition for system support user IDs. The ability to be logged on to multiple members of an SSI cluster at the same time allows the user (such as the IBM-supplied MAINT user ID) to perform tasks on various members without having to log off one member before logging on to another. This type of definition can also be used for SVMs and servers (such as the IBM-supplied TCPIP server) that require an instance to be running on each member of the SSI cluster. Each instance supports only that member of the cluster.

Figure 41 on page 723 shows a sample multiconfiguration virtual machine definition for the TCPIP user ID.

```
IDENTITY TCPIP password 128M 256M ABG
 INCLUDE TCPCMSU
 BUILD ON VMSYS01 USING SUBCONFIG TCPIP-1
 BUILD ON VMSYS02 USING SUBCONFIG TCPIP-2
 BUILD ON VMSYS03 USING SUBCONFIG TCPIP-3
 BUILD ON VMSYS04 USING SUBCONFIG TCPIP-4
 OPTION QUICKDSP SVMSTAT MAXCONN 1024 DIAG98 APPLMON
 SHARE RELATIVE 3000
 IUCV ALLOW
 IUCV ANY PRIORITY
 IUCV *CCS PRIORITY MSGLIMIT 255
 IUCV *VSWITCH MSGLIMIT 65535

SUBCONFIG TCPIP-1
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M01RES MR RTCPIP WTCPIP MTCPIP

SUBCONFIG TCPIP-2
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M02RES MR RTCPIP WTCPIP MTCPIP

SUBCONFIG TCPIP-3
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M03RES MR RTCPIP WTCPIP MTCPIP

SUBCONFIG TCPIP-4
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M04RES MR RTCPIP WTCPIP MTCPIP
```

Figure 41. Example of a Multiconfiguration Virtual Machine Definition

When z/VM is installed, an *SSI-enabled* or *SSI-ready* source directory is created, depending on the installation procedure:

SSI-enabled

An SSI-enabled source directory is created when an SSI installation is performed. The source directory for an SSI cluster must be SSI-enabled, and an SSI-enabled source directory can be used only for an SSI cluster.

This type of source directory contains a single DIRECTORY statement that specifies the SSI operand and identifies the DASD volumes where the object directory for each member will be created. The source directory can contain both single-configuration virtual machine definitions and multiconfiguration virtual machine definitions.

SSI-ready

An SSI-ready source directory is created when a non-SSI installation is performed. This type of source directory can be used only for a single non-SSI system or a single-member SSI cluster. An SSI-ready source directory can easily be converted to an SSI-enabled source directory.

In this type of source directory, the SSI operand is not specified on the DIRECTORY statement. The source directory can contain both single-configuration virtual machine definitions and multiconfiguration virtual machine definitions. However, a multiconfiguration virtual machine definition can include at most one BUILD statement and its associated subconfiguration entry. On the BUILD statement the system name must be specified as an asterisk (*) to default to the IPLed system. Only one instance of the user ID can be created from this multiconfiguration virtual machine definition. The definition is functionally equivalent to a single-configuration virtual machine definition but is ready for conversion as part of an SSI-enabled source directory.

Cross-System Spool

Cross-system spool enables the CP spooling functions on each member of an SSI cluster to create, manage, and share spool files cooperatively with the other members and enables spool files to be shared among the members. Each member creates spool files only on its own spool volumes, but has access to the spool volumes owned by the other members.

Users defined by single-configuration virtual machine definitions (beginning with a USER statement) have a single logical view of all their spool files. Users can log on to any member of the cluster and access all of their spool files, regardless of the member on which the files were created. (For a spool file to be accessible, the member on which it was created must be joined to the cluster.) Users can manipulate their spool files in the same ways as on a non-SSI system.

Users defined by multiconfiguration virtual machines definitions (beginning with an IDENTITY statement) do not participate in cross-system spool. Each virtual machine instance can view only the spool files created on the member where the instance is logged on. A spool file created on one member of the cluster is not visible to an instance of that multiconfiguration virtual machine on another member. Spool file transfer services, such as RSCS, can be used to move spool files to the member where the intended instance of the multiconfiguration virtual machine resides.

For more information, see Chapter 30, “Cross-System Spool in a z/VM SSI Cluster,” on page 747.

Cross-System SCIF, Observer, and CP Commands

The SSI cluster environment supports cross-system use of the single console image facility (SCIF), the observer function, and many CP commands.

Cross-System SCIF and Observer

Cross-system SCIF and observer functions allow a secondary user or observer and the primary or observed user to be logged on to different members of the SSI cluster.

If either user in the relationship is a multiconfiguration virtual machine, the secondary user or observer will function in that capacity only when it is local (logged on to the same member as the primary or observed user). If the primary or observed user is a multiconfiguration virtual machine, the secondary user or observer must be logged on to the same member as the instance of the primary or observed user for which the secondary user or observer setting was established. If the secondary user is remote:

- The secondary user can issue SEND commands to the primary user. The AT *sysname* operands are required if the primary user is a multiconfiguration virtual machine instance.
- The secondary user will not receive responses to SEND commands or any other output from the primary user.

Cross-System CP Commands

Some CP commands are enabled for cluster use through the AT *sysname* operands, which allow you to specify the target member. Examples are:

- INDICATE LOAD
- MESSAGE (MSG)
- MSGNOH
- QUERY NAMES
- QUERY USERS
- SMSG
- WARNING

In addition, most privileged CP commands (that is, not class Any or G) can be issued remotely on another active member of the SSI cluster by specifying the command and the remote member on the AT command on the local member:

```
AT sysname CMD command
```

Note: When executed with the AT command, a command that provides a system or user ID specification might have additional syntax requirements or restrictions. For more information, see the AT command in *z/VM: CP Commands and Utilities Reference*.

When the HOLD LOGON or FREE LOGON command is issued on a member of an SSI cluster for a user ID defined as a single-configuration virtual machine, the command is propagated to all the other members of the SSI cluster. If the user ID is defined as a multiconfiguration virtual machine, the command is executed only on the member where it is issued.

Note: If another member of the SSI cluster is specified by the AT *node* operands on a CMS productivity aid (such as the TELL or SENDFILE command), CMS views the specified system as a remote node and will attempt to route the associated message or file through RSCS. In most cases it is best to avoid using the AT *node* operands on these commands. For more information, see "How the CMS Productivity Aids Process Spool Files in an SSI Cluster" on page 750.

Cross-System Minidisk Management

Users on different members of an SSI cluster can use the LINK command and the LINK directory statement to share minidisks. The members of the cluster exchange information about minidisk links to determine who has access to a minidisk. When a minidisk link is requested, CP checks for link conflicts on all members. The QUERY LINKS command displays information about links by users on all members of the cluster.

Notes:

1. XLINK system configuration statements and commands are not required or accepted in an SSI cluster.
2. Do not share minidisks between an SSI cluster and a CSE complex. CP does not analyze link conflicts between an SSI cluster and CSE.
3. Sharing minidisks between two SSI clusters or between an SSI cluster and a non-SSI system is not recommended. If such links are defined, the DASD volume should be defined as SHARED YES. CP does not analyze link conflicts beyond the scope of a single SSI cluster.

The SHARED operand on the RDEVICE (DASD) system configuration statement or the SET RDEVICE (DASD) command specifies whether the DASD volume is shared outside the SSI cluster:

SHARED NO
The volume is not shared outside the SSI cluster. This is the default. Minidisks defined on this volume are eligible for minidisk cache, which is managed autonomically as write links are established and released.

SHARED YES
The volume is shared with a system outside the SSI cluster. Minidisks defined on this volume are not eligible for minidisk cache. Link conflicts must be managed manually by the system administrator.

The scope of a minidisk in an SSI cluster can be *global* or *local*, which refers to the accessibility of the minidisk from the members of the cluster:

Global
The minidisk is accessible from all members of the SSI cluster.

Local
The minidisk is accessible from only one member of the SSI cluster. A local minidisk is exempt from cross-system minidisk management. A guest with a link to a local minidisk cannot be relocated.

In addition to its scope, a minidisk is either *shareable* or *private* (nonshareable), which refers to the accessibility of the minidisk from other virtual machines. The scope and shareability of a minidisk in an SSI cluster are determined by the type of minidisk and how it is defined, as shown in Table 48 on page 726.

Table 48. Minidisk Scope and Shareability in an SSI Cluster

Minidisk Type and Definition	**Scope**	**Shareability**
Permanent minidisk defined by an MDISK directory statement in a user or identity entry	Global	Shareable
Permanent minidisk defined by an MDISK directory statement in a subconfiguration entry	Local	Shareable
Minidisk defined by a DEFINE MDISK command	Local	Private
Virtual disk in storage defined by an MDISK...V-DISK directory statement in a user or subconfiguration entry[1]	Local	Shareable
Virtual disk in storage defined by a DEFINE VFB-512 command	Local	Private
Temporary minidisk defined by an MDISK...T-DISK directory statement[2]	Local	Private
Temporary minidisk defined by a DEFINE (Temporary Disk) command	Local	Private

Notes:

1. An MDISK statement for a virtual disk in storage is not permitted in an identity entry. If a virtual disk in storage is defined in a user entry, the minidisk can be created on any member of the cluster (when the owner logs on or when the first user links to it), but subsequently can be shared only by other users on that member until the last user detaches it or logs off.
2. If a temporary minidisk is defined in an identity entry, a separate temporary minidisk is created for each logon instance of the virtual machine.

If a write link is established to a minidisk eligible for minidisk cache, the other members of the SSI cluster will stop using minidisk cache for that minidisk and discard the cache, in order to avoid providing obsolete data to any guest with read-only links to the minidisk. If only one member has write links to the minidisk, minidisk cache can remain active on that member. When the write link is released, the other members will reinstate the minidisk cache.

Virtual reserve/release for full-pack minidisks is implemented as real reserve/release and is supported on multiple members of the SSI cluster. A non-full-pack minidisk authorized for virtual reserve/release can be linked by users on only one member of the SSI cluster at any given time.

For information about how minidisk link conflicts are resolved, see the LINK command in *z/VM: CP Commands and Utilities Reference*.

Real Device Management

z/VM provides a mechanism for uniquely identifying real devices within an SSI cluster, to ensure that the members of the cluster are using the same physical devices. For devices that identify themselves uniquely, such as FICON-attached DASD and tape devices, CP generates an equivalency identifier (EQID) that is unique on a member of the cluster (because no other device has the same EQID) but the same from member to member (because each member is accessing the same device).

Adapters, such as channel-to-channel (CTC), Fiber-Channel-Protocol (FCP), HiperSockets, and Open Systems Adapter (OSA) devices, require additional information to establish equivalence, such as the fabric to which the adapter is connected and the access rights of the individual device. For these device classes and for devices that do not provide a unique identifier, the system administrator can use the RDEVICE system configuration statement or the SET RDEVICE command to assign EQIDs that tell CP which devices are equivalent. This manual setting indicates that the devices have equivalent connectivity capability. An administrator-assigned EQID must be unique or be shared only by other devices of the same type and with the same access rights.

The EQID is used instead of the real device number to determine device eligibility for guest relocation. Within a virtual configuration, a device that has been assigned an EQID is considered to be a candidate for use in a relocation. These real devices are associated with the relocating guest's virtual devices. The relocation process searches the real devices on the destination member for the specific device with the same CP-generated EQID or an available device with the same administrator-assigned EQID. If a match is found, the real device can be associated with the guest's virtual I/O configuration on the destination member. If a device does not have an EQID or if no match is found, it cannot be added to the guest's virtual I/O configuration on the destination member, and therefore the relocation should not be attempted. If it is attempted, it might lead to a failure of the guest operating system.

Virtual Networking Management

The assignment of Ethernet Media Access Control (MAC) addresses to virtual network interface cards (NICs) is coordinated within an SSI cluster. The members of the cluster ensure that a MAC address is not already in use by another guest within the cluster. z/VM generates locally administered MAC addresses for network devices using MAC prefixes. These MAC prefixes are set to CP-defined defaults or can be configured by the system administrator. For more information about MAC addresses, see Planning for Guest LANs and Virtual Switches in *z/VM: Connectivity*.

Each member of an SSI cluster should have identical network connectivity. To accomplish this, you should:

- Define a virtual switch with the same name and attributes on each member.
- Define the virtual switch to have one or more physical OSA ports connected to the same physical LAN segment.

Live Guest Relocation

A z/VM SSI cluster provides a virtual server mobility function called live guest relocation. A running virtual server (guest virtual machine) can be relocated from one member to another. Relocating guests can be useful for load balancing and for moving workload off a physical server or member system that requires maintenance.

Use the VMRELOCATE command to initiate and manage guest relocations. Both the guest to be relocated and the destination environment must meet eligibility requirements, such as:

- The destination member and physical server must provide an architecturally and functionally comparable environment.
- The relocation must meet any policy requirements that have been established. Relocation domains can be defined to identify sets of eligible members among which guests can be relocated freely.
- The destination member must have sufficient capacity to accommodate the guest.
- The devices and resources needed by the guest must be shared, so they are also available on the destination member.

When a relocation is initiated, the guest continues to run on the source member until the destination environment is completely prepared. At that point the guest is briefly quiesced on the source member and then resumed on the destination member.

For additional information about relocation eligibility and options, see Chapter 31, "Preparing for Guest Relocations in a z/VM SSI Cluster," on page 753.

Planning for a z/VM SSI Cluster

Before you create a z/VM SSI cluster, ensure that your configuration conforms to the requirements and restrictions.

Attention: Also ensure that you have allocated enough system resources to account for the necessary synchronization and communication among the members of the SSI cluster. Independent systems that are not formally clustered do not require this synchronization overhead. For example, two independent systems that run satisfactorily at peak utilization close to 100%, when joined in a cluster might fail.

SSI Cluster Requirements

The following items are required to define and run a z/VM SSI cluster:

- Servers must be IBM z13 or later.
- Shared and nonshared DASD:
  - Shared 3390 volume for the PDR.
  - Shared spool volumes.
  - If RACF is used, the RACF database must be on shared 3390 DASD.
  - All DASD volumes should be cabled to all members of the cluster. This allows volumes containing nonshared disks (for example, minidisks owned by multiconfiguration virtual machines) to be accessed by members other than the owning member when IPLed with the REPAIR parameter to fix configuration problems.

    For normal operations, only the owning member should include a volume containing nonshared minidisks on a CP_OWNED or USER_VOLUME_INCLUDE statement. All other members should configure the volume as OFFLINE_AT_IPL on a DEVICES statement. In the event repair is required, the volume can be manually varied online and attached to the system.
  - See *z/VM: Installation Guide* for installation requirements.
  - See *z/VM: General Information* for information about the DASD supported by z/VM.
- Connectivity:
  - A direct ISFC connection from each member of the SSI cluster to each of the other members. See "Planning the ISFC Network in an SSI Cluster" on page 730.
  - FICON channels to the shared DASD.
  - OSA access to the same LAN segments.
  - (If needed) FCP access to the same storage area networks (SANs) with the same storage access rights.
  - Shared access to any devices required by relocatable guests.

- Single system configuration file for all members of the SSI cluster (created when the SSI cluster is installed).
- Common source directory for all members of the SSI cluster (created when the SSI cluster is installed).
- Capacity planning for the SSI cluster, as well as for the individual members, to ensure that sufficient resources are available for guests that relocate from one member to another or that log on to different members of the cluster.
- If you are using an external security manager (ESM), such as RACF, the ESM database must be shared by the members of the SSI cluster. For information about sharing RACF databases in an SSI cluster, see *z/VM: RACF Security Server System Programmer's Guide*.

SSI Cluster Restrictions

An SSI cluster has the following restrictions:

- The physical systems must be close enough to allow FICON CTCs, shared DASD, and common network and disk fabric connections.
- Installation on SCSI devices is not supported; however, post-installation use of SCSI devices by guests is permitted.
- The live guest relocation function is currently supported only for Linux guests.

Suggested Practices for Setting Up an SSI Cluster

You also might find it beneficial to follow these practices:

- To simplify the references in the system configuration file and the source directory, use common real device numbers across the LPARs. For example:
  - Use the same real device number to reference a DASD.
  - Use the same real device numbers for FICON CTCs between pairs of member systems.
  - Use the same range of real device numbers for the OSA or HiperSockets subchannels connected to the same network.
  - Use the same range of real device numbers for the FCP subchannels connected to the same fabric.
- For availability reasons, install no more than two members of an SSI cluster on the same server.
- Maintain a parallel volume layout (minidisk definitions) on the corresponding nonshared DASD volumes for each member of the cluster.
- For each member of the cluster, allocate object directory extents only on the system residence volume.
- To simplify upgrading from release to release, do not place user data on the installation volumes.
- To simplify expansion of an SSI cluster (by cloning an existing member), keep member-specific data and SSI cluster data on separate volumes.
- If the members of the SSI cluster will not have the same hardware features and architectural enhancements, consider the effect on a guest virtual machine should you want to relocate it. This consideration could affect the relocation domains that you define and which guests you assign to those domains. You might want to record the hardware features or architectural enhancements that are important for each guest that might be relocated. For example, a guest might need a particular type of cryptographic feature or High Performance FICON for IBM Z (zHPF). For additional details see "Using Relocation Domains" on page 754.

For additional information on topics related to setting up and running an SSI cluster, see z/VM Single System Image Overview (https://www.ibm.com/vm/ssi).

Using CPU Pools in an SSI Cluster

If you plan to use CPU pools in the SSI cluster, you should define compatible CPU pools on all members of the SSI cluster where users assigned to the CPU pool could log on or be relocated. A user assigned to a CPU pool can be relocated only to another member of the SSI cluster that contains a CPU pool with the same name and the same type of virtual CPU being limited. All users in the CPU pool need to use the same underlying CPU resource for CPU pooling to work, which is why all users assigned to a CPU pool must have CPU affinity on and can relocate only to a CPU pool with the same virtual CPU type as their current CPU pool. Based on the availability of IFL CPUs on the destination system, z/VM will toggle the relocated user between CPU affinity on and CPU affinity suppressed, so that the CPU affinity settings for the user and the CPU pool are identical.

Because the scope of a CPU pool is a single z/VM system, the relocation will essentially remove the user from the CPU pool on the source system and assign the user to the CPU pool on the destination system. CPU limit settings are enforced separately on each system, so it might be necessary or desirable to adjust the CPU pool capabilities on the destination system before relocation or on the source system after relocation.

Planning the ISFC Network in an SSI Cluster

Each member in the SSI cluster must have a direct ISFC connection to every other member in the SSI cluster. SSI traffic from one member to another never flows through an intermediate member. Figure 42 on page 731 illustrates this topology with no intermediaries, sometimes called "fully connected."

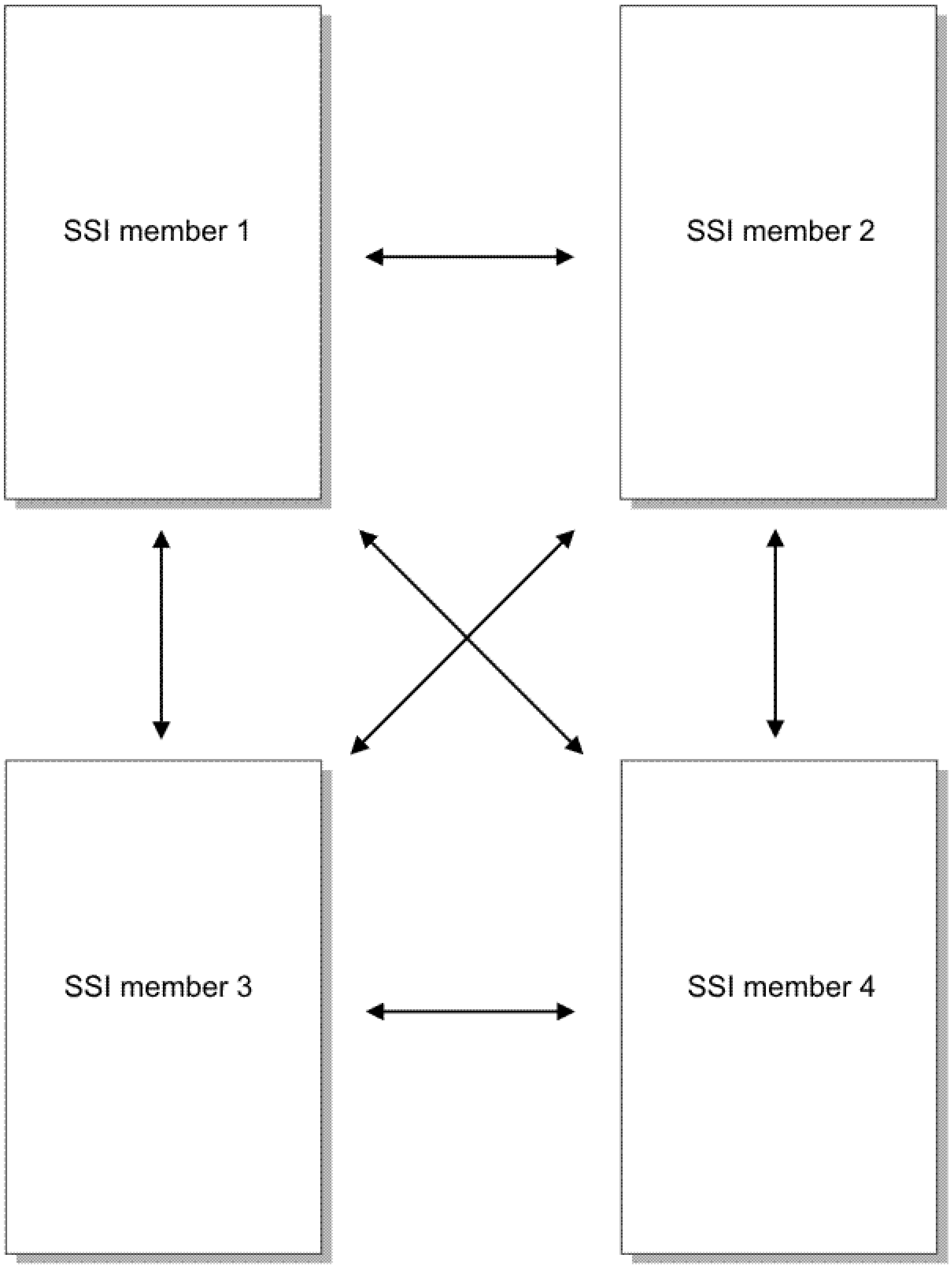


Figure 42. Fully Connected ISFC Topology

The ISFC connection from one member to another is called an *ISFC logical link*, or simply a *logical link*. There is always exactly one ISFC logical link between two members. Figure 43 on page 732 illustrates that there is exactly one ISFC logical link connecting member 1 to member 2.

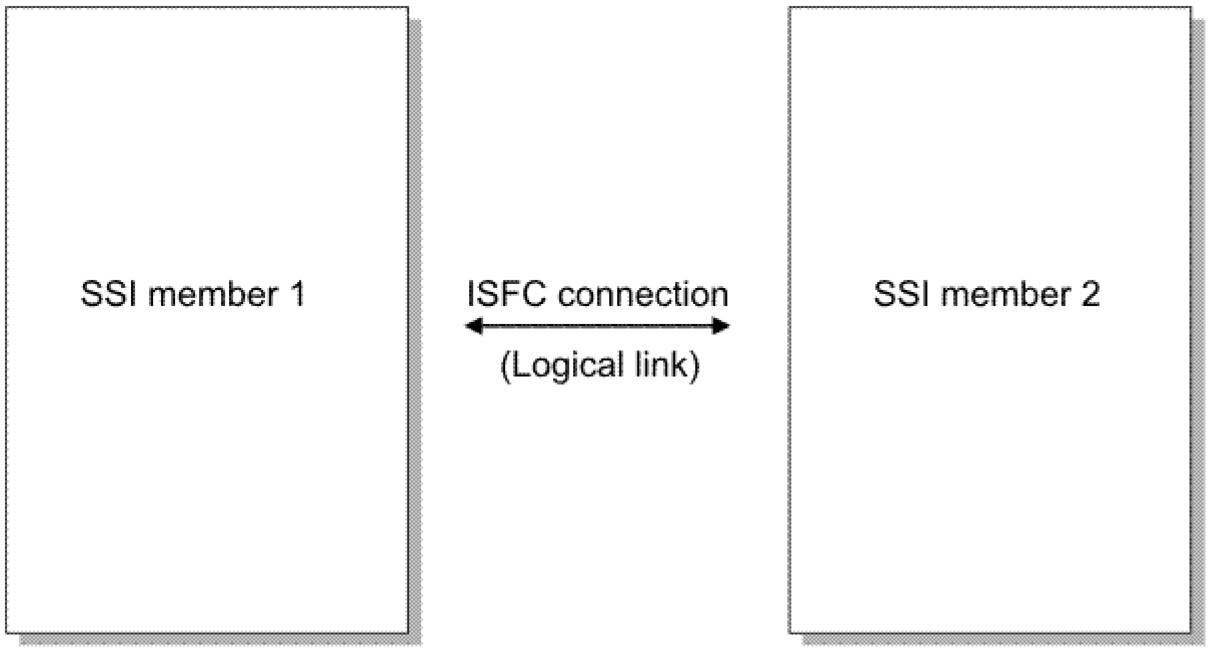


Figure 43. Logical Link

A logical link is composed of 1 - 16 channel-to-channel (CTC) devices. The channel paths on which these CTCs reside can be either unswitched channel paths or switched channel paths. For example, the logical link might be composed of eight CTCs spread among two unswitched FICON channel paths. Figure 44 on page 732 illustrates the example, depicting four CTCs on one FICON channel path and four more CTCs on another FICON channel path. Together these eight CTCs compose the logical link.

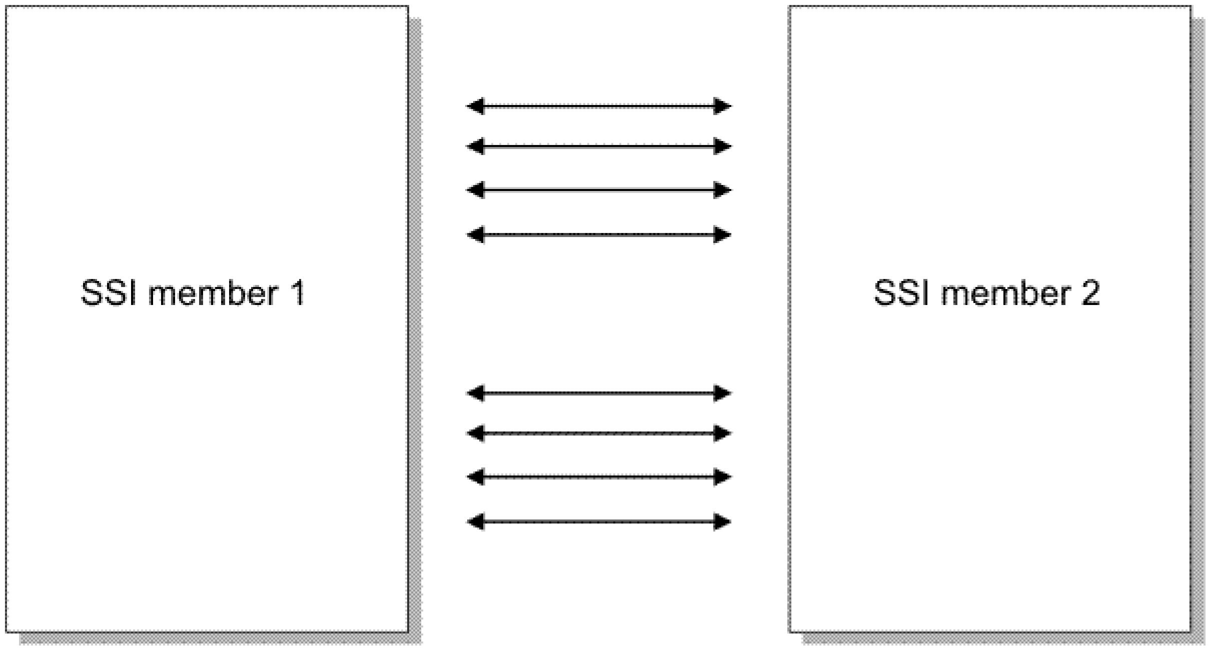


Figure 44. One Logical Link Composed of Multiple CTC Devices

For best operation, follow these guidelines for configuring logical links:

- Compose the logical link with more than one CTC. Having more than one CTC lets the two members avoid a CTC phenomenon called *write collisions*. A write collision happens when the two members both try to transmit data on the same CTC at the same instant in time. Write collisions have a severe negative impact on the performance of the logical link. When the logical link is composed of more than one CTC, write collisions rarely occur.

- Distribute the logical link CTCs among multiple FICON channel paths. For example, if you were going to use six CTCs for your logical link, place three of the CTCs on one FICON channel path and three on another. Using multiple channel paths helps protect the logical link from becoming severed in the event that a fiber is inadvertently disconnected or damaged.
- Performance measurements of guest relocation workloads suggest that ISFC is able to use the entire data-carrying capacity of a FICON channel path's optical fiber when only four CTCs are configured on that channel path. Thus, it is generally advised not to put more than four CTCs onto any one FICON channel path.
- As a preferred practice, make sure that for each CTC you use, the real device number of the CTC is the same on the two members. For example, CTC 6000 on member 1 would connect to CTC 6000 on member 2. Using different device numbers on the two ends of a CTC leads to confusion.
- If possible, use blocks of contiguous device numbers for the CTCs of a logical link. For example, CTCs 6000 - 6003 on member 1 would connect to CTCs 6000 - 6003 on member 2.
- Configure your logical link so that it exhibits *symmetric performance*. Symmetric performance means that a given virtual server relocates from member 1 to member 2 with the same performance as when the virtual server relocates from member 2 to member 1.
  - Whether a logical link exhibits symmetric performance is determined by the speeds of the FICON channel paths on which the logical link's CTCs reside.
  - Keep in mind that the speed of a FICON channel path is determined by the types of FICON Express adapters used on the two ends, and by the type of Gigabit Interface Converter (GBIC) installed into each of the two FICON channel adapters; and, if the FICON channel is switched, then also by the GBICs installed into the FICON switch and by any management or tuning selections made in the FICON switch.
  - A logical link composed of FICON channel paths of all the same speed always has symmetric performance.
  - A logical link composed of FICON channel paths of differing speeds must be built very carefully to assure symmetric performance.
  - If the FICON channel paths have differing speeds, symmetric performance is achieved by carefully choosing the relationship between CTCs, their device numbers, and their FICON channel path speeds, *on the end of the logical link whose z/VM member name comes first in the alphabet.*

    On the end whose z/VM member name comes first in the alphabet, make sure that you arrange the CTCs and channel paths so that when you sort the device numbers of the CTCs in ascending order, the CTCs in the middle of the sorted list are on the slow FICON channel paths and the CTCs on the ends of the sorted list are on the fast FICON channel paths.

    For example, if you have:

    - z/VM members NYORKVM and BOSTONVM connected by a logical link
    - The logical link is composed of sixteen CTCs spread among four FICON channel paths
    - Two of those FICON channel paths are 2 Gb/sec links and two others are 4 Gb/sec links

    you could assure symmetric performance by configuring the relationship between CTC device numbers and FICON channel path speeds on the BOSTONVM side to look like Figure 45 on page 734:

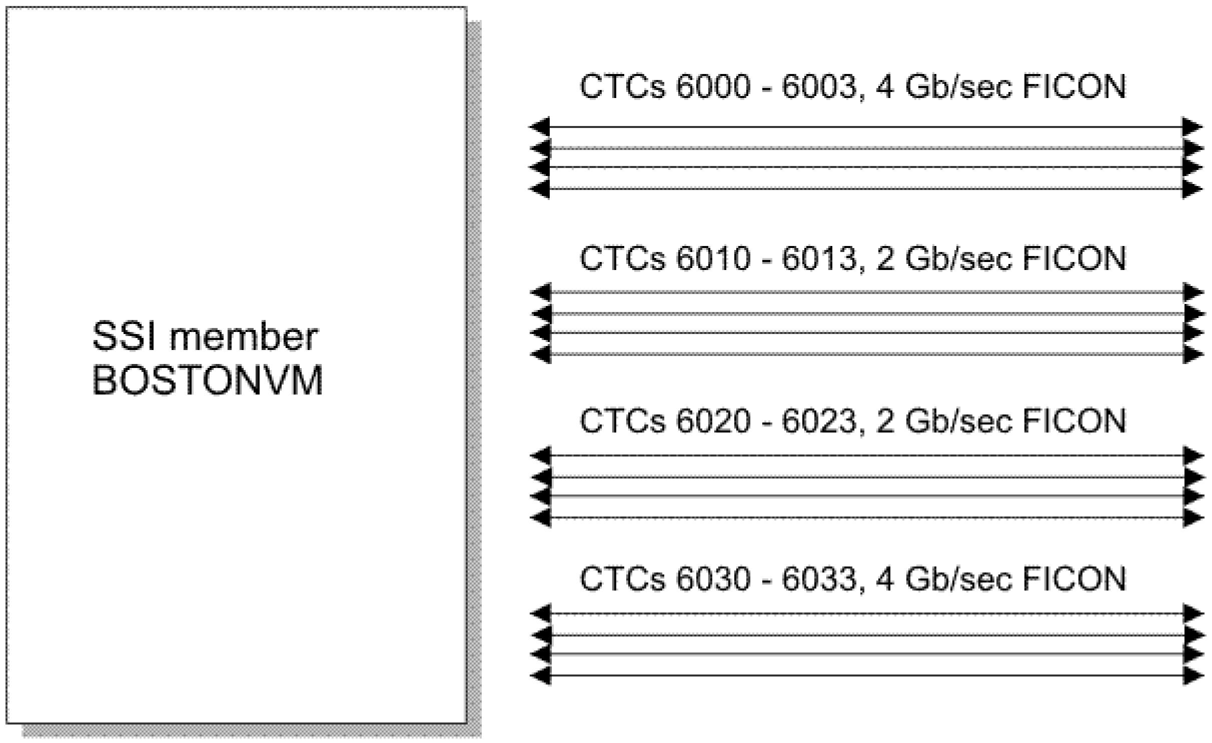

SSI member BOSTONVM
CTCs 6000 - 6003, 4 Gb/sec FICON
CTCs 6010 - 6013, 2 Gb/sec FICON
CTCs 6020 - 6023, 2 Gb/sec FICON
CTCs 6030 - 6033, 4 Gb/sec FICON

Figure 45. Example of Symmetric Performance with FICON Channel Paths of Differing Speeds

For symmetric performance, it makes no difference what the CTC device numbers are on the NYORKVM side. However, for manageability purposes and to avoid confusion, it is a preferred practice that CTC device numbers match on the two ends of the logical link.

- You can build the logical link out of more CTCs and faster FICON channel paths than you think your relocation workload can saturate. There is no negative performance consequence for having over-designed or over-provisioned the logical link.

Including an SSI Cluster in a Larger ISFC Collection

The members of an SSI cluster are more tightly connected than typical nodes in an ISFC collection and are subject to more stringent timing constraints. If an SSI cluster is included in a larger ISFC collection, you should avoid creating a connectivity structure that requires the members of the SSI cluster to carry traffic for the collection that might interfere with critical communications within the cluster. As shown in Figure 46 on page 735, the communication links should be designed so that the members of the SSI cluster do not have to act as intermediate nodes for ISFC nodes that are not members of the same SSI cluster.

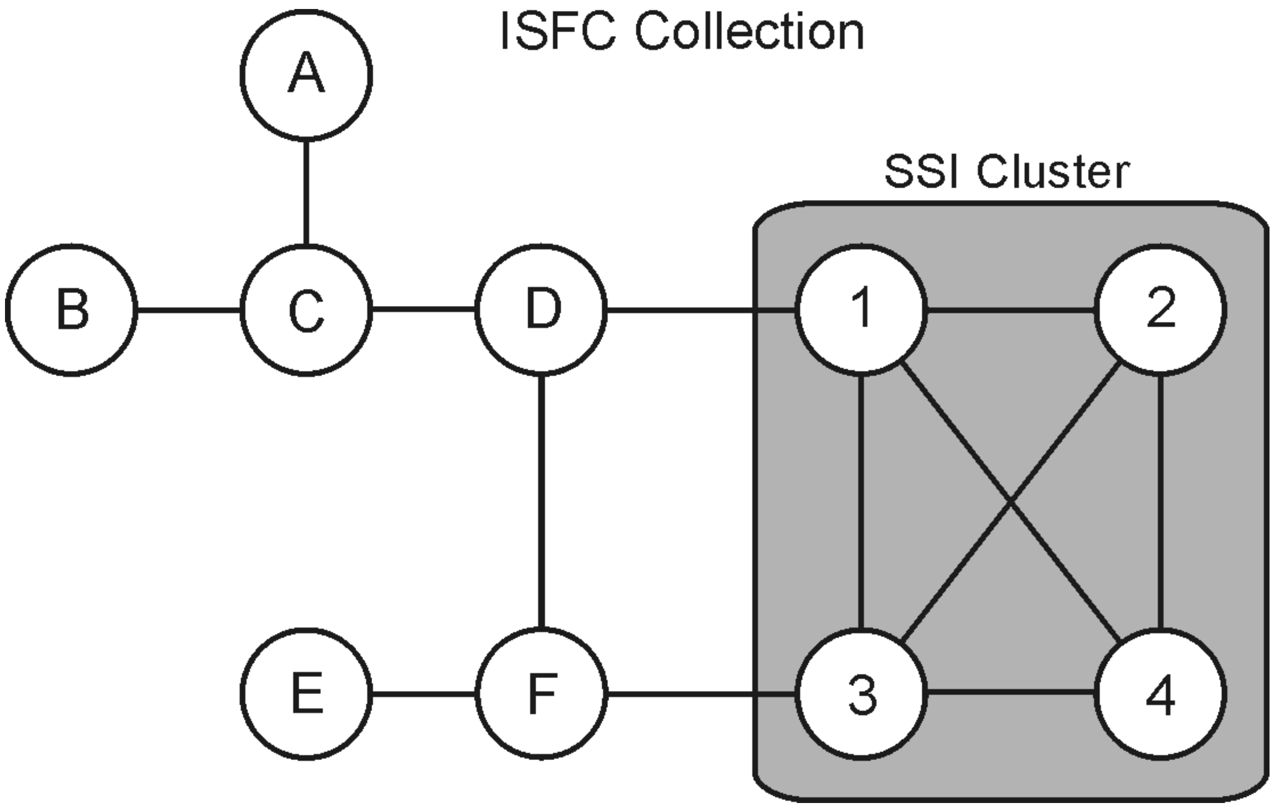


Figure 46. Example of an SSI Cluster within a Larger ISFC Collection

Creating a z/VM SSI Cluster

A z/VM SSI cluster can be created all at once or 1 member at a time. Table 49 on page 735 identifies tasks for creating an SSI cluster and adding or removing members and indicates where you can find the corresponding information and procedures.

Note: Creating a single-member SSI cluster is intended to be the first step in moving to a multimember SSI cluster. At a later time, the single member can be cloned to add other members to the cluster.

Table 49. Creating an SSI Cluster and Adding or Removing Members

Task	Information and procedures
Install an SSI cluster consisting of up to 4 z/VM images.	Use one of the SSI installation procedures documented in *z/VM: Installation Guide*. **Notes:** 1. You can install an SSI cluster with more members than you need right now. The installation program will install the z/VM images and create the cluster structure for the additional members, and you can activate them when you need them. 2. You can install the SSI cluster first-level (where the members will be IPLed in logical partitions) or second-level (where the members will be IPLed in virtual machines). A second-level SSI cluster can be moved to first-level at another time.
Convert a non-SSI z/VM 6.4 or later system to a single-member SSI cluster.	Follow the instructions in Chapter 32, “Converting a z/VM System to a Single-Member z/VM SSI Cluster,” on page 763.

Table 49. Creating an SSI Cluster and Adding or Removing Members (continued)

Task	Information and procedures
Combine two non-SSI z/VM 6.4 or later systems to create an SSI cluster with two members.	Follow the instructions in Chapter 33, "Combining Two Non-SSI z/VM Systems to Create a z/VM SSI Cluster," on page 779.
Clone an existing member to create a new member in an SSI cluster	Follow the instructions in Chapter 34, "Adding a Member to a z/VM SSI Cluster by Cloning an Existing Member," on page 785.
Add 1 to 4 more members to a 4-member SSI cluster.	Follow the instructions in Chapter 35, "Adding Members to a 4-Member SSI Cluster," on page 811.
Move an SSI cluster installed second-level (where the members are IPLed in virtual machines) to first-level (where the members are IPLed in logical partitions).	Follow the instructions in Chapter 36, "Moving a Second-Level z/VM SSI Cluster to First-Level," on page 815.
Remove a member from an SSI cluster permanently.	Follow the instructions in Chapter 37, "Decommissioning a Member of a z/VM SSI Cluster," on page 821.
Convert a CSE complex to an SSI cluster.	See Chapter 38, "Converting a CSE Complex to a z/VM SSI Cluster," on page 835.
Migrate a pre-z/VM 6.2 system (not part of a CSE complex) to z/VM 7.3 and convert it into the initial member of an SSI cluster.	This process has three phases: 1. Install a z/VM 7.3 system using one of the non-SSI installation procedures documented in *z/VM: Installation Guide*. 2. Transfer the files, applications, and other information you need from your old system to the new system. To ensure a smooth upgrade to the new release, follow the z/VM migration procedure documented in *z/VM: Installation Guide*. 3. Convert the new system to a single-member SSI cluster. Follow the instructions in Chapter 32, "Converting a z/VM System to a Single-Member z/VM SSI Cluster," on page 763.

z/VM SSI Cluster Operation

A z/VM system that is configured as a member of an SSI cluster joins the cluster when the system is IPLed. To successfully join the cluster, the member:

- Verifies that its configuration is compatible with the cluster
- Establishes communication with other members
- Verifies compatible definitions with the other members

A member leaves the cluster when it shuts down.

The state of each member and the operating mode of the cluster determine the degree of communication, resource sharing, and data sharing among the members. The operating mode is determined by the states of all the members at any point in time.

Cluster Modes

The operating modes of an SSI cluster are:

Stable
The SSI cluster is fully operational.

Influx
A member is joining or leaving the SSI cluster.

The cross-system functions of the SSI cluster are temporarily suspended, and negotiations on access to shared resources are deferred. Accesses already negotiated are not affected.

This is a transient condition. No intervention is required unless a problem occurs.

Safe
A member of the SSI cluster is in an unknown state.

Communication with a member has been lost or has timed out without the member formally leaving the cluster. To preserve data integrity, new attempts to access shared resources will fail until the situation is resolved. Accesses already negotiated are not affected.

This mode is automatically corrected if communication is restored or if the member is recognized as being in the Down state. If necessary, this mode can be corrected manually by an operator who confirms that the system is down and declares it so with the SET SSI command.

Normal Member States

Typically, each member of an SSI cluster will be in one of the following states:

Down
A member is in the Down state when:

- The member has not been IPLed.
- The member has been IPLed but has not yet joined the cluster.
- The member has been shut down.
- The member has abended but has not yet restarted.
- The SET SSI command has been issued from another member to declare this member is down.

Joined
Usually, all active (IPLed) members are in the Joined state and are fully participating in SSI cluster operations. When all active members are in the Joined state, the cluster is in the Stable mode.

Joining
When a member is in the Joining state, it is in the process of exchanging information and verifying compatible definitions with the Joined members. A member is in the Joining state during IPL or when it is recovering from an error state (see “Member Error States” on page 737). When a member being IPLed is in the Joining state, the cluster is in the Influx mode. When a member recovering from an error condition is in the Joining state, the cluster can be in either the Safe mode or the Influx mode. This is a transient state and mode combination that lasts only for a short time. The Joining member will transition to the Joined state and the cluster will return to the Stable mode.

Leaving
When the SHUTDOWN command is issued on a member that is in the Joined state, that member's state changes to Leaving. When a member is in the Leaving state, the cluster is in the Influx mode. This is a transient state and mode that lasts only for a short time. The Leaving member will transition to the Down state and the cluster will return to the Stable mode.

Note: Only one member can be in the Joining state or the Leaving state at any given time.

Member Error States

When an error condition occurs in the cluster, communication and exchange of information about shared resources that normally occurs among members might be inhibited. This type of error results in one of the following states:

Suspended
A member enters the Suspended state when it unexpectedly loses connectivity to another member. When a member is in the Suspended state, the cluster is in the Safe mode. Depending on the cause of the error condition, this state and mode might be either transient or long lasting.

When any member enters the Suspended state, all active members must enter the Suspended state before the cluster can recover and return to a normal mode.

Some error conditions that result in the Suspended state will be corrected automatically, but others might require operator intervention. When the error condition no longer exists, each member automatically will go through the Joining process. Eventually all members will return to the Joined state and the cluster will return to the Stable mode.

Isolated

When an error occurs that prevents a member from joining the cluster during IPL, the member enters the Isolated state. An isolated member does not communicate with the other members and does not participate in SSI cluster operations.

When a member is in the Isolated state, that member sees the cluster mode as Safe. The states of the other members and the overall cluster mode are not affected by the state of the isolated member.

When the error is corrected, the isolated member must be shut down and re-IPLed in order to attempt to join the cluster again.

Checking the Member States and the Cluster Mode

Each member of the cluster keeps track of the states of all members as well as the mode of the cluster. Normally, all members' views of each member's state are the same. However, in some error conditions, the members' views of the other members' states might not match. Each member's state is also recorded in the persistent data record (PDR).

The QUERY SSI command can be used to check the states of all the members and the mode of the cluster. For example:

```
query ssi

SSI Name: CLUSTERA
SSI Mode: Influx
Cross-System Timeouts: Enabled
SSI Persistent Data Record (PDR) device: VMCOM1 on EFE0
SLOT SYSTEMID STATE     PDR HEARTBEAT       RECEIVED HEARTBEAT

   1 VMSYS01  Joined    2022-07-11 21:22:00 2022-07-11 21:22:00
   2 VMSYS02  Joined    2022-07-11 21:21:40 2022-07-11 21:21:40
   3 VMSYS03  Joining   2022-07-11 21:21:57 None
   4 VMSYS04  Down (not IPLed)
```

The FORMSSI utility (located on the PMAINT 551 disk) can be used to check the states of all the members (as well as other information) in the PDR. For example:

```
formssi display efe0

HCPPDF6618I Persistent Data Record on device EFE0 (label VMCOM1) is for CLUSTERA
HCPPDF6619I PDR                 state: Unlocked
HCPPDF6619I                time stamp: 07/11/22 21:22:03
HCPPDF6619I     cross-system timeouts: Enabled
HCPPDF6619I PDR slot  1        system: VMSYS01
HCPPDF6619I                     state: Joined
HCPPDF6619I                time stamp: 07/11/22 21:22:00
HCPPDF6619I               last change: VMSYS01
HCPPDF6619I PDR slot  2        system: VMSYS02
HCPPDF6619I                     state: Joined
HCPPDF6619I                time stamp: 07/11/22 21:21:40
HCPPDF6619I               last change: VMSYS02
HCPPDF6619I PDR slot  3        system: VMSYS03
HCPPDF6619I                     state: Joining
HCPPDF6619I                time stamp: 07/11/22 21:21:57
HCPPDF6619I               last change: VMSYS03
HCPPDF6619I PDR slot  4        system: VMSYS04
HCPPDF6619I                     state: Down
HCPPDF6619I                time stamp: 07/02/22 17:02:25
HCPPDF6619I               last change: VMSYS02
```

SSI Cluster Problem Diagnosis

Problem diagnosis and resolution procedures in an SSI cluster depend on the type of problem and whether the problem occurred during system IPL or normal system operation.

Tip: One way to help avoid system IPL problems is to use the CPSYNTAX utility to validate the syntax of the common system configuration file.

Table 50. Diagnosing and Resolving Problems in an SSI Cluster

Problem	Diagnosis and Resolution
During system IPL	
Member is active but in the Isolated state.	The error messages that were written to the operator's console can be used to help determine the cause of the problem.
Member is stopped in a disabled wait.	1. The error messages that were written to the operator's console can be used to help determine the cause of the problem. 2. If the problem occurred before the operator's console was initialized, error messages might have been written to the line mode system console: • When the member is running in a logical partition (first level), these messages appear in the Operating System Messages window on the Hardware Management Console (HMC). • When the member is running as a guest (second level), these message appear on your virtual machine console. **Note:** If the console screen clears before you are able to read the messages, you can display the messages (and force them to remain on the screen) by setting your console mode to 3215 and IPLing with the LOADPARM CONSSYSC operand. For example: `IPL 1111 CLEAR LOADPARM CONSSYSC`
Member has a configuration error.	1. Log on as a privileged user ID on a different active member of the SSI cluster and fix the error. 2. If the error occurred on a first level system and no other member is active, you might need to IPL with the REPAIR parameter. For more information, see Passing IPL Parameters in *z/VM: System Operation*. 3. If the error occurred when IPLing a guest (second level) system, fix the error from the first level user ID.
Communication to the other members cannot be established.	During system IPL, a member must establish communication to all other active members of the cluster. System IPL will not proceed until all necessary communication is established. Ensure that all joined remote members have active ISLINKs (ISFC connections) to the member being IPLed.
During normal operation	
Communication among the members is inhibited.	If a hardware or software malfunction causes an unexpected loss of connectivity in the cluster, each member enters the "Suspended" state. 1. Issue the QUERY SSI CONNECTIVITY and QUERY ISLINK commands. Analyze the responses from these commands to determine which ISFC connection is not working. 2. When the error condition is fixed, the members will automatically go through the Joining process.

Table 50. Diagnosing and Resolving Problems in an SSI Cluster (continued)

Problem	Diagnosis and Resolution
Connectivity has been temporarily lost due to a system abend.	A system abend has caused a loss of connectivity while the member dumps and restarts: 1. Before dumping, the member specified its state as Down in the PDR. 2. Once all of remaining members recognize this state change, after they have all entered the Suspended state they will go through the Joining process, and the cluster will go back to the Stable mode without the member that abended. 3. Eventually the member that abended will restart and go through the Joining process to rejoin the cluster.
Member is in a disabled loop.	1. The system operator for that member should invoke the system restart function to cause the member to go through system termination and specify its state as Down in the PDR. 2. When all of remaining members recognize this state change, after they have all entered the Suspended state they will go through the Joining process, and the cluster will go back to the Stable mode without the member that is down. 3. In the rare case that a member is not active and has not marked itself as Down in the PDR: a. A privileged user ID on any active member can issue the following command to specify the member is down. `set ssi member sysname down` b. When all of remaining members recognize this state change, after they have all entered the Suspended state they will go through the Joining process, and the cluster will go back to the Stable mode without the member that is down.

Considerations for Migrating the SSI Common Volume

The SSI common volume (default label VMCOM1) contains the PDR, the system configuration file, and other shared data files. Moving the common volume includes the following tasks:

1. Relocate the PDR by issuing the SET SSI PDRVOLUME command and update the system configuration file with the new volume ID.
2. Relocate the minidisks on the volume:
   - PMAINT CF0, which contains the system configuration file (must be allocated as PERM space)
   - PMAINT 2CC, which contains the source directory (if a directory manager is not in use)
   - PMAINT 41D, which is the VMSES/E production inventory disk
   - PMAINT 551, which contains common utilities for the SSI cluster
   - The minidisk that contains the common VMPSFS file pool

     The SFS server must be shut down during this process.
   - The minidisk that contains the DirMaint database

     If DirMaint is enabled, it must be shut down during this process.
   - The service disks for cross-release facilities, such as OSA and VMHCD

During this process, the service virtual machines dependent on these minidisks must be shut down, and related activities such as software service must be suspended.

3. Update all the IPL volumes to point to the system configuration file in its new location. This involves running the SALIPL utility with the IPLPARMS option to specify the updated PDVOL= value.
4. Update the user directory to reflect the new locations of all the minidisks formerly located on the old volume.

Implications for Vendor Products and Customer Applications

Setting up an SSI cluster can have implications for vendor products and customer applications, such as those identified in the following topics. For information about how a particular vendor product works with SSI clusters, contact the vendor.

Service Virtual Machines

Depending on the application, a service virtual machine (SVM) can be defined in one of two ways:

- A single instance within the SSI cluster

  An SVM defined by a single-configuration virtual machine definition (USER definition) can log on to any member of the SSI cluster, but can be logged on to only one member at a time. In this case, the SVM will probably be useful only if it can supply services to virtual machines running on any member of the cluster.
- An instance on each member of the SSI cluster

  An SVM defined by a multiconfiguration virtual machine definition (IDENTITY definition) can be logged on simultaneously to multiple members of the SSI cluster. Separate identity and subconfiguration entries in the definition can be used to tailor the SVM instance for each member of the cluster. In this situation, it is likely that each instance of the SVM will provide services only to the virtual machines that are running on the same member as that instance.

  Using shared configuration files can simplify the deployment of member-specific SVMs. Member-specific configuration files for SVMs can be provided by placing the files on member-specific volumes, on local minidisks owned by multiconfiguration virtual machines, or on shared volumes using member-specific naming conventions.

Guest Relocation

A key aspect of an SSI cluster is the ability of single-configuration virtual machines to run on any member of the cluster (one member at a time). A guest virtual machine can move between members by simply logging off one member and logging on another. Alternatively, the live guest relocation function can be used to move the guest from one member to another while still running. As a result, there might be implications for products and applications, particularly with respect to guest relocations. When reviewing the level of awareness of guest relocation:

- Products that do entitlement checking based on system name, processor CPUID, and other single-member criteria need to accommodate multiple possibilities if entitlement is for an SSI cluster.
- Products that do internal resource accounting or performance monitoring might need to recognize that different members of a cluster could exist on different physical servers and therefore have different performance characteristics, such as processor speed.
- When the implementation involves client and server virtual machines, the implementation or configuration might need to address communication or authentication requirements.
- Products that monitor other virtual machines for performance, accounting, availability, or other reasons might need to make changes to handle guests on different systems and guest relocations, where virtual machines seem to disappear or appear without logging off or on.
- Products sensitive to time might need to make changes to accommodate time differences between the members of the cluster.

System Configuration File for a z/VM SSI Cluster

Table 51 on page 742 shows examples of some of the statements in a system configuration file for a four-member SSI cluster and provides a brief explanation of how each is used. Depending on the installation configuration, other statements can also be used. For complete information about all system configuration statements, see Chapter 6, "The System Configuration File," on page 47.

Table 51. Examples of Statements in a System Configuration File for an SSI Cluster

Extract	Explanation

`/**********************************************************/`
`/*     System_Identifier Information                     */`
`/**********************************************************/`
 	
` System_Identifier LPAR LP01 VMSYS01`
` System_Identifier LPAR LP02 VMSYS02`
` System_Identifier LPAR LP03 VMSYS03`
` System_Identifier LPAR LP04 VMSYS04`
	A **SYSTEM_IDENTIFIER** statement is required for each member of the SSI cluster, to define the unique identifier (system name) for the z/VM system to be run in a specified logical partition (LPAR). The system name on this statement (for example, VMSYS01) is used as the record qualifier parameter on configuration statements that apply only to that member of the cluster. When a member system is initialized in an LPAR, CP locates the SYSTEM_IDENTIFIER statement with that LPAR name (for example, LP01) and uses the system name to select the other configuration statements to be processed for that member — all statements qualified with that identifier and all statements with no qualifier.
	
	If the SSI cluster is installed to be IPLed second level, "LPAR *name*" identifies the first level user ID of the virtual machine in which the second level member will be run.

`/**********************************************************/`
`/*     SSI Statement                                     */`
`/**********************************************************/`
 	
` SSI CLUSTERA PDR_Volume VMCOM1 ,`
`      Slot 1 VMSYS01,`
`      Slot 2 VMSYS02,`
`      Slot 3 VMSYS03,`
`      Slot 4 VMSYS04`
	The **SSI** statement indicates this system configuration file defines an SSI cluster. The statement specifies the name of the cluster, the label of the DASD volume that contains the SSI persistent data record (PDR), and the system names of the members.

Table 51. Examples of Statements in a System Configuration File for an SSI Cluster (continued)

Extract	Explanation

`/*********************************************************/`
`/*     Relocation Domains                                */`
`/*********************************************************/`
 	
` Relocation_Domain DOMAIN1 Members VMSYS01 VMSYS03`
` Relocation_Domain DOMAIN2 Members VMSYS02 VMSYS04`
	RELOCATION_DOMAIN statements are optional and can be used to define subsets of the SSI cluster membership. A domain specifies a set of systems among which a guest that is associated with that domain (by a VMRELOCATE directory statement) may relocate freely. The virtual architecture of the domain ensures that a guest is presented with the same set of architectural facilities in whichever member of the domain the guest runs. For more information, see "Using Relocation Domains" on page 754.

`/*********************************************************/`
`/*     Checkpoint and Warmstart Information              */`
`/*********************************************************/`
 	
` VMSYS01:  System_Residence,`
`           Checkpoint  Volid M01RES   From CYL 21  For 9,`
`           Warmstart   Volid M01RES   From CYL 30  For 9`
` VMSYS02:  System_Residence,`
`           Checkpoint  Volid M02RES   From CYL 21  For 9,`
`           Warmstart   Volid M02RES   From CYL 30  For 9`
` VMSYS03:  System_Residence,`
`           Checkpoint  Volid M03RES   From CYL 21  For 9,`
`           Warmstart   Volid M03RES   From CYL 30  For 9`
` VMSYS04:  System_Residence,`
`           Checkpoint  Volid M04RES   From CYL 21  For 9,`
`           Warmstart   Volid M04RES   From CYL 30  For 9`
	A **SYSTEM_RESIDENCE** statement is required for each member to identify its checkpoint and warm start areas.

Table 51. Examples of Statements in a System Configuration File for an SSI Cluster (continued)

Extract	Explanation

```
/**********************************************************/
/*     CP_Owned Volume Statements                         */
/**********************************************************/
/*          RES volume                                    */
/**********************************************************/

 VMSYS01: CP_Owned   Slot   001  M01RES
 VMSYS02: CP_Owned   Slot   001  M02RES
 VMSYS03: CP_Owned   Slot   001  M03RES
 VMSYS04: CP_Owned   Slot   001  M04RES

/**********************************************************/
/*          COMMON volume                                 */
/**********************************************************/

 CP_Owned   Slot  005  VMCOM1

/**********************************************************/
/*          DUMP & SPOOL volumes                          */
/**********************************************************/

 CP_Owned   Slot  010  M01S01
 CP_Owned   Slot  011  M02S01
 CP_Owned   Slot  012  M03S01
 CP_Owned   Slot  013  M04S01

/**********************************************************/
/*          PAGE & TDISK volumes                          */
/**********************************************************/

 VMSYS01: BEGIN
          CP_Owned   Slot 254  M01T01
          CP_Owned   Slot 255  M01P01
 VMSYS01: END


 VMSYS02: BEGIN
          CP_Owned   Slot 254  M02T01
          CP_Owned   Slot 255  M02P01
 VMSYS02: END


 VMSYS03: BEGIN
          CP_Owned   Slot 254  M03T01
          CP_Owned   Slot 255  M03P01
 VMSYS03: END


 VMSYS04: BEGIN
          CP_Owned   Slot 254  M04T01
          CP_Owned   Slot 255  M04P01
 VMSYS04: END
```
	CP_OWNED statements for the system residence volume and paging and temporary disk volumes must be qualified for each member. CP_OWNED statements for the common (parm disk) volume and dump and spool volumes should not be qualified, so those volumes can be shared.
	
	Note: The OWN and SHARED operands are ignored when the system configuration file defines an SSI cluster.

Table 51. Examples of Statements in a System Configuration File for an SSI Cluster (continued)

Extract	Explanation

`/*********************************************************/`
`/*     User_Volume_List Statements                       */`
`/*********************************************************/`
`/*          COMMON user volumes                          */`
`/*********************************************************/`
 	
` User_Volume_List 730RL1 730RL2 USRVL1`
 	
`/*********************************************************/`
`/*          Member-specific user volumes                 */`
`/*********************************************************/`
 	
` VMSYS01: User_Volume_List M01PV1`
 	
` VMSYS02: User_Volume_List M02PV1`
 	
` VMSYS03: User_Volume_List M03PV1`
 	
` VMSYS04: User_Volume_List M04PV1`
	At least one **USER_VOLUME_LIST** statement should be qualified for each member to identify volumes to be used for member-specific minidisks. USER_VOLUME_LIST statements for common user volumes should not be qualified, so those volumes can be shared.

`/*********************************************************/`
`/*     DISTRIBUTE Statement                              */`
`/*********************************************************/`
 	
` Distribute IUCV NO`
	For an SSI cluster, the **DISTRIBUTE IUCV** statement governs use of this feature beyond the cluster (for example, if the cluster is part of a larger ISFC collection). NO is the default value. However, regardless of the setting, cross-system IUCV is automatically available among the members of the cluster.

`/*********************************************************/`
`/*     Activate ISLINK statements                        */`
`/*********************************************************/`
`/*********************************************************/`
`/*          ISFC links for Member 1                      */`
`/*********************************************************/`
 	
` VMSYS01: Activate ISLINK m1_rdev1 m1_rdev2 Node VMSYS02`
` VMSYS01: Activate ISLINK m1_rdev3 m1_rdev4 Node VMSYS03`
` VMSYS01: Activate ISLINK m1_rdev5 m1_rdev6 Node VMSYS04`
 	
`/*********************************************************/`
`/*          ISFC links for Member 2                      */`
`/*********************************************************/`
 	
` VMSYS02: Activate ISLINK m2_rdev1 m2_rdev2 Node VMSYS01`
` VMSYS02: Activate ISLINK m2_rdev3 m2_rdev4 Node VMSYS03`
` VMSYS02: Activate ISLINK m2_rdev5 m2_rdev6 Node VMSYS04`
 	
`/*********************************************************/`
`/*          ISFC links for Member 3                      */`
`/*********************************************************/`
 	
` VMSYS03: Activate ISLINK m3_rdev1 m3_rdev2 Node VMSYS01`
` VMSYS03: Activate ISLINK m3_rdev3 m3_rdev4 Node VMSYS02`
` VMSYS03: Activate ISLINK m3_rdev5 m3_rdev6 Node VMSYS04`
 	
`/*********************************************************/`
`/*          ISFC links for Member 4                      */`
`/*********************************************************/`
 	
` VMSYS04: Activate ISLINK m4_rdev1 m4_rdev2 Node VMSYS01`
` VMSYS04: Activate ISLINK m4_rdev3 m4_rdev4 Node VMSYS02`
` VMSYS04: Activate ISLINK m4_rdev5 m4_rdev6 Node VMSYS03`
	A set of **ACTIVATE ISLINK** statements is required for each member of the SSI cluster, to identify channel-to-channel adapters (CTCAs) for ISFC links to the other members. Each member must have at least one direct ISFC link to each of the other members; this example shows devices for two links between members. The node ID specified on each statement is the system name of the member to be linked.

Chapter 30. Cross-System Spool in a z/VM SSI Cluster

With cross-system spool, the CP spooling system can include up to 8 z/VM systems connected in an SSI cluster. Spool volumes are shared in an SSI cluster. Each member of the cluster creates spool files only on spool volumes that it owns, but each member has access to the spool volumes owned by the other members.

Example of Cross-System Spooling

The following example shows how cross-system spooling works for a user in a four-member SSI cluster.

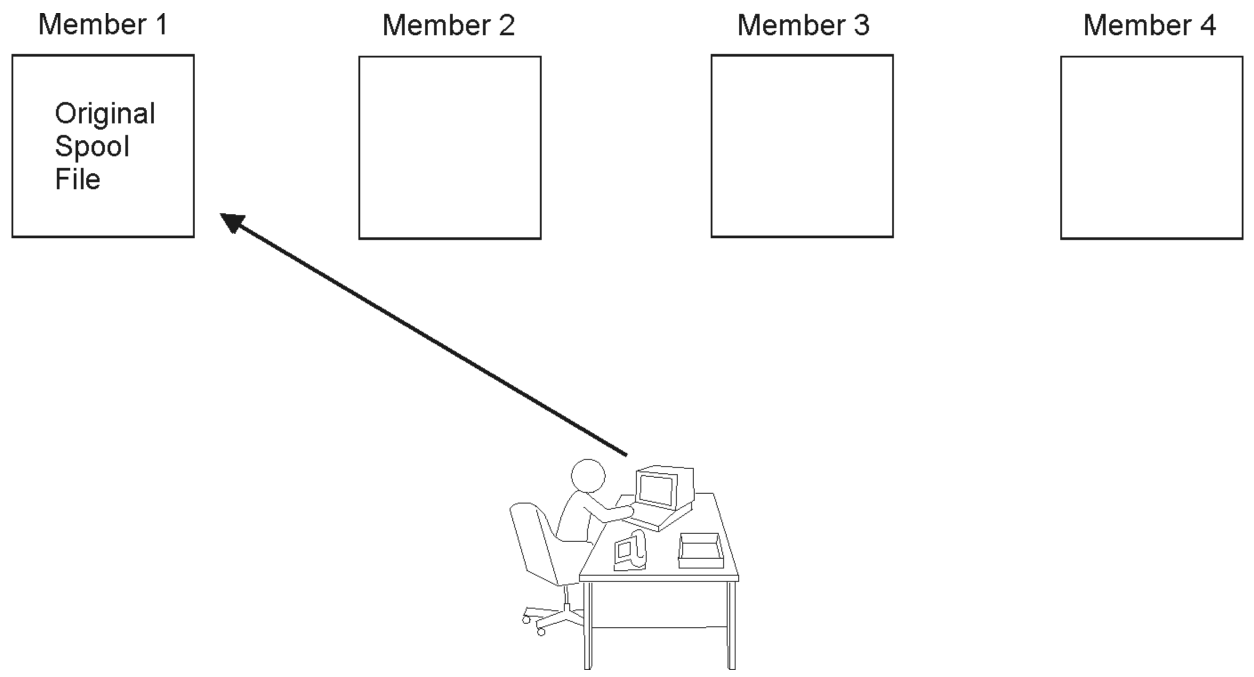


Figure 47. Example of Cross-System Spooling in an SSI Cluster, Part 1

In Figure 47 on page 747:

1. The user logs on to member 1.
2. The user creates a reader spool file on member 1.

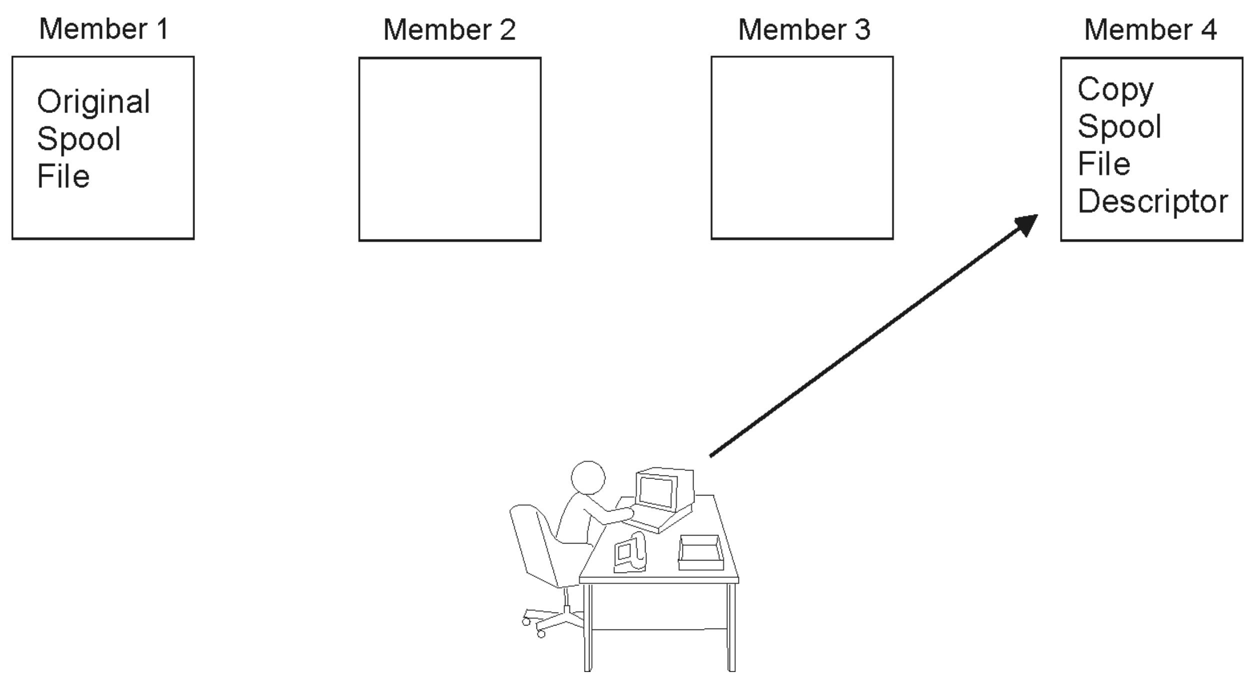


Figure 48. Example of Cross-System Spooling in an SSI Cluster, Part 2

In Figure 48 on page 748:

1. The user logs off from member 1 and logs on to member 4.
2. Member 1 sends member 4 a copy of the spool file descriptor from the original spool file.
3. The user sees the spool file as being on member 4 and can process the reader spool file there.

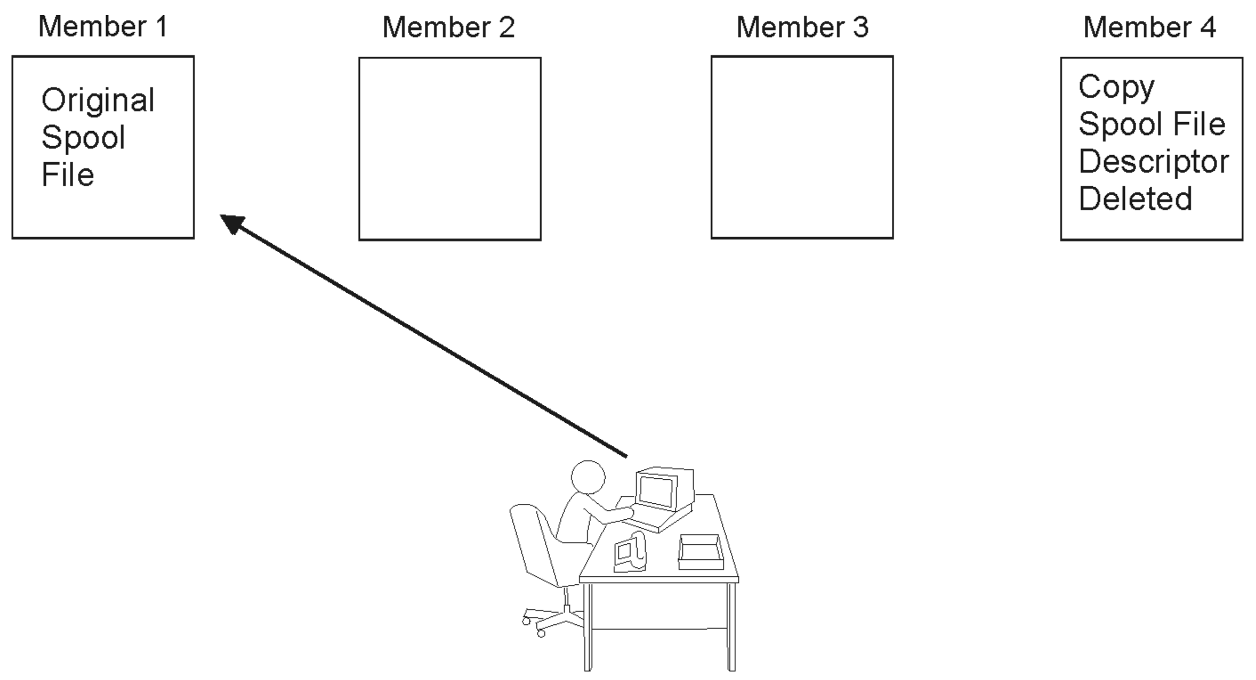


Figure 49. Example of Cross-System Spooling in an SSI Cluster, Part 3

In Figure 49 on page 748:

1. The user logs off from member 4 and logs on to member 1.

2. The user can process the reader spool file on member 1.
3. The copy of the spool file descriptor on member 4 is deleted.

How CP Spooling is Extended in an SSI Cluster

Within the SSI cluster, users defined as single-configuration virtual machines have a single logical view of all their spool files. Spooling devices (such as printers) are still controlled by the spooling operator logged on to the system that physically controls those devices. Users logged on to one of the members of the SSI cluster can access all of their spool files regardless of which member they have logged on to for that session. (For a spool file to be accessible, the member on which it was created must be joined to the cluster.) Users can transfer their spool files to other users or to service virtual machines anywhere in the cluster. They can select printers and other spool output devices for processing spool files created on any member of the cluster. Thus, the end-user view of spooling is nearly transparent. Although the CP commands that manipulate spool files operate differently internally for an SSI cluster, additional user options or parameters are not required.

Note: User IDs defined as multiconfiguration virtual machines do not participate in cross-system spool. See "Spool Files for Multiconfiguration Virtual Machines" on page 751.

In an SSI cluster, CP spooling functions on each member of the cluster manage and share spool files cooperatively with the other members. Because of this, when all the members are operating normally, there is little awareness of the SSI cluster and no special measures are required either for users or for operators.

Cross-system spool is concerned only with closed spool files and not with the allocation of CP spool space. Each member of the cluster has its own set of spool volumes to allocate from exclusively. All other members have read and write access to these spool volumes because they are on shared DASD. However, the other members cannot allocate or release space on these volumes.

The member on which a spool file is created owns the spool space. The spool descriptor on that member is called an *original spool descriptor*. The identity of the originating member, the one that owns the spool space the file occupies, is kept in the spool descriptor. Another member can receive a copy of the original spool descriptor, which is known as a *copy spool descriptor*. Copy spool descriptors can be deleted by any of the members, but the actual returning of spool file resources is performed by the originating member. Each member has peer status. This means that each member has responsibility for its original spool descriptors and for keeping the other members' copy spool descriptors up to date.

CP tries to minimize the number of copy spool descriptors by capitalizing on the inherent differences between input and output spool files. Output spool files must be available on all members so that they can be handled by real printers or punches. Input spool files must be available only on the member where their owning user ID is active.

Because more users are involved, the number of spool file blocks in an SSI cluster might easily exceed the number normally expected on a single system. Across the SSI cluster, the total amount of real storage required for original and copy spool file blocks increases each time a member receives a copy spool file block.

Copy Input (Reader) Spool Files

When a spool file is closed and placed on the input (reader) queue and the owner is logged on to the same member of the SSI cluster, a copy spool descriptor is not sent to another member. There is just one original spool descriptor for this spool file. If the owner is logged on to a member other than the originating member, a copy input spool descriptor is sent to the member where the user is currently logged on. In this case, the original spool descriptor and one copy spool descriptor exist. If the owner is not logged on, only the original spool descriptor on the originating member exists.

When a user logs on, all other members of the cluster send a copy of any available input (reader) spool descriptors owned by the user to the member where the user is logged on. A copy input spool descriptor exists on a member as long as the owner remains logged on to that member, assuming nothing is done to change its place in the input queue. When a user logs off, all copy input spool descriptors on that member are deleted. Only the original spool descriptors remain on the originating member.

Note: If a reader spool file is being dumped to tape by the SPXTAPE DUMP command on the originating member, and the owner of the file is logged on to another member, there is no indication to the owner during the dump that the file is not available.

Copy Output (Print and Punch) Spool Files

A copy spool descriptor is sent to all members of the SSI cluster when a spool file is added to the output (printer or punch) queue. The spool descriptor remains on all members until the file is processed, purged, or transferred to the input queue. When a copy output spool descriptor is transferred to the input queue and the owner is logged on to the member that initiated the transfer, the copy output spool descriptor becomes a copy input spool descriptor and the original output spool descriptor becomes an original input spool descriptor. If the owner is not logged on to the member that originated the transfer, the copy output spool descriptor is deleted and the original output spool descriptor becomes an original input spool descriptor.

When a member receives a copy spool descriptor (output or input), it allocates the same spool ID for the file as the original, or it does not allow the copy spool descriptor to remain on this member. A file's spool ID is always the same on all members of the cluster.

Extended CP Spooling Commands

Cross-system spool extends the scope of the following CP commands to apply to spool files anywhere in the SSI cluster:

- CHANGE
- ORDER
- PURGE
- QUERY FILES
- QUERY { READER | PRINTER | PUNCH } ALTID
- QUERY { READER | PRINTER | PUNCH } XFER
- TAG
- TRANSFER

The syntax and operands of these commands are unchanged when used in an SSI cluster.

How the CMS Productivity Aids Process Spool Files in an SSI Cluster

In an SSI cluster, if a user issues the CMS NOTE, SENDFILE, or TELL command without specifying the node, and the recipient is a single-configuration virtual machine, the recipient can display or receive the note, file, or message when logged on to any member of the cluster. If the node is omitted and the recipient is a multiconfiguration virtual machine, the recipient can display or receive the note, file, or message only when logged on to the same member as the sender. (If the node is specified for either type of user, the communication is routed to RSCS, and RSCS must be running on both members.)

When a user in an SSI cluster issues the CMS PEEK, RDRLIST, or RECEIVE command, any spool file that resides in the shared spool of the cluster (except files transmitted through RSCS) will appear to be local. That is, the file will appear to originate from the member where the command is issued, although the file might have been created on another member of the cluster and reside on a spool volume owned by that member. (For files transmitted through RSCS, the actual origin node is displayed or recorded.)

These CMS commands do not distinguish between the instances of a multiconfiguration virtual machine. For example, if a multiconfiguration virtual machine is logged on to two members of the SSI cluster and both instances of the user ID send a file with the same file ID (but which could have different content) to the same user (logged on to any member of the cluster), in the recipient's RDRLIST display both files will appear to originate from the instance of the multiconfiguration virtual machine on the member where the recipient is logged on.

Spool File ID Assignment and Limits

The MAXSPOOL operand of the SPOOLFILE directory statement specifies the number of spool files that the user may own in the SSI cluster (collectively, on all members). The maximum number is 9999, which is the default.

If SSI_CONTROLS SPOOL_MEMBERS is set to 4

For each user, spool file IDs are assigned according to the relative position (slot number) of the member where the file is created in the list of members on the SSI system configuration statement. For each new spool file created by the user on that member, the spool ID is increased by 4, regardless of how many members are currently in the cluster. Therefore, on member 1, the user's spool files are assigned spool IDs 1, 5, 9, and so on, up to 9997. On member 2, the user's spool files are assigned spool ID numbers 2, 6, 10, and so on, up to 9998. On member 3, the user's spool files are assigned spool ID numbers 3, 7, 11, and so on, up to 9999. On member 4, the user's spool files are assigned spool ID numbers 4, 8, 12, and so on, up to 9996.

The maximum number of spool files that the user can create on any one particular member of the cluster is one-fourth of the MAXSPOOL value, regardless of how many members are currently in the cluster. If the MAXSPOOL value is not evenly divisible by 4, the distribution is determined by the members' slot numbers. Therefore, if the MAXSPOOL value is 9999, the user can create 2500 spool files on member 1, 2500 spool files on member 2, 2500 spool files on member 3, and 2499 spool files on member 4.

If SSI_CONTROLS SPOOL_MEMBERS is set to 8

For each user, spool file IDs are assigned according to the relative position (slot number) of the member where the file is created in the list of members on the SSI system configuration statement. For each new spool file created by the user on that member, the spool ID is increased by 8, regardless of how many members are currently in the cluster. Therefore, on member 1, the user's spool files are assigned spool IDs 1, 9, 17, and so on, up to 9993. On member 2, the user's spool files are assigned spool ID numbers 2, 10, 18, and so on, up to 9994. On member 8, the user's spool files are assigned spool ID numbers 8, 16, 24, and so on, up to 9992.

The maximum number of spool files that the user can create on any one particular member of the cluster is one-eighth of the MAXSPOOL value, regardless of how many members are currently in the cluster. If the MAXSPOOL value is not evenly divisible by 4, the distribution is determined by the members' slot numbers. Therefore, if the MAXSPOOL value is 9999, the user can create 1,250 spool files on members 1 to 7 and 1,249 on member 8.

Access to Spool Files

Privileged users have access to all output spool files from any of the members of the SSI cluster. Privileged users can access the reader files of other users only on the member where the owner is logged on and on the member having the original spool descriptor. However, because privileged users need access only to the spool files that their member can process or to the input files that exist on their member, this is not an important limitation.

Spool Files for Multiconfiguration Virtual Machines

User IDs defined as multiconfiguration virtual machines do not participate in cross-system spool. Each instance of a multiconfiguration virtual machine can view only the spool files owned by the instance on that member. A spool file owned by another instance of the virtual machine (that is, logged on to another member) is not visible. Spool file transfer services, such as RSCS, can be used to move spool files to the member where the intended instance of the multiconfiguration virtual machine resides.

If a SPOOLFILE statement is included in the identity entry of the multiconfiguration virtual machine definition or any included profile entry, and no SPOOLFILE statements are included in the subconfiguration entries, the MAXSPOOL value is the number of spool files that user ID can own on each member of the cluster where the user ID can log on. The maximum value on each member is 9999, which is the default. On each member the spool ID numbers begin at 1 and are incremented by 1 for each new file, up to MAXSPOOL. A SPOOLFILE statement can be included in a subconfiguration entry to set the spool file limit for the virtual machine instance on a particular member of the cluster. This overrides the default value or the value set by a SPOOLFILE statement in the identify entry or a profile entry, but only for that member.

Chapter 31. Preparing for Guest Relocations in a z/VM SSI Cluster

In a z/VM SSI cluster, a running guest virtual machine can be moved from one member to another. This process is called *live guest relocation*. The functions for initiating and managing guest relocations are provided by the CP VMRELOCATE command.

To prepare for live guest relocation, you need to ensure that the virtual machine has a relocatable configuration and that a matching configuration can be set up on the destination system. There are configuration attributes the guest must not have, because they cannot be relocated, and there are also characteristics the destination system must have in order to provide an identical virtual machine configuration. The relocation will fail if the guest has a configuration attribute that the relocation process does not support or if the destination system cannot satisfy the guest's configuration requirements.

To help you determine a guest's eligibility for relocation, you can use the TEST option of the VMRELOCATE command. Messages will be issued to identify conditions that need to be resolved. Eliminating or correcting conditions that will prevent a relocation will reduce the number of messages issued by the command.

Note: An eligibility confirmation from VMRELOCATE TEST applies only to the specific point in time when the command was executed and therefore does not guarantee eligibility when you attempt the actual relocation.

You can define relocation domains in which guests can be relocated freely between machines of different models or with different firmware or features. In special circumstances you can force a relocation to be attempted, even if certain conditions exist that would ordinarily prevent it.

Supported Configuration for Relocation

A guest can be relocated if it conforms to the following configuration:

Note: Linux is currently the only guest environment supported for relocation.

- The guest is running in an ESA, XA, or Z virtual machine in ESA/390 or z/Architecture mode.
- The guest is running with a disconnected virtual console.
- The virtual machine is defined as a single-configuration virtual machine; that is, the virtual machine definition begins with a USER statement.
- The guest's virtual processors are either all type CP or all type IFL.
- The guest was IPLed from a device, or IPLed from a named saved system (NSS) that meets the NSS criteria identified in the next item.
- The guest can use a discontiguous saved segment (DCSS) or NSS as long as the DCSS or NSS contains no SW (shared write), SC (shared with CP), or SN (shared with no data) page ranges, and the identical DCSS or NSS is available on the destination system.
- The guest can have any of the following:
  - Mindisks.
  - Dedicated devices, including OSA and FCP devices, provided that equivalent device access is available on the destination system.

    Note: To achieve equivalent device access when device numbers are not uniform across systems, use the following:

    ```
    COMMAND ATTACH EQID eqid TO * AS vdev
    ```

    Using `DEDICATE vdev rdev` in the guest's user directory entry doesn't work if *rdev* numbers are different on different members of the cluster.

- Private virtual disks in storage (created with the DEFINE VFB-512 command).
- Virtual unit record devices with no open spool files other than open console files. If the guest has an open console file, the relocation process will close the file on the source system and open a new one on the destination system.
- Virtual network interface cards (NICs) coupled to active virtual switches.

- The guest can be assigned to a CPU pool, as long as a compatible CPU pool exists on the destination system. For more information, see "Using CPU Pools in an SSI Cluster" on page 729. This restriction cannot be overridden, but you could remove the guest from the CPU pool before relocation.
- The guest can be using any of the following facilities:
  - Cryptographic adapter, when enabled by a CRYPTO APVIRTUAL statement in the virtual machine definition, as long as the shared pool resources have the same capability on the source and destination systems. The adapter types and AP and domain numbers can be different, but the AP modes must be the same. See "Live Guest Relocation of APVIRT Virtual Machines" on page 37.
  - High Performance FICON for IBM Z (zHPF), if this I/O architecture is supported by all processors, channel paths, and CP software levels on every member of the guest's relocation domain, as well as the storage controller.
  - Virtual machine time bomb (DIAGNOSE code X'288').
  - IUCV connections to the following CP system services:
    - *MSG
    - *MSGALL
  - Application monitor record collection (DIAGNOSE code X'DC', authorized by the APPLMON option in the virtual machine definition), if the guest buffer is not in a shared saved segment.
  - Single Console Image Facility (SCIF).
  - Collaborative Memory Management Assist (CMMA).

Attention:

- A Linux guest to be relocated that uses FCP devices must have multipathing support enabled with the **`queue_if_no_path`** option.
- To avoid possible data loss or file corruption, a relocation should not be performed when the Linux guest is actively using SCSI tapes accessed via FCP.
- Do not relocate a Linux guest that is being monitored by the Virtual Machine Resource Manager (VMRM). System and guest performance results are unpredictable and very likely to be unsatisfactory.

If the guest has any other characteristics (such as IUCV connections to other system services or other users), then it cannot be relocated. Configuration attributes not supported for relocation are identified in "Conditions That Will Prevent a Relocation" on page 757.

Using Relocation Domains

A relocation domain defines a set of members of an SSI cluster among which virtual machines can relocate freely. A domain can be used to define the subset of members of an SSI cluster to which a particular guest can be relocated. Relocation domains can be defined for business or technical reasons. For example, a domain can be defined that has all of the architectural facilities necessary for a particular application, or a domain can be defined to allow access only to systems with a particular software tool. Whatever the reason for the definition of a domain, CP will allow relocation among the members of the domain without any change to architectural characteristics or CP functionality as seen by the guest.

The relocation domain for a virtual machine can be defined with the VMRELOCATE directory statement. When the user ID logs on, CP assigns a *virtual architecture level* to the virtual machine that is the maximal common subset of the architectural features (hardware architecture facilities and CP-supplied features) of all the members of the SSI cluster that belong to that relocation domain. The guest cannot use architectural features that are not included in this virtual architecture level. This ensures that the guest can be freely relocated to other members of the domain because they provide the same architectural features. Note that a feature must be supported in every relevant component (processor, channel, and device hardware, as well as the z/VM software level) on every domain member in order to be usable to the guest.

Note: *Freely relocated* means relocated without regard to differences in machine models, firmware, or features. The guest must still conform to the other configuration requirements identified in "Conditions That Will Prevent a Relocation" on page 757.

To determine the virtual architecture level provided in the domain, CP gathers information on the instruction set, features, and software support available on each domain member. If a member is not currently joined, then CP uses the record of that member's capabilities when it was last joined. This ensures that in the most likely case, when that member is later IPLed with the same or a higher level of capability, the guests assigned to the domain will be able relocate there. If instead the member is IPLed on less capable hardware or at a lower CP software level, then the capabilities of the domain are adjusted downward. Guests that log on or re-IPL thereafter will be limited to the new maximal common subset of architected features, so that they can relocate throughout the domain. Guests that are already running when the domain is downgraded will keep the function level they have until the next guest IPL, but in the meantime, the deficient member will be excluded from the set of eligible relocation targets for that guest.

Relocation domains can be defined with the RELOCATION_DOMAIN system configuration statement or dynamically using the DEFINE RELODOMAIN command.

Two types of relocation domains are defined implicitly:

- A domain that includes all of the members of the SSI cluster. The name of this domain is SSI.
- A domain that includes one member of the SSI cluster. A single-member domain is defined for each member. The name of the domain is the member's system name.

When a system defined as a member of an SSI cluster is IPLed and joins the cluster, any relocation domains defined in the system configuration file for the joining member are compared with the domains defined for the existing members. If they do not match exactly, the differences are reconciled as follows:

- If the existing members and the joining member have the same domain but the lists of members in the domain do not match, the definition of the domain in use by the existing members is used for the joining member.
- If the joining member has a new domain definition that does not currently exist in the set of domains defined in the SSI cluster, the new domain is added and is propagated to the other members.

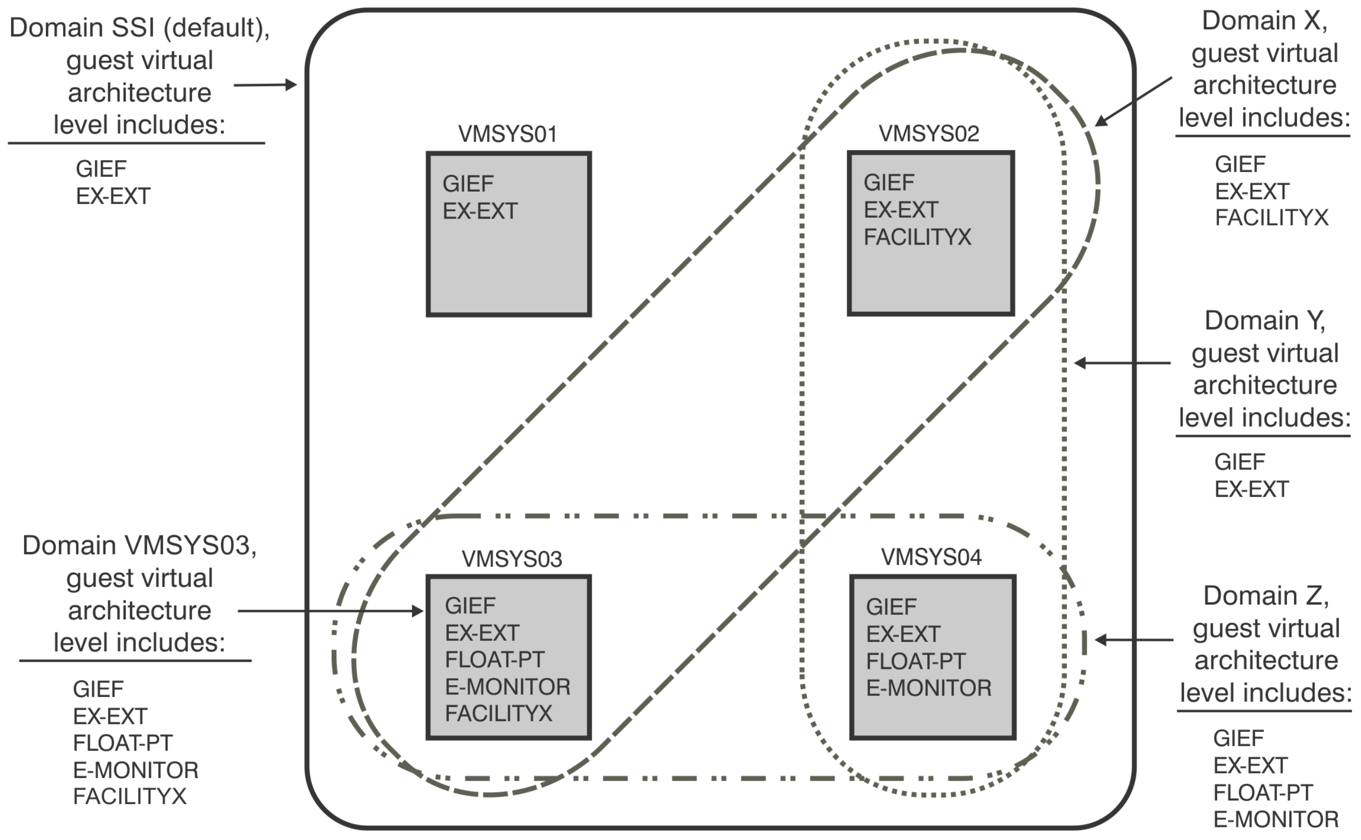
When a user ID defined by a single-configuration virtual machine definition logs on, the default associated relocation domain is the entire SSI cluster (domain SSI), unless a different relocation domain has been set by a VMRELOCATE statement in the user's virtual machine definition. If relocation of the virtual machine has been disabled on the VMRELOCATE statement and no relocation domain has been specified, the default relocation domain is the single-member domain of the system where the user logs on, and the user is assigned a virtual architecture level that is the set of all the architectural features of that system.

A virtual machine can be dynamically reassigned to a different relocation domain (or relocation can be enabled or disabled) with the SET VMRELOCATE command, but only if the new domain includes all of the architectural features in the current virtual architecture level (unless the FORCE ARCHITECTURE option is used).

Note: A virtual machine defined by a multiconfiguration virtual machine definition cannot be relocated. Therefore a relocation domain cannot be set for it with the VMRELOCATE statement or the SET VMRELOCATE command.

To determine the current relocation domain setting for a virtual machine logged on to the local member of the SSI cluster, use the QUERY VMRELOCATE command. (To determine the relocation domain setting for a virtual machine logged on to a remote member of the cluster, you can use the AT command to issue the QUERY VMRELOCATE command.) To find out what members of the SSI cluster are included in a particular relocation domain, use the QUERY RELODOMAIN command. The QUERY VMRELOCATE command will also identify any domain members to which the guest is temporarily excluded from relocating because they lack the capabilities the guest depends on (for example, because the hardware or software level of the member has been downgraded, or the member was added to the relocation domain since this guest last IPLed).

Figure 50 on page 756 shows an SSI cluster that includes members VMSYS01, VMSYS02, VMSYS03, and VMSYS04. The members are located on different servers (VMSYS03 and VMSYS04 are on a newer machine type), which provide different sets of architectural facilities. In addition to the implicitly defined cluster-wide relocation domain (SSI) and single-member relocation domains (VMSYS01, VMSYS02, VMSYS03, and VMSYS04), three relocation domains have been explicitly defined. Domain X includes members VMSYS02 and VMSYS03, domain Y includes members VMSYS02 and VMSYS04, and domain Z includes members VMSYS03 and VMSYS04.


Domain SSI (default), guest virtual architecture level includes:
GIEF
EX-EXT
VMSYS01
GIEF
EX-EXT
VMSYS02
GIEF
EX-EXT
FACILITYX
Domain X, guest virtual architecture level includes:
GIEF
EX-EXT
FACILITYX
Domain Y, guest virtual architecture level includes:
GIEF
EX-EXT
Domain VMSYS03, guest virtual architecture level includes:
GIEF
EX-EXT
FLOAT-PT
E-MONITOR
FACILITYX
VMSYS03
GIEF
EX-EXT
FLOAT-PT
E-MONITOR
FACILITYX
VMSYS04
GIEF
EX-EXT
FLOAT-PT
E-MONITOR
Domain Z, guest virtual architecture level includes:
GIEF
EX-EXT
FLOAT-PT
E-MONITOR

Architectural facilities used in this example:

GIEF	General-instructions-extension facility
EX-EXT	Execute-extensions facility
FLOAT-PT	Floating-point extension facility
E-MONITOR	Enhanced-monitor facility
FACILITYX	Possible future facility

Figure 50. Example of Relocation Domains and Guest Virtual Architecture Levels in an SSI Cluster

The following scenarios apply to the environment shown in Figure 50 on page 756:

- If a single-configuration virtual machine called USER1 is assigned to relocation domain X and logs on to member VMSYS03, the virtual architecture level that CP assigns to USER1 includes the general-instructions-extension facility, the execute-extensions facility, and the FACILITYX facility. USER1 can be freely relocated to member VMSYS02 because VMSYS02 is in the same relocation domain and provides the same architectural features.
- Although the floating-point extension facility is available on the server where member VMSYS03 is located, USER1 cannot use that facility because it is not included in USER1's virtual architecture level. To use the floating-point extension facility on member VMSYS03, USER1 must be reassigned to either

domain Z or domain VMSYS03. However, if reassigned to domain Z, USER1 will no longer be able to use the FACILITYX facility.

- If a single-configuration virtual machine called USER2 is not assigned to any relocation domain and logs on to any member of the cluster, for example VMSYS03, the default associated relocation domain is domain SSI. The virtual architecture level that CP assigns to USER2 includes the general-instructions-extension facility and the execute-extensions facility. Because this virtual architecture level is supported on every member of the SSI cluster, USER2 can be freely relocated to any other member.
- If a single-configuration virtual machine called USER3 is assigned to relocation domain X and logs on to member VMSYS04, which is not included in domain X, CP assigns a virtual architecture level that includes the facilities that are common between domain X and member VMSYS04, which are the general-instructions-extension facility and the execute-extensions facility.
- If a multiconfiguration virtual machine called USER4 logs on to members VMSYS02 and VMSYS04, the associated relocation domain for the instance on member VMSYS02 is domain VMSYS02, and the associated relocation domain for the instance on member VMSYS04 is domain VMSYS04. The virtual architecture level for USER4 on VMSYS02 includes the general-instructions-extension facility, the execute-extensions facility, and the FACILITYX facility. The virtual architecture level for USER4 on VMSYS04 includes the general-instructions-extension facility, the execute-extensions facility, the floating-point extension facility, and the enhanced-monitor facility.

Forcing a Relocation

In an emergency situation it might be necessary to ignore the results of certain parts of relocation eligibility checking and force a relocation to occur. This can be done by specifying the FORCE option on the VMRELOCATE MOVE command.

FORCE ARCHITECTURE
Attempts the relocation even though the virtual machine is currently running at a virtual architecture level that includes hardware architectural facilities or CP-supplied features not available on the destination system.

FORCE DOMAIN
Attempts the relocation even though the virtual machine would be moved outside of its domain.

FORCE STORAGE
Overrides the following memory capacity assessment checks:

- Guest storage size exceeds auxiliary paging capacity on the destination.
- Guest maximum storage size exceeds available space on the destination.
- Guest maximum storage size exceeds auxiliary paging capacity on the destination.

The FORCE STORAGE option will not override the check for guest current storage size exceeding available space on the destination. For more information, see “Resource Limit Conditions” on page 760, or see the VMRELOCATE command in *z/VM: CP Commands and Utilities Reference*.

Attention: The FORCE option is to be used only in an emergency situation and does not guarantee that the guest can run on the new system. The relocated guest might fail on the new system because a facility in use by the guest was removed by the forced relocation.

Conditions That Will Prevent a Relocation

The live guest relocation process will fail if any of these conditions are encountered.

- “Guest State Conditions” on page 758
- “Device Conditions” on page 758
- “Device State Conditions” on page 759
- “Virtual Facility Conditions” on page 759
- “Configuration Conditions” on page 760
- “Resource Limit Conditions” on page 760

- “Other Conditions” on page 761

Guest State Conditions

The relocation will fail if any of these guest state conditions exist:

- The user ID is still logging on or is being autologged.
- The user ID is logging off or is being forced off the system.
- The guest is terminating; that is, CP is waiting for the guest to respond to a shutdown signal.
- The guest is IPLing. (You must verify that the guest has completed the IPL, as the system will not detect it.)
- A system reset of the guest is in progress.
- The guest is the primary system operator.
- The guest is defined as a multiconfiguration virtual machine; that is, the user's virtual machine definition begins with an IDENTITY statement.
- The guest has an architecture incompatibility as determined by the guest's relocation domain and the architectural capabilities of the source and destination systems.
- The SET VMRELOCATE OFF command has been issued for this guest.
- The guest is an XC mode virtual machine.
- A relocation of this guest is already in progress.

Device Conditions

The relocation will fail if any of these device conditions exist:

- The guest has a virtual 3270 device or session (device type GRAF) as a virtual console or has a dialed or attached device.
- The ASCII console is attached to the guest.
- The guest has a temporary disk (T-disk) attached.
- The guest has a shareable virtual disk in storage attached (created by an MDISK directory statement with the V-DISK operand).
- The guest configuration includes an unsupported device.
- The guest configuration includes a logically connected virtual communication line (device type LINE).
- The guest configuration includes a virtual channel-to-channel adapter (CTCA).
- The guest configuration includes a dedicated dynamic switching device.
- The guest has a link to a local minidisk or a minidisk that cannot be linked on the destination system.

  Local minidisks include minidisks owned by local instances of multiconfiguration virtual machines (such as MAINT 190, 193, 19D, and 19E).
- The guest has a link to a minidisk created with the DEFINE MDISK command.
- The guest has a link to a minidisk currently using virtual working allegiance simulation (set with the SET WRKALLEG ON command).
- The guest has a link to a non-fullpack minidisk authorized for virtual reserve/release support (defined with access mode suffix V on the MDISK directory statement).
- The guest has a virtual DASD associated with a real device currently quiesced for HyperSwap®®.
- The guest has a virtual FBA DASD associated with a real device for which a reserve might still be held.
- The guest configuration includes a real device for which an associated equivalency ID (EQID) does not exist. (For more information about EQIDs, see “Real Device Management” on page 727.)
- The guest configuration includes a real FCP, OSA, IQD (HiperSockets), or CTC device for which the destination system has no free device with the same EQID.

- The guest configuration includes an emulated FBA DASD for which an associated EQID has not been created.
- The guest configuration includes a device for which the associated device on the destination system is of a different device class or type.
- The guest has a simulated network device (network interface card, NIC) that does not meet the networking eligibility requirements. For example, the guest has a NIC that is coupled to a virtual switch without external connectivity or that does not exist on the destination system. For more information about the relocation requirements for networking devices, see Live Guest Relocation Networking Considerations in *z/VM: Connectivity*.
- The guest is not using the Single Path CHPID Virtualization facility; that is, the CHPIDVIRTUALIZATION option is not specified in the user's virtual machine definition or on the GLOBALOPTS directory statement.
- The guest configuration includes PCI function(s).
- The guest configuration includes an EAV minidisk defined above cylinder 65520, but the destination system does have EAV support installed.

Device State Conditions

The relocation will fail if any of these device state conditions exist:

- The guest has a device for which the matching device on the destination system is offline.
- The guest has a dedicated device for which the matching device on the destination system is not free.
- The guest has a non-full-pack minidisk on a device for which the matching real device on the destination system is not attached to the system.
- The guest has a DEVNO full-pack minidisk (that is, defined by an MDISK statement with the DEVNO operand) on a device for which the matching real device on the destination system is not:
  - Already defined as a DEVNO full-pack minidisk
  - Free such that a DEVNO full-pack minidisk can be created on it
- The guest has a Parallel Access Volumes (PAV) device for which the matching real device on the destination system is not in the same PAV mode.
- The guest has active I/O to a device that will result in a delayed response (for example, certain Peer-to-Peer Remote Copy (PPRC) operations for DASD or mount requests for an Automated Tape Library (ATL)).
- The guest configuration includes a TAPE device and a real assign exists for the device or the guest owns the assign.
- The guest configuration includes a multiuser TAPE device and the equivalent device on the destination system is neither defined as multiuser nor free.
- The guest configuration includes a TAPE device defined with an encryption key and the equivalent device on the destination system is associated with a different encryption key.
- The guest has an open SPOOL file (other than a console file). This includes a VMDUMP operation in continuous output mode (CONT option).

Virtual Facility Conditions

The relocation will fail if any of these virtual facility conditions exist:

- The guest is a Coupling Facility Service Machine; that is, the CFVM option is specified in the user's virtual machine definition.
- The guest configuration includes access to a Coupling Facility Service Machine (that is, the CFUSER option is specified in the user's virtual machine definition), or the guest has a device that is coupled to a Coupling Facility Service Machine.
- The guest has expanded storage attached.
- The guest is using VM data spaces (by using the ALSERV ADD macro and DIAGNOSE code X'248').

- The guest has an SPXTAPE operation in progress.
- This user ID has active CP traces (activated with the TRACE command).
- The guest is the subject of an enabled TRSOURCE trap of type GT.
- The guest is using DIAGNOSE code X'E0' to read from or write data to a system trace file.
- The guest has created a logical device or is using a logical device created by another virtual machine or the system.
- The guest is a worker machine; that is, the guest is using the Set Alternate User ID function (DIAGNOSE code X'D4').
- The guest is using the Virtual Machine Communication Facility (VMCF); that is, the guest has invoked the VMCF AUTHORIZE function with DIAGNOSE code X'68'.
- The guest is using CMS POSIX facilities.
- The guest has used DIAGNOSE code X'214' to establish a pending page release.
- The guest has directory authorization to use DIAGNOSE code X'98'; that is, the DIAG98 option is specified in the user's virtual machine definition.
- The guest has an IUCV connection to a system service (other than to *MSG or *MSGALL, which are allowed) or to another virtual machine.
- The guest has an APPC/VM connection.
- The guest has directory authorization to use DIAGNOSE code X'254' to issue tape library control commands; that is, the user's virtual machine definition includes the STDEVOPT statement with the LIBRARY CTL operands.
- The guest has directory authorization to use the virtual Multi-Path Lock Facility; that is, the LKFAC option is specified in the user's virtual machine definition.

Configuration Conditions

The relocation will fail if any of these configuration conditions exist:

- The guest has dedicated cryptographic capability. Dedicated cryptographic capability occurs when APDEDICATED is specified on a CRYPTO statement in the user's virtual machine definition or when a dedicated cryptographic devices is attached by using the ATTACH CRYPTO command.
- The guest is using any of the following:
  - An NSS defined with the VMGROUP option
  - A DCSS that is a member of a segment space
  - An NSS or DCSS with a storage range of type SW, SC, or SN
- The guest is using an NSS or DCSS that does not exist on the destination system or that does not have matching attributes and content.
- The guest configuration includes more than one virtual processor type; this precludes the relocation of any guest with a virtual ZIP or ZAP.
- The guest is defined with all type CP virtual processors and the destination is an all type IFL environment.
- The guest's virtual configuration mode (defined by the SET VCONFIG command) is not LINUX and the mode of the destination LPAR is LINUX only.
- The guest has a storage configuration defined with the DEFINE STORAGE CONFIGURATION command.
- The guest has 370ACCOM mode set on.
- The guest is in a CPU pool and a compatible pool does not exist on the destination system.

Resource Limit Conditions

The relocation will fail if any of these resource limit conditions exist:

- Guest storage size will not fit into available space on the destination.

- The guest's virtual disk in storage usage would violate the usage restrictions on the destination system.
- Relocating the guest would cause the destination system to exceed the maximum number of allowed users.

The relocation might fail if any of these resource limit conditions exist, but these checks may be overridden by using the FORCE STORAGE operand on the VMRELOCATE command:

- Guest storage size exceeds auxiliary paging capacity on the destination.
- Guest maximum storage size exceeds available space on the destination.
- Guest maximum storage size exceeds auxiliary paging capacity on the destination.

Guest storage size is defined as the sum of current guest storage (assumed to be fully populated), CP supporting structures, and private virtual disks in storage. Guest maximum storage size is defined as the sum of current guest storage, CP supporting structures, private virtual disks in storage, standby storage, and reserved storage. Available space on the destination is defined as the sum of available central, expanded, and auxiliary storage.

Note: Expanded storage is not supported by z/VM 6.4 and later.

For additional information concerning resource limit conditions that affect relocation for a Linux guest virtual machine, see z/VM Single System Image Overview (https://www.ibm.com/vm/ssi).

Other Conditions

The relocation will fail if any of these conditions exist:

- A customized CP control block field appears to be in use. Fields in the VMDBK control block (VMDUSER1 - VMDUSER8 and VMDEBUG1 - VMDEBUG8) and the VDEV control block (VDEVUSR1 - VDEVUSR4) are reserved for customer and vendor use, and at least one of those fields contains something other than its initial value.
- The user ID is in system hold state on the destination system.
- The user ID was logged on to the source system by LOGON BY and the issuing user ID is in system hold state on the destination system.
- The guest is in the process of a service processor operation, such as writing the input/output configuration data set (IOCDS).

Chapter 32. Converting a z/VM System to a Single-Member z/VM SSI Cluster

Use this procedure to convert a non-SSI z/VM system to the first member of a z/VM SSI cluster. This is the first step in moving to a multimember SSI cluster. To create a multimember SSI cluster, complete this procedure and then follow the procedure in Chapter 34, "Adding a Member to a z/VM SSI Cluster by Cloning an Existing Member," on page 785.

See "Before You Begin the Conversion Procedure" on page 763 for requirements, suggested practices, and preparations.

In this procedure you will complete the following tasks:

"Task 1: Prepare the Member-Specific User Volume" on page 765
"Task 2: Update the System Configuration File" on page 765
"Task 3: Restructure the User Directory" on page 768
"Task 4: Manage the User Spool Files" on page 774
"Task 5: Prepare the CP-Owned Volumes" on page 774
"Task 6: Modify the Startup Parameters for the VMPSFS File Pool" on page 775
"Task 7: Shut Down and Warm Start" on page 775
"Task 8: Load the Spool Files" on page 776
"Task 9: Change the User Directory to SSI-Enabled" on page 776
"Task 10: Clean Up" on page 777

Before You Begin the Conversion Procedure

Attention
The conversion procedure will change your system configuration. You need to plan these tasks, and you might want to use a maintenance window to execute certain steps. Before you begin the procedure, make sure that you understand the conversion process and its implications to your system. Do not use this procedure if the system to be converted was installed with SMAPI enabled.

Make sure that you meet all of the requirements and complete the necessary preparations before you start. The examples in this procedure assume that you are following the suggested practices.

Requirements:

- Make sure that you understand how a z/VM SSI cluster is set up and maintained. See Chapter 29, "Setting Up z/VM Single System Image Clusters," on page 715.
- Adhere to the "SSI Cluster Requirements" on page 728 and "SSI Cluster Restrictions" on page 729.
- The system to be converted must be z/VM 6.4 or later, installed by following one of the non-SSI installation procedures documented in *z/VM: Installation Guide*. The system must be installed to ECKD DASD.
- If the system to be converted was migrated from a previous z/VM release, all migration tasks must be completed before starting this procedure. For example, the customer-specific user information from the old system must be merged into the user directory on the new system.

  ⚠ **Attention:** The source directory created by the non-SSI installation procedure is in SSI-ready format, which means the IBM-supplied virtual machine definitions already have the correct structure for an SSI cluster. Do not attempt to use a source directory that is not in SSI-ready format, and do not modify the IBM-supplied virtual machine definitions except as instructed in this procedure.

- If you are using RACF, the RACF database must be configured to be shared by the members of the SSI cluster, and the shared database should be set up before you begin this conversion procedure. See the information on sharing RACF databases in an SSI cluster in *z/VM: RACF Security Server System Programmer's Guide*.

Suggested practices:

- See "Suggested Practices for Setting Up an SSI Cluster" on page 729.

Preparations:

- Gather the information you need to define the SSI cluster. The following example values are used in this procedure:
  - Logical partition (LPAR) name = LP01

    Note: If z/VM is installed second level, this is the user ID of the virtual machine in which the system is running.
  - Cluster name = CLUSTERA
  - System name = VMSYS01

    Attention: The system name of a member of an SSI cluster may include only alphabetic and numeric characters. If the current system name does not conform to this requirement, change the name before starting this procedure.
  - Real device numbers and labels for all CP-owned and user DASD volumes.

    The examples in this procedure use the default volume labels created by the installation procedure. For most volumes the naming convention is M0*mvnn*, where *m* is the number of this member in the SSI member list (on the SSI system configuration statement) and *vnn* identifies the volume type and number (for example, P01 for the first paging volume).
    - Common CP-owned volume = VMCOM1
      - Created by the installation process
      - Contains the parm disk, which contains the system configuration file that will be common for the SSI cluster
      - Contains the source directory that will be common for the SSI cluster
      - Contains the cross-release utilities disk, which contains utilities such as CPFMTXA, DIRECTXA, DIRMAP, DISKMAP, and FORMSSI
      - Contains the VMPSFS file pool, which will be shared by all members of the SSI cluster
      - Includes the area where the SSI persistent data record (PDR) should be written
    - Other CP-owned volumes:
      - System residence volume = M01RES
      - Paging volume = M01P01
      - Spool volume = M01S01
    - User volumes:
      - Common release-level service volumes = *vrm*RL1, *vrm*RL2
      - Shared user volume = USRVL1

        This is a customer-defined volume used for minidisks for customer-defined single-configuration virtual machines.
      - Member-specific user volume = M01PV1

        This is a customer-defined volume used for VMSYS01-specific minidisks for customer-defined multiconfiguration virtual machines.
- If necessary, notify your users to clean up their spool files. Although a user in an SSI cluster can own up to the number of spool files defined by the MAXSPOOL operand on the SPOOLFILE directory statement

(9999 by default), only one quarter of the MAXSPOOL number of files may be originated on any one particular member of the cluster. You might want to set a lower limit for this system, such as 2000, and tell your users that spool files over the 2000 limit will be purged (over the weekend, for example).

Task 1: Prepare the Member-Specific User Volume

Prepare the customer-defined member-specific user volume.

Note: For simplicity, this procedure does not show all of the system responses.

1. Log on to the MAINT*vrm* user ID for the release level of VMSYS01. Ensure that you have access to the cross release utilities disk (551).
2. Attach the volume to your virtual machine:

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

3. Format, label, and allocate the volume:

```
cpfmtxa vdev m01pv1

CPFMTXA:
FORMAT WILL ERASE CYLINDERS 00000-nnnnn ON DISK vdev
DO YOU WANT TO CONTINUE? (YES | NO)

yes

HCPCCF6209I INVOKING ICKDSF.
⋮
ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
ENTER ALLOCATION DATA
TYPE CYLINDERS

perm 0-end
end

HCPCCF6209I INVOKING ICKDSF.
⋮
ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Task 2: Update the System Configuration File

Prepare a revised system configuration file that will be common for the SSI cluster.

1. Access the PMAINT CF0 minidisk:

```
access cf0 fm
```

2. Make a copy of the SYSTEM CONFIG file named TEMP CONFIG on the CF0 minidisk and edit the copy.
3. Update the SYSTEM_IDENTIFIER information.
   a. If the file contains any SYSTEM_IDENTIFIER_DEFAULT statements, remove them.
   b. Include a SYSTEM_IDENTIFIER statement for this member of the cluster, to define the unique identifier (system name) for the z/VM system to be run in the specified LPAR.

      Note: If z/VM is installed second level, the LPAR name field specifies the user ID of the virtual machine in which this system is running.

      The system name is used as a record qualifier parameter on other configuration statements that apply only to that member of the cluster. At system initialization, CP locates the SYSTEM_IDENTIFIER statement with that LPAR name and uses the system name to select the configuration statements to be processed for that member — all statements qualified with that identifier and all statements with no qualifier.

      You should put the SYSTEM_IDENTIFIER statement at the top of the file, following any EQUATE statements.

```
/**********************************************************************/
/*                    System_Identifier Information                   */
/**********************************************************************/

   System_Identifier LPAR LP01 VMSYS01
```

4. Add an SSI statement to specify that this system configuration file defines an SSI cluster.

   The SSI statement specifies the name of the cluster, the label of the DASD volume that contains the SSI persistent data record (PDR), and the system names of the members of the cluster.

   ```
   /**********************************************************************/
   /*         SSI Statement                                              */
   /**********************************************************************/

     SSI CLUSTERA PDR_Volume VMCOM1 ,
           Slot 1 VMSYS01
   ```

   Note: A member of an SSI cluster is sometimes referred to by its slot number on this statement. That is, in this example system VMSYS01 is member 1 of cluster CLUSTERA.

5. Each member of the SSI cluster must have its own SYSTEM_RESIDENCE statement, identifying the checkpoint and warm start areas for that member. Qualify the current SYSTEM_RESIDENCE statement for member VMSYS01.

   ```
   /**********************************************************************/
   /*                Checkpoint and Warmstart Information                */
   /**********************************************************************/

     VMSYS01: System_Residence,
       Checkpoint  Volid M01RES    From CYL 21  For 9 ,
       Warmstart   Volid M01RES    From CYL 30  For 9
   ```

6. Organize the CP_OWNED statements. Qualify the statements for volumes that are specific to member VMSYS01. Do not qualify the statements for volumes that will be shared with the other members.

   Qualify:

   - System residence volume (M01RES)
   - System paging volume (M01P01)

   Do not qualify:

   - Common volume (VMCOM1)
   - Spool volume (M01S01)

   Important: Do not change your current spool volume slot numbers.

```
/**********************************************************************/
/*                     CP_Owned Volume Statements                     */
/**********************************************************************/
/*                                                  RES   VOLUME      */
/**********************************************************************/

    VMSYS01: CP_Owned   Slot   1  M01RES

/**********************************************************************/
/*                                                 COMMON VOLUME      */
/**********************************************************************/

    CP_Owned   Slot   5  VMCOM1

/**********************************************************************/
/*                                           DUMP & SPOOL VOLUMES     */
/**********************************************************************/

    CP_Owned   Slot  10  M01S01

/**********************************************************************/
/*                                           PAGE & TDISK VOLUMES     */
/**********************************************************************/

        VMSYS01: CP_Owned   Slot 255  M01P01
```

7. Organize the USER_VOLUME_LIST statements. Qualify the statements for volumes that are specific to member VMSYS01. Do not qualify the statements for volumes that will be shared with the other members.

   Qualify:

   - Member-specific user volume (M01PV1)

   Do not qualify:

   - Release-level service volumes (*vrm*RL1 and *vrm*RL2)
   - Shared user volume (USRVL1)

   Before:

```
/**********************************************************************/
/*                          User_Volume_List                          */
/**********************************************************************/


   User_Volume_List vrmRL1 vrmRL2 USRVL1
```

   After:

```
/**********************************************************************/
/*                          User_Volume_List                          */
/**********************************************************************/
/* Shared user volumes                             */
/**********************************************************************/

   User_Volume_List vrmRL1 vrmRL2 USRVL1

/**********************************************************************/
/* Member-specific user volumes for Member 1                          */
/**********************************************************************/

   VMSYS01: User_Volume_List M01PV1
```

8. If the file includes a VMLAN statement with the MACIDRANGE operand, that operand is ignored in an SSI cluster and the operand should be removed.

In an SSI cluster, system-defined locally administered MAC addresses are created using the prefix value specified on the MACPREFIX operand. The MACPREFIX value must be different for each member of the cluster. The default value is 02*xxxx*, where *xxxx* is the member's slot number on the SSI statement. If the MACPREFIX value is explicitly defined, the VMLAN statement must be qualified for the member to which it applies. Therefore if a VMLAN statement with the MACPREFIX operand is retained from the non-SSI system or created in this step, it must be qualified for member VMSYS01.

User-defined locally administered MAC addresses in an SSI cluster are created using the prefix value specified on the USERPREFIX operand. The USERPREFIX value must be identical for all members of the cluster, so the same MAC address will be created when the virtual machine is logged on to any member. The USERPREFIX value for the cluster cannot be the same as the MACPREFIX value for any member. The default USERPREFIX value in an SSI cluster is 020000.

User-defined MAC addresses from the non-SSI system will be retained in the SSI cluster if the USERPREFIX value for the cluster is the same as the USERPREFIX value in the non-SSI system. If the USERPREFIX value was not explicitly defined in the non-SSI system, the default was set to the MACPREFIX value. If the MACPREFIX value was not explicitly defined, the default value was 020000. Therefore, to retain the user-defined MAC addresses if the USERPREFIX value for the non-SSI system was set or defaulted to any value other than 020000, that value must be explicitly defined as the USERPREFIX value for the SSI cluster on a VMLAN statement for member 1.

```
VMSYS01: VMLAN MACPREFIX xxxxxx USERPREFIX yyyyyy
```

9. Add a FEATURES statement (below all other FEATURES statements) to cause the system to prompt for the type of start when processing a SHUTDOWN REIPL command. (You will remove this statement at the end of this procedure.)

```
FEATURES ENABLE PROMPT AFTER_SHUTDOWN_REIPL
```

10. After you finish editing the file, use the CPSYNTAX utility (available on the MAINT 193 minidisk) to check the syntax of the configuration statements for cluster member VMSYS01:

```
cpsyntax temp config * (lpar lp01
```

Note: If z/VM is installed second level, the LPAR name field specifies the user ID of the virtual machine in which this system is running. The value of the LPAR operand must match the value specified on the SYSTEM_IDENTIFIER statement that was added above.

Correct any errors before proceeding to the next task.

Task 3: Restructure the User Directory

An explanation with concepts and examples supports the task of restructuring the user directory. One set of instructions is provided for a system with DirMaint installed. Another set of instructions is provided for a system without DirMaint installed.

Explanation of the Changes

Many of the IBM-supplied user IDs are defined by multiconfiguration virtual machine definitions. A multiconfiguration virtual machine definition begins with an IDENTITY statement instead of a USER statement. When included in the common source directory for an SSI cluster, a multiconfiguration virtual machine definition can be configured to define multiple virtual machine instances, which enables the user ID to be logged on concurrently and independently to multiple members of the cluster. The virtual machine instances have common attributes but can also be configured to access different resources (for example, member-specific minidisks).

This type of virtual machine definition is used for many system support user IDs. The ability to be logged on to multiple members of the SSI cluster at the same time allows the user (such as the IBM-supplied MAINT user ID) to perform tasks on various members without having to log off one member before logging on to another. This type of virtual machine definition can also be used for service virtual machines (SVMs) and servers (such as the IBM-supplied TCPIP server) that require instances on multiple members of the cluster. Each instance supports only that member of the SSI cluster.

The source directory created by the non-SSI installation procedure is in SSI-ready format, which means the IBM-supplied multiconfiguration virtual machine definitions are configured for a single system. Each definition includes at most one active BUILD statement and its associated subconfiguration entry. The BUILD statement specifies an asterisk (*) instead of a system name to default to the IPLed system. The following example shows the SSI-ready format of the multiconfiguration virtual machine definition for the OPERATOR user ID.

Note: The statements that would be needed for additional SSI cluster members (except member-specific MDISK statements) are included as comments in the default directory.

```
IDENTITY OPERATOR password 32M 32M ABCDEFG
 INCLUDE IBMDFLT
 BUILD ON * USING SUBCONFIG OPERTR-1
* BUILD ON @@member2name USING SUBCONFIG OPERTR-2
* BUILD ON @@member3name USING SUBCONFIG OPERTR-3
* BUILD ON @@member4name USING SUBCONFIG OPERTR-4
  AUTOLOG AUTOLOG1 OP1 MAINT
  ACCOUNT 2 OPERATOR
  MACH ESA
  OPTION MAINTCCW

SUBCONFIG OPERTR-1
 LINK OP1   191 192 RR
 MDISK 191 3390 2924 005 MO1RES MR READ     WRITE    MULTIPLE

*SUBCONFIG OPERTR-2
* LINK OP1   191 192 RR

*SUBCONFIG OPERTR-3
* LINK OP1   191 192 RR

*SUBCONFIG OPERTR-4
* LINK OP1   191 192 RR
```

If you have customer-defined virtual machines that you want to be able to run concurrently on multiple members of the SSI cluster, restructure them as multiconfiguration virtual machine definitions.

For example, suppose you have created a virtual machine called WATCHER that issues periodic queries and probes to monitor the health of other virtual machines on your system. Depending on the nature of the tests this machine performs, you might want to have a separate instance of WATCHER on each member of your SSI cluster, responsible for watching the guests running on that member. In that case, you would want to set up WATCHER as a multiconfiguration virtual machine.

Before you convert the system to an SSI cluster, the virtual machine definition for WATCHER might look like this:

```
USER WATCHER password 64M 64M G
 IPL CMS
 LINK MAINT 190 190
 LINK MAINT 19E 19E
 MDISK 191 3390 200 100 USRVL1 MR
 MDISK 192 3390 300 100 USRVL1 RR
⋮
```

In preparation for deploying an SSI cluster, restructure WATCHER as a multiconfiguration virtual machine definition that clearly indicates which aspects of the definition are common across all instances of WATCHER and which are specific to the instance of WATCHER on a particular member of the SSI cluster.

This distinction is not particularly significant in a single-member SSI cluster, but it will facilitate the process of cloning that member to add other members to the cluster.

In this example, the WATCHER 192 minidisk contains the software run by WATCHER (execs, control files, and so on) and is read-only during normal execution. Therefore the 192 minidisk should be common across all instances of WATCHER. The WATCHER 191 minidisk is read/write, containing log files, work files, and similar items. Therefore the 191 minidisk should be specific to each instance of WATCHER.

The revised definition might look like this:

```
IDENTITY WATCHER password 64M 64M G
 BUILD ON * USING SUBCONFIG WATCHR-1
 IPL CMS
 LINK MAINT 190 190
 LINK MAINT 19E 19E
 MDISK 192 3390 300 100 USRVL1 RR
⋮

SUBCONFIG WATCHR-1
 MDISK 191 3390 100 100 M01PV1 MR
⋮
```

The WATCHER virtual machine is now defined in two parts. The common aspects of WATCHER are in the identity entry. The aspects that are specific to the instance of WATCHER on VMSYS01 are in subconfiguration entry WATCHR-1. The 191 minidisk has been moved to volume M01PV1. This is the volume that you prepared in "Task 1: Prepare the Member-Specific User Volume" on page 765. Note that in this example the 191 minidisk also has been moved from cylinders 200-299 on USRVL1 to cylinders 100-199 on M01PV1.

Restructure a User Directory By Using a Directory Manager

Restructure the user directory of your customer-defined virtual machines. You must complete this task when you convert a z/VM system to a single-member z/VM SSI cluster and you use a directory manager.

Before you begin

Attention: Make sure that the virtual machine is logged off before you restructure its definition. Verify that DirMaint is installed and enabled.

Procedure

1. Restructure your customer-defined virtual machines, as needed, in the manner described in the WATCHER example.
   a) Update the DirMaint configuration to define the DIRMSAT server and DATAMOVE server for member 1 of the SSI cluster.

      Create a file called CONFIGSS DATADVH (on the DIRMAINT 11F disk and the 7VMDIR20 41F disk) that includes a SATELLITE_SERVER statement and a DATAMOVE_MACHINE statement. For example:

      ```
      SATELLITE_SERVER= DIRMSAT VMSYS01
      DATAMOVE_MACHINE= DIRMSAT VMSYS01
      ```

      Note: If you are using an external security manager (ESM), such as RACF, the DIRMSAT and DATAMOVE servers also must be identified to the ESM. See the external security manager considerations in *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*.
   b) Complete the following steps for each virtual machine definition that you want to change.

Step	Task
i.	Temporarily suspend placing the updated source directory online:

Step Task

```
dirmaint offline
```

ii. Get a copy of the user entry to be changed:

```
dirmaint foruser watcher get nolock
```

iii. Convert the user entry to an identity entry and a subconfiguration entry.

iv. Put the subconfiguation entry into a separate file, using the SUBCONFIG ID as the file name. The file name of the file containing the identity entry remains unchanged, because the user ID is still the same. So for WATCHER, the files are WATCHER DIRECT and WATCHR-1 DIRECT.

v. Delete the user entry from the directory, but keep any existing links:

```
dirmaint foruser watcher purge keeplinks
```

vi. Add the identity entry to the directory:

```
dirmaint add watcher
```

vii. Add the associated subconfiguration entry to the directory:

```
dirmaint add watchr-1 build on vmsys01 in watcher
```

viii. Return to ii and repeat the steps for the next definition to be changed.

2. Use the DISKMAP utility to check the minidisk assignments:

   Run the DISKMAP utility:

   ```
   diskmap user direct fm
   ```

   The output file is sent to your A disk. If any overlapping minidisks are flagged, make the necessary adjustments in the source directory and run DISKMAP again.
3. Copy appropriate minidisks to the new volume. Appropriate minidisks are those minidisks whose definitions were changed from common user volumes to the VMSYS01-specific user volume (M01PV1) when the virtual machines were restructured.

   a) If the M01PV1 volume is not still attached to your virtual machine (from “Task 1: Prepare the Member-Specific User Volume” on page 765), attach it now.

   Use the ATTACH command:

   ```
   attach rdev to * as vdev1
   ```

   b) Link to a minidisk that you must copy.
   For example, link to the WATCHER 191 minidisk:

   ```
   link watcher 191 as vdev2
   ```

   c) Copy the minidisk to M01PV1.
   For example, copy the WATCHER 191 minidisk:

   ```
   flashcopy vdev2 0 end to vdev1 100 199 synchronous
   ```

   Notes:

   i. The SYNCHRONOUS operand is required only for FlashCopy Version 1 hardware.

   ii. You can use the DDR utility as an alternative to the FLASHCOPY command to copy the minidisks.

Notes:

iii. In this example, WATCHER 191 is moved to a different location on M01PV1 than on USRVL1.

4. Define full-pack minidisks for appropriate volumes so you can mark them with SSI ownership information.

   Appropriate volumes are any CP-owned volumes that you added after you installed this system. SSI ownership information is required preparation for "Task 5: Prepare the CP-Owned Volumes" on page 774.

   Full-pack minidisks for the installation volumes are defined in the user directory, as shown in the following example table. Define full-pack minidisks for any CP-owned volumes that you added after you installed this system.

   For example, the M01T01 volume for temporary minidisks that is listed in the table is not included in the initial installation. If you added this volume, the user directory includes a USER definition for the $TDISK$ user ID where you can add an MDISK statement for a full-pack minidisk.

   Marking ownership information on user volumes is optional. If you want to mark the user volumes, you need to define full-pack minidisks for those volumes as well.

Table 52. Full-Pack Minidisks that are Defined for the Installation Volumes

Volume	**Full-Pack Minidisk**
VMCOM1	PMAINT 141
M01S01	MAINT 122
M01RES	MAINT 123
M01P01	$PAGE$ A01
M01T01	$TDISK$ *vdev*
*vrm*RL1	MAINT*vrm* 131
*vrm*RL2	MAINT*vrm* 132

5. Create the new object directory for VMSYS01.

   Enter the following command:

```
directxa user direct  vfm
```

Restructure a User Directory Without Using a Directory Manager

Restructure the user directory of your customer-defined virtual machines. You must complete this task when you convert a z/VM system to a single-member z/VM SSI cluster and you do not use a directory manager.

Before you begin

Attention: Make sure that the virtual machine is logged off before you restructure its definition.

Procedure

1. Restructure your customer-defined virtual machines, as needed, in the manner described in the WATCHER example.
   a) Make a backup copy and then edit the USER DIRECT file.
   b) Change the selected USER definitions to IDENTITY definitions.
   c) Save your changes.

2. Use the DISKMAP utility to check the minidisk assignments:

   Run the DISKMAP utility:

   ```
   diskmap user direct fm
   ```

   The output file is sent to your A disk. If any overlapping minidisks are flagged, make the necessary adjustments in the source directory and run DISKMAP again.
3. Copy appropriate minidisks to the new volume. Appropriate minidisks are those minidisks whose definitions were changed from common user volumes to the VMSYS01-specific user volume (M01PV1) when the virtual machines were restructured.
   a) If the M01PV1 volume is not still attached to your virtual machine (from "Task 1: Prepare the Member-Specific User Volume" on page 765), attach it now.

   Use the ATTACH command:

   ```
   attach rdev to * as vdev1
   ```

   b) Link to a minidisk that you must copy.
   For example, link to the WATCHER 191 minidisk:

   ```
   link watcher 191 as vdev2
   ```

   c) Copy the minidisk to M01PV1.
   For example, copy the WATCHER 191 minidisk:

   ```
   flashcopy vdev2 0 end to vdev1 100 199 synchronous
   ```

   Notes:

   i. The SYNCHRONOUS operand is required only for FlashCopy Version 1 hardware.

   ii. You can use the DDR utility as an alternative to the FLASHCOPY command to copy the minidisks.

   iii. In this example, WATCHER 191 is moved to a different location on M01PV1 than on USRVL1.
4. Define full-pack minidisks for appropriate volumes so you can mark them with SSI ownership information.

   Appropriate volumes are any CP-owned volumes that you added after you installed this system. SSI ownership information is required preparation for "Task 5: Prepare the CP-Owned Volumes" on page 774.

   Full-pack minidisks for the installation volumes are defined in the user directory, as shown in the following example table. Define full-pack minidisks for any CP-owned volumes that you added after you installed this system.

   For example, the M01T01 volume for temporary minidisks that is listed in the table is not included in the initial installation. If you added this volume, the user directory includes a USER definition for the $TDISK$ user ID where you can add an MDISK statement for a full-pack minidisk.

   Marking ownership information on user volumes is optional. If you want to mark the user volumes, you need to define full-pack minidisks for those volumes as well.

Table 53. Full-Pack Minidisks that are Defined for the Installation Volumes

Volume	Full-Pack Minidisk
VMCOM1	PMAINT 141
M01S01	MAINT 122
M01RES	MAINT 123

Table 53. Full-Pack Minidisks that are Defined for the Installation Volumes (continued)

Volume	Full-Pack Minidisk
M01P01	$PAGE$ A01
M01T01	$TDISK$ *vdev*
*vrm*RL1	MAINT*vrm* 131
*vrm*RL2	MAINT*vrm* 132

5. Create the new object directory for VMSYS01.

   Enter the following command:

   ```
   directxa user direct  vfm
   ```

Task 4: Manage the User Spool Files

In an SSI cluster, a single-configuration virtual machine cannot own more than one quarter of the MAXSPOOL number of spool files originated on any one particular member of the cluster.

Note: A multiconfiguration virtual machine can have up to the MAXSPOOL number of spool files on each member of the cluster.

1. Verify that no single-configuration virtual machine has more than 2500 spool files. Read in or purge the excess.
2. Use the SPXTAPE command to dump all standard spool files and system data files to tape:

   ```
   vary online rdev
   attach rdev to * as vdev
   ⋮
   spxtape dump vdev spool
   ```

   Note: If you do not have a tape drive, either read in or delete the standard spool files (RDR, PRT, and PUN files only).

Task 5: Prepare the CP-Owned Volumes

You need to mark the CP-owned volumes with SSI ownership information (cluster name and system name of the owning member) and create the SSI persistent data record. Marking SSI ownership information on the user volumes is optional.

1. Link to the full-pack minidisk for a volume to be marked (see Table 52 on page 772). For example, to link the VMCOM1 volume:

   ```
   link pmaint 141 as vdev mr
   ```

2. Mark the volume with SSI ownership information:

   - Mark the common volume (VMCOM1) with the cluster name but no system name:

     ```
     cpfmtxa vdev vmcom1 owner clustera.nosys
     ```

   - Mark each member-specific CP-owned volume (system residence, paging, and spool) with the cluster name and the system name:

     ```
     cpfmtxa vdev m01res owner clustera.vmsys01
     cpfmtxa vdev m01p01 owner clustera.vmsys01
     cpfmtxa vdev m01s01 owner clustera.vmsys01
     ```

   - Optionally, you can record ownership information on the user volumes.

- If the volume is for member-specific minidisks, mark the volume with the SSI cluster name and the system name of the member. For example, to mark the M01PV1 volume:

  ```
  cpfmtxa vdev m01pv1 owner clustera.vmsys01
  ```

- If the volume is to be shared among the cluster members, mark the volume with the SSI cluster name and no system name. For example, to mark the USRVL1 volume:

  ```
  cpfmtxa vdev usrvl1 owner clustera.nosys
  ```

3. Create the SSI persistent data record (PDR):

   ```
   link pmaint 141 141 w
   ⋮
   formssi create 141 clustera
   ```

   Note:

   a. The PDR should reside on the VMCOM1 volume, and cylinder 0 must be allocated as PERM.

   b. In this example, PMAINT 141 is a full-pack overlay of the VMCOM1 volume.

Task 6: Modify the Startup Parameters for the VMPSFS File Pool

VMPSFS is the product service file pool for products loaded into SFS. VMPSFS is used by VMSES/E even if you did not load any products into SFS when you installed this z/VM system. The server for the VMPSFS file pool is VMSERVP. In the VMSERVP DMSPARMS file, the default startup parameter LOCAL must be changed to SSI to allow connections to VMSERVP from all members of the SSI cluster.

1. Log on or reconnect to the VMSERVP server.
2. Stop the file pool:

   ```
   stop
   ```

3. Edit the VMSERVP DMSPARMS file and change the LOCAL startup parameter to SSI:

   ```
   ADMIN MAINT MAINTvrm
   NOBACKUP
   SAVESEGID CMSFILES
   SSI
   FILEPOOLID VMPSFS
   USERS 100
   ```

Task 7: Shut Down and Warm Start

1. Prepare to use the revised system configuration file:

   a. Rename the current SYSTEM CONFIG file to BACKUP CONFIG.

   b. Rename the TEMP CONFIG file to SYSTEM CONFIG.

2. Shut down the system and do a warm start.

   ```
   shutdown reipl

   SYSTEM SHUTDOWN STARTED
   ⋮
   HCPWRP962I VM SHUTDOWN COMPLETED in nn SEC
   HCPWRP9277I SYSTEM TERMINATION COMPLETE,
                ATTEMPTING RESTART
   ⋮
   hh:mm:ss Start ((Warm|Force|COLD|CLEAN) (DRain)
             (DIsable) (NODIRect) (NOAUTOlog)) or (SHUTDOWN)

   warm
   ⋮
   ```

3. Log on to the MAINT*vrm* user ID.

Task 8: Load the Spool Files

1. If you dumped the spool files to tape in "Task 4: Manage the User Spool Files" on page 774, load them onto the system. This resets the spool file IDs to the numbering scheme used in an SSI cluster. (For more information, see "Spool File ID Assignment and Limits" on page 751.)

```
attach rdev to * as vdev
⋮
spxtape load vdev spool nodup
```

2. The system is now effectively running as a single-member SSI cluster. You should notice little or no difference in behavior or operation. Run the system for a while and make sure that you can log on the various user IDs. After you SSI-enable the user directory in the next task, you cannot easily back off to the old-style user directory.

   When you are satisfied that the modified system is operating properly, complete the remaining tasks in this procedure.

Task 9: Change the User Directory to SSI-Enabled

To complete the conversion to an SSI-enabled system and prepare for the addition of a second member system, modify the source directory to change it from an SSI-ready type to an SSI-enabled type.

1. In the source directory file, add the SSI operand to the DIRECTORY statement and change all BUILD ON * statements to BUILD ON VMSYS01.

DirMaint Alternative
If you are using DirMaint, go to step "2" on page 776

   If you are not using a directory manager, complete the following steps:

   a. Make a backup copy and then edit the USER DIRECT file.
   b. Update the DIRECTORY statement to insert the SSI operand. For example:

   ```
   DIRECTORY SSI 0123 3390 m01res
   ```

   Note: This must be the first statement in the directory and is the only DIRECTORY statement allowed. If the file currently contains other DIRECTORY statements, remove them.

   c. Update the BUILD statements in all of the multiconfiguration virtual machine definitions (IBM-supplied and customer-defined) to replace the asterisk with system name VMSYS01. For example:

   ```
   IDENTITY OPERATOR password 32M 32M ABCDEFG
    INCLUDE IBMDFLT
    BUILD ON VMSYS01 USING SUBCONFIG OPERTR-1
   ⋮
   IDENTITY WATCHER password 64M
    BUILD ON VMSYS01 USING SUBCONFIG WATCHR-1
   ⋮
   ```

   Save your changes.

   d. Create the new object directory for VMSYS01:

   ```
   directxa user direct fm
   ```

2. **If you are using DirMaint,** complete the following steps:

   a. Temporarily suspend placing the updated source directory online:

```
dirmaint offline
```

b. Prepare the directory to be used for member VMSYS01 of the SSI cluster:

```
dirmaint ssi vmsys01
```

This command adds the SSI operand to the DIRECTORY statement (removing any existing options) and changes all BUILD ON * statements to BUILD ON VMSYS01.

c. Create the new object directory for VMSYS01:

```
dirmaint direct
dirmaint online
```

3. Prior to adding additional members, the first system must be reIPLed with the SSI-enabled directory in order for the second and any subsequent systems to function properly. If you do not reIPL after switching to an SSI-enabled directory, you will have problems logging on multiconfiguration virtual machines on other members while the corresponding user IDs remain logged on to the first system.

Task 10: Clean Up

You have completed the conversion of your z/VM system to a single-member SSI cluster.

1. Edit the SYSTEM CONFIG file and remove the FEATURES statement that you added in "Task 2: Update the System Configuration File" on page 765.
2. Discard the backup system configuration file and the backup user directory. They are no longer usable.
3. To add another member to the SSI cluster, see Chapter 34, "Adding a Member to a z/VM SSI Cluster by Cloning an Existing Member," on page 785.

Chapter 33. Combining Two Non-SSI z/VM Systems to Create a z/VM SSI Cluster

This procedure outlines the process for combining two existing non-SSI z/VM systems to create a z/VM SSI cluster with two members. The process consists of converting one of the systems to cluster member 1, cloning member 1 to create member 2, and moving workload from the other non-SSI system into the cluster.

Before you begin the procedure, see "Requirements and Preparations" on page 779.

In this procedure you will complete the following tasks:

Requirements and Preparations

- Make sure that you understand how a z/VM SSI cluster is set up and maintained. For more information, see Chapter 29, "Setting Up z/VM Single System Image Clusters," on page 715.

  It is particularly important to review the following information:

  - "SSI Cluster Requirements" on page 728
  - "SSI Cluster Restrictions" on page 729
  - "Suggested Practices for Setting Up an SSI Cluster" on page 729
- Some tasks in this procedure direct you to complete certain other procedures that might have additional requirements. Make sure that you meet the requirements and complete the preparations documented in those procedures.
- The z/VM systems to be combined must meet the following requirements:
  - The systems must be z/VM 6.4 or later, installed by following one of the non-SSI installation procedures documented in *z/VM: Installation Guide*. The systems must be installed to ECKD DASD.
  - If either system was migrated from a previous z/VM release, all migration tasks must be completed before starting this procedure. For example, the customer-specific user information from the old system must be merged into the user directory on the new system.

    Attention: In z/VM 6.2 and later, the IBM-supplied source directory is arranged to facilitate SSI deployment (referred to as *SSI-ready*). The structure and layout of the directory differ significantly from pre-6.2 releases, **even in a non-SSI installation**. The directories for the two non-SSI systems to be combined must conform to the SSI-ready format so they can be merged into a single source directory for the SSI cluster.
  - The systems must not be part of a CSE complex. For information about converting a CSE complex to an SSI cluster, see Chapter 38, "Converting a CSE Complex to a z/VM SSI Cluster," on page 835.
  - This procedure does not cover the situation where the systems are members of an existing ISFC collection.

  - If an external security manager (ESM) is being used, it must be used on both members of the SSI cluster and the ESM database must be shared. For further information on how to configure the RACF database DASD for use in an SSI cluster, see *z/VM: RACF Security Server System Programmer's Guide*.
- Choose one system as the "master" system (called system A in this procedure). This is the system that you will convert to member 1 of the SSI cluster, which will set the attributes that will be common for all the members of the cluster. This system will also, by default, supply the virtual machine definitions for user IDs that exist on both systems.

  Some key considerations in deciding which of the two systems to designate as system A include:

  - If one system has significantly more resources (DASD, virtual machines, and so on) than the other, it might be advantageous to pick that one as system A. The non-master system (system B) will need to be replicated to some extent in the cluster. By using the smaller system for system B, you reduce the amount of replication and the extra resources needed.
  - Although member 1 of the SSI cluster will retain the system name of system A, member 2 must have a different system name than system B (because system B will still be running when member 2 is created). If there is a larger impact to one system for changing the system name, then that might be the reason to use it as system A.
  - Virtual machines with customer-assigned MAC addresses on system A can retain those MAC addresses in the SSI cluster. However, virtual machines with customer-assigned MAC addresses on system B might have different MAC addresses in the cluster. (For more information, see "Task 6: Move System B Workload into the SSI Cluster" on page 782.) If that is a problem for one of the systems, then make that one system A.
  - If an ESM is used, the SSI cluster will inherit the ESM characteristics of system A.
  - Service considerations:
    - System A must be at the same service level or a higher service level than system B.
    - Because cluster member 1 will be created from system A, and member 2 will be cloned from member 1, only local modifications on system A will be retained in the SSI cluster. Local modifications on system B will be lost.
- An assumption for using this procedure is that system A and system B are first-level systems. Therefore cluster member 1 will be first-level. If member 2 will be created first-level, a new logical partition is required. Ensure that the logical partition is defined and configured with access to the devices that will be shared with member 1. To enable you to move system B workload into the cluster, member 2 (and also member 1) will require access to the same devices as system B. To facilitate this, consider defining the new logical partition on the same central processor complex as system B.

Task 1: Convert System A to Member 1

Follow the procedure in Chapter 32, "Converting a z/VM System to a Single-Member z/VM SSI Cluster," on page 763. Make sure that you meet all of the requirements and complete the necessary preparations before you begin.

Task 2: Install Program Products and Vendor Products on Member 1

If there are products currently installed only on system B that you intend to run in the SSI cluster, install them on member 1 before it is cloned.

Note: Ensure that each product is licensed for member 1 and member 2.

Task 3: Clone Member 1 to Create Member 2

Follow the procedure in Chapter 34, "Adding a Member to a z/VM SSI Cluster by Cloning an Existing Member," on page 785. Make sure that you meet all of the requirements and complete the necessary preparations before you begin.

Additional requirements:

- Provide sufficient paging, spooling, and temporary disk volumes on member 2, and common user volumes in the cluster, for the workload that you intend to move from system B. You can simplify the updates in the SSI cluster directory by allocating the additional common user volumes one-for-one for the user volumes on system B, so that each minidisk on system B can be copied to the same cylinder (or block) range on the corresponding new volume.

  Note: For a system B user ID that will be defined as a multiconfiguration virtual machine in the SSI cluster, minidisks that might be located on the same user volume on system B will be located on separate user volumes in the cluster. Some minidisks might be located on a common user volume, but other minidisks will be located on member-specific user volumes.
- Give member 2 a new system name (different from system B).
- Customize the IP addresses of the TCP/IP service machines on member 2 (that is, use different IP addresses than on system B) to enable member 2 to coexist with production on system B.

Complete the entire cloning procedure, which will make member 2 a replica of member 1, capable of running the workload that member 1 runs (the workload from system A). Then return to this procedure and complete the remaining tasks to move system B configuration definitions and workload into the SSI cluster.

Task 4: Replicate System B Configuration Definitions in the SSI Cluster

Most member-specific updates to the SSI cluster system configuration file should have been handled in the cloning procedure. However, additional updates might be required to include system and network configuration definitions from system B. The following list identifies examples of some configuration definitions that should be examined and adjusted if necessary.

- Virtual switches
  - If a virtual switch with the same name exists on both system A and system B, but they are not intended to interconnect the same virtual machines, define a different name for one of the virtual switches on a new DEFINE VSWITCH statement.
  - Generally, a virtual switch is defined to be included in all members of the SSI cluster, so users can access that network when logged on to any member. However, if any virtual switch is intended to be member-specific, qualify the DEFINE VSWITCH statement for that member.

  Note: If you change the virtual switch configuration for a cluster member, you can issue the DEFINE VSWITCH and SET VSWITCH commands on that member to dynamically make the same changes for the current system IPL.
- MAC addresses
  - If the system B configuration includes a VMLAN statement with the MACIDRANGE operand, that operand is ignored in the SSI cluster and should not be copied.
  - Customer-assigned locally administered MAC addresses are created using the VMLAN USERPREFIX value. This value, which must be identical for all members of the SSI cluster, was established in the conversion of system A to member 1. Therefore virtual machines with customer-assigned MAC addresses on system B might have different MAC addresses in the SSI cluster. For more information, see "Task 6: Move System B Workload into the SSI Cluster" on page 782.
  - System-assigned locally administered MAC addresses are created using the VMLAN MACPREFIX value. This value must be different for each member of the cluster and also different from the USERPREFIX value. The MACPREFIX value for member 2 was established when member 2 was cloned from member 1.
- Operating parameters
  - In general, most operating parameters (defined on statements such as FEATURES and PRIV_CLASSES) should be set consistently for all members of the SSI cluster. Therefore the

    unqualified statements included from the conversion of system A to member 1 should suffice for the cluster.
  - If any operating parameters were unique to system A, those statements should be qualified for member 1. If any operating parameters are unique to system B, copy those statements and qualify them for member 2.
- System user IDs
  - Virtual machines identified on SYSTEM_USERIDS statements provide services that should be available on all members of the SSI cluster. Therefore they should be defined as multiconfiguration virtual machines. The user IDs for these virtual machines were inherited from system A in the conversion to member 1.
  - If any system user IDs defined only on system B will be added to the SSI cluster source directory, add the appropriate SYSTEM_USERIDS statements. If any of the user IDs apply only to member 2, qualify the statements for member 2. If any system user IDs from system A apply only to member 1, qualify those SYSTEM_USERIDS statements for member 1.
- Device definitions
  - Device definitions (statements such as RDEVICE, DEVICES, OPERATOR_CONSOLES, and EMERGENCY_MESSAGE_CONSOLES) that were common between system A and system B should also be common in the SSI cluster and should be handled by the unqualified statements included from the conversion of system A to member 1.
  - If any device definitions were unique to system A, qualify those statements for member 1. If any device definitions are unique to system B, copy those statements and qualify them for member 2.
- Product and feature licensing and enablement
  - Product and feature enablement for member 1 has been inherited from system A. If a product or feature on system A was not licensed for system B, obtain a license for member 2.
  - A product or feature that was licensed only for system B should be licensed for member 1 and member 2.
  - PRODUCT statements generally should not be qualified by member name, so that a properly licensed product is enabled for the entire SSI cluster.

Task 5: Re-IPL Member 2

If you changed the configuration for member 2 in the previous task, re-IPL member 2 to include those changes.

Note:

1. Re-IPL member 1 also if you changed its configuration. Qualifying statements for member 1 that were previously unqualified does not change its configuration.
2. You do not need to re-IPL a cluster member to include changes to the virtual switch configuration if those changes were also made dynamically.

Task 6: Move System B Workload into the SSI Cluster

Move customer-defined virtual machines and associated user data from system B into the SSI cluster.

The IBM-defined virtual machines are already included in the directory from the conversion of system A to member 1. The IBM-supplied multiconfiguration virtual machine definitions were updated to include member 2 in the cloning procedure. However, the current specifications for member 2 are essentially a replica of the specifications for member 1 and might need to be customized with the specifications from system B.

If an ESM is being used, it might be necessary to merge the system B ESM database into the shared ESM database for the SSI cluster.

Note: Although this task describes operating on one virtual machine at a time, it might be more convenient and efficient to process a set of virtual machines in parallel, such as those comprising a particular workload, or even all of the virtual machines being moved from system B. Before you can log on the user IDs in the cluster and test them, you will need to bring the updated directory online on both member 2 and member 1.

1. For each system B user ID to be included in the cluster, define the virtual machine in the cluster source directory:
   - If the user ID is to be defined as a single-configuration virtual machine:
     - If the user ID did not exist on system A, you can copy the virtual machine definition from the system B directory. Update each MDISK statement to define the minidisk on the common user volume added in the cloning procedure that corresponds to the system B volume that holds the minidisk.
     - If the user ID existed on system A, a virtual machine definition for that user ID is already included in the cluster source directory from the conversion of system A to member 1. Either select a version to keep (the one from system A or the one from system B), merge the two versions, or copy the virtual machine definition from system B and give one of the versions a new user ID in the cluster. If you keep the version from system B, merge the versions, or copy the virtual machine definition, update each MDISK statement copied from system B to define the minidisk on the common user volume added in the cloning procedure that corresponds to the system B volume that holds the minidisk. If you select one of the versions or merge them, you can place the other version's minidisks at different virtual device numbers, to allow continued access.
   - If the user ID is to be defined as a multiconfiguration virtual machine:
     - If the user ID did not exist on system A, create the multiconfiguration virtual machine definition.
     - If the user ID existed on system A, a virtual machine definition for that user ID is already included in the cluster source directory from the conversion of system A to member 1 and should be in the correct format. However, if you did not restructure the virtual machine definition as a multiconfiguration virtual machine definition in either the conversion procedure or the cloning procedure, do it now.
     - In either case, if an aspect of the virtual machine definition applies when the user ID logs on to any member of the cluster, such as an MDISK statement for a global minidisk, include the statement in the identity entry. Define a global minidisk on a common user volume.

       If an aspect of the virtual machine definition applies only when the user ID logs on to a specific member of the cluster, such as an MDISK statement for a local minidisk, include the statement in the subconfiguration entry for that member. Define a local minidisk on the corresponding member-specific user volume.

       MDISK statements and other specifications from system A that apply only to member 1 should be included in the subconfiguration entry for member 1.

       For each MDISK statement copied from system B that applies only to member 2, include the statement in the subconfiguration entry for member 2 and define the minidisk on the member 2 user volume added in the cloning procedure that corresponds to the system B volume that holds the minidisk.

       For an example of restructuring a virtual machine definition as a multiconfiguration virtual machine definition, see “Task 3: Restructure the User Directory” on page 768 in the conversion procedure. For an example of updating a multiconfiguration virtual machine definition for a new cluster member, see “Task 6: Update the User Directory” on page 791 in the cloning procedure.

   - If the virtual machine definition on system B includes a NICDEF statement with the MACID operand, the MAC address for that virtual network adapter will be retained in the SSI cluster **only** if the USERPREFIX value on system B (set or defaulted) is the same as the USERPREFIX value established for the SSI cluster in the conversion of system A to member 1. However, if that is the case, the system B user must not have the same MACID as a user on system A. In an SSI cluster, two devices cannot have the same MAC address. Therefore if two users in the cluster have the same MACID, the device will be defined for the first user to log on to any member.

- If you change the name of a virtual switch or define a new one, ensure that a corresponding NICDEF statement is included in the definition of each virtual machine intended to use that virtual switch.

2. To make the directory changes effective, create a new object directory on both member 1 and member 2, either manually (using the DIRECTXA utility) or using DirMaint or another directory manager.
3. For each system B user ID that was moved to the cluster and is now included in the online directory on member 1 and member 2, transfer the user data:

   a. Log off the user ID on system B.
   b. Change the password to NOLOG in the system B directory.

      ⚠ **Attention:** Before you log on this user ID in the SSI cluster, ensure that the NOLOG on system B has been made effective by creating a new object directory for system B.

   c. On member 2 (or member 1), use the FLASHCOPY command (or use the DDR utility) to copy the system B volumes or minidisks containing the user data to the volumes specified in the virtual machine definition in the SSI source directory.

      Note: If the user volume in the cluster has been allocated one-for-one for the user volume on system B, and you have moved all the users represented on that volume, you can copy the entire volume.

   d. On system B, use the SPXTAPE DUMP command to dump the spool files onto tape. This includes customer-defined saved segments containing user data (for example, Linux file systems) and customer-defined named saved systems (for example, an IPLable Linux).

      Note: In an SSI cluster, the number of spool files that a single-configuration virtual machine can own that were originated on any one particular member of the cluster is limited to one quarter of the MAXSPOOL value. Therefore you might need to purge some files for this user before dumping them.

   e. Log on the user ID on member 2 and member 1 and verify that the virtual machine operates correctly.
   f. On member 2, use the SPXTAPE LOAD command to load the spool files onto a spool volume owned by member 2.

Task 7: Test and Verify

Verify that everything that previously worked on system B now works correctly on members 1 and 2.

Task 8: Shut Down System B

Shut down system B and decommission its system and user volumes.

Chapter 34. Adding a Member to a z/VM SSI Cluster by Cloning an Existing Member

Use this procedure to add a member to a z/VM SSI cluster by cloning an existing member of the cluster.

See "Before You Begin the Cloning Procedure" on page 785 for requirements, suggested practices, and preparations. If you are adding a 5th member for the first time in slot 5, 6, 7, or 8, see Chapter 35, "Adding Members to a 4-Member SSI Cluster," on page 811 for additional considerations.

In this procedure you will complete the following tasks:

"Task 1: Prepare the CP-Owned Volumes for the Target Member" on page 786
"Task 2: Create the TCP/IP Configuration for the Target Member" on page 787
"Task 3: Update the Configuration Files for Other Service Virtual Machines and Servers" on page 788
"Task 4: Customize the System Startup Virtual Machine " on page 788
"Task 5: Copy the Source Volumes to the Target Volumes" on page 789
"Task 6: Update the User Directory" on page 791, or "Task 6A: Update the User Directory Using DirMaint" on page 796
"Task 7: Update the System Configuration File" on page 801
"Task 8: Enable the Existing Members to Access the Target Member" on page 806
"Task 9: IPL the Target Member" on page 806
"Task 10: Start the ESM Service Virtual Machine" on page 806
"Task 11: Update the VMSES/E System-Level Product Inventory Table" on page 807
"Task 12: Update the CRR Server LU Name" on page 807
"Task 13: Build the Saved Segments and Named Saved Systems" on page 807
"Task 14: Start the Service Virtual Machines" on page 808
"Task 15: Test and Verify" on page 809

Before You Begin the Cloning Procedure

Attention

Make sure that you understand the cloning process. Failure to adhere to the documented cloning procedure can result in an unstable or unusable system.

Do not use this procedure if the current members of the cluster were installed with SMAPI enabled.

Make sure that you meet all of the requirements and complete the necessary preparations before you start. The examples in this procedure assume that you are following the suggested practices.

Requirements:

- Adhere to the "SSI Cluster Requirements" on page 728 and "SSI Cluster Restrictions" on page 729.
- Apply and place into production all needed service on the member to be cloned before you begin the cloning procedure.
- Once you begin the cloning procedure, do not apply service to any member of the cluster until the procedure is completed.
- If you are using RACF, the RACF database must be configured to be shared by the members of the SSI cluster. If the shared database is not already set up, you must set it up before you use this cloning procedure. See the information on sharing RACF databases in an SSI cluster in *z/VM: RACF Security Server System Programmer's Guide*.

Suggested practices:

- See "Suggested Practices for Setting Up an SSI Cluster" on page 729.

Preparations:

- Ensure that a logical partition for the new member has been defined and configured with access to the necessary devices.

  Note: Although this procedure describes how to create a new member in a first-level SSI cluster, the new member could be created second-level in a virtual machine.

- Gather the information you need to define the new member. The following values are used in this procedure:
  - Logical partition (LPAR) name = LP02

    Note: At second level, this is the user ID of the virtual machine in which the new member will be initialized.
  - Cluster name = CLUSTERA
  - Source member (member to be cloned) = VMSYS01

    In this example, VMSYS01 is a single-member SSI cluster. However, in a multimember SSI cluster, the source member could be any existing member of the cluster.
  - Target member (member to be created) = VMSYS02

    In this example, VMSYS02 is the second member of the SSI cluster. However, the same procedure would be used to add other members.
- Acquire a set of DASD volumes for the new member.

  Attention: Volume labels must be unique within the SSI cluster.

  The examples in this procedure use the default volume labels that would be created by the installation procedure for a multimember SSI cluster. For most volumes the naming convention is M0*mvnn*, where *m* is the number of this member in the SSI member list (on the SSI configuration statement) and *vnn* identifies the volume type and number (for example, P01 for the first paging volume).
  - CP-owned volumes:
    - System residence volume = M02RES
    - System paging volume = M02P01
    - Spool volume = M02S01
  - User volume:
    - Member-specific user volume = M02PV1
- Obtain the TCP/IP configuration information for the new member:
  - If possible, obtain the IP addresses for the new member before cloning.
  - If you are running an SSL server, you will need either a multiple-server certificate that covers the new member or a single-server certificate for the new member. For information on defining your SSL server environment, see *z/VM: TCP/IP Planning and Customization*.

Task 1: Prepare the CP-Owned Volumes for the Target Member

Prepare the following CP-owned volumes for the target member:

- System paging volume (M02P01)
- Spool volume (M02S01)

Important: Do not format the system residence volume in this task.

Note: For simplicity, this procedure does not show all of the system responses.

1. On the source member, log on to the MAINT*vrm* user ID for the release level of that system. Ensure that you have access to the cross release utilities disk (551).
2. Attach the target volumes to your virtual machine:

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

3. Format, label, and allocate each volume.

   Attention: Cylinder 0 is allocated as PERM by the formatting process and must not be reallocated.

   For example, to prepare the system paging volume:

   ```
   cpfmtxa vdev m02p01

   CPFMTXA:
   FORMAT WILL ERASE CYLINDERS 00000-nnnnn ON DISK vdev
   DO YOU WANT TO CONTINUE? (YES | NO)

   yes

   HCPCCF6209I INVOKING ICKDSF.
   ⋮
   ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
   ENTER ALLOCATION DATA
   TYPE CYLINDERS

   page 1-end
   end

   HCPCCF6209I INVOKING ICKDSF.
   ⋮
   ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
   ```

   Note: When preparing the spool volume, specify `spol 1-end` instead of `page 1-end`.
4. Mark the SSI ownership of the volumes. Mark each volume with the cluster name and system name. For example, to mark the paging volume, enter the following command:

   ```
   cpfmtxa vdev m02p01 owner clustera.vmsys02
   ```

Task 2: Create the TCP/IP Configuration for the Target Member

Note: For more information about the TCP/IP configuration files identified in this task, see *z/VM: TCP/IP Planning and Customization*.

1. Configure the TCP/IP stack:

   a. Link and access the TCPMAINT 198 minidisk:

      ```
      link to tcpmaint 198 as 198 m
      access 198 fm
      ```

      i) The TCPIP server configuration file for VMSYS01 must be named VMSYS01 TCPIP (*sysname* TCPIP). If it is currently named PROFILE TCPIP or some other name, rename it to VMSYS01 TCPIP.

      ii) Make a copy of the VMSYS01 TCPIP file on the 198 minidisk and name the copy VMSYS02 TCPIP.

      iii) If you know the IP addresses for VMSYS02, update the HOME statement in the VMSYS02 TCPIP file.

      iv) There might be other files on this minidisk that should be made multinode capable before cloning. For example:

      - By putting member-specific logic in the global profile exit (TCPRUNXT EXEC).

  - If you need member-specific configurations for other TCP/IP servers, by creating separate *sysname* DTCPARMS files (VMSYS01 DTCPARMS and VMSYS02 DTCPARMS).

b. Link and access the TCPMAINT 592 minidisk:

```
link to tcpmaint 592 as 592 m
access 592 fm
```

i) Edit the TCPIP DATA file and add a HOSTNAME statement to identify the host name for the target member. Qualify each HOSTNAME statement with the system name of the member.

```
VMSYS01: HOSTNAME VMSYS01
VMSYS02: HOSTNAME VMSYS02
```

2. Update the SSL server certificate:

   - If you obtained a new multiple-server certificate that covers system VMSYS02, replace the current certificate with the new certificate now. For information about managing the certificate database, see *z/VM: TCP/IP User's Guide*.
   - If you obtained a new single-server certificate for VMSYS02, you will replace the certificate in a later task in this procedure.

Task 3: Update the Configuration Files for Other Service Virtual Machines and Servers

If service virtual machines (SVMs) or servers for other facilities, features, or products will have member-specific instances in the SSI cluster, update the corresponding configuration files to provide the member-specific information, if necessary. For example, if you will use the Performance Toolkit for centralized monitoring in your SSI cluster, see the preparation information in *z/VM: Performance Toolkit Guide*.

Note: DirMaint configuration updates are identified in "Task 6A: Update the User Directory Using DirMaint" on page 796.

Task 4: Customize the System Startup Virtual Machine

The system startup virtual machine (AUTOLOG1) is defined by a multiconfiguration virtual machine definition. Therefore AUTOLOG1 will have a separate logon instance and A-disk on each member, and potentially a member-specific profile. However, when the work volume for the source member is copied to the target volume later in this procedure, AUTOLOG1 will have the identical profile on both members.

- If the profile includes an XAUTOLOG command for an SVM or server that you want logged on only to a specific member of the SSI cluster, provide some logic in the profile to process the XAUTOLOG command for that user ID only when that member is initialized.
- If the profile includes an XAUTOLOG command for an SVM or server that needs to be logged on but the specific member of the cluster is not important, you can include an XAUTOLOG command with no conditional processing. The service virtual machine will be logged on automatically to the member that is initialized first.
- If the profile includes an XAUTOLOG command for an SVM or server that you want to be able to run concurrently on multiple members of the cluster, you can include an XAUTOLOG command with no conditional processing in the profile, but you need to restructure the virtual machine definition as a multiconfiguration virtual machine definition (when you update the source directory later in this procedure).

Customize the system startup machine:

1. Link and access the AUTOLOG1 191 minidisk:

```
link to autolog1 191 as vdev m
access vdev fm
```

Note: If you are using RACF, make the changes to the AUTOLOG2 virtual machine instead of AUTOLOG1.

2. Edit the PROFILE EXEC file and provide the necessary logic to conditionally process the XAUTOLOG commands that are member-specific.

Task 5: Copy the Source Volumes to the Target Volumes

Copy the source member's DASD volumes to the target member's volumes.

1. Attach the VMSYS02 system residence volume:

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

   The M01RES volume (full-pack minidisk MAINT 123) is already linked to MAINT*vrm* as 123.

2. Copy M01RES to the VMSYS02 system residence volume and label it M02RES:

```
flashcopy 123 0 end to vdev 0 end synchronous label m02res
```

   Notes:

   - The SYNCHRONOUS operand is required only for FlashCopy Version 1 hardware.
   - Alternatively, you can use the DDR utility to copy the volume. After the copy completes, issue the CPFMTXA command to change the label of the target volume to M02RES:

```
cpfmtxa vdev m02res label
```

3. Format the checkpoint and warm start areas on M02RES to erase the VMSYS01 data.

   a. Link (read-only) and access the PMAINT CF0 minidisk:

```
link to pmaint cf0 as cf0 r
access cf0 fm
```

   b. Open the SYSTEM CONFIG file and examine the SYSTEM_RESIDENCE statement for VMSYS01 to determine the location and size of the checkpoint and warm start areas.

```
/**********************************************************************/
/*                Checkpoint and Warmstart Information                */
/**********************************************************************/

   VMSYS01:  System_Residence,
             Checkpoint  Volid M01RES   From CYL 21  For 9 ,
             Warmstart   Volid M01RES   From CYL 30  For 9
```

   c. Format those cylinders on the M02RES volume. For example:

```
cpfmtxa vdev m02res 21.9

INVOKING ICKDSF.
FORMAT WILL ERASE CYLINDERS 21-29 ON DISK vdev
DO YOU WANT TO CONTINUE? (YES | NO)

yes

HCPCCF6209I INVOKING ICKDSF.
⋮
ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
ENTER ALLOCATION DATA
TYPE CYLINDERS

end

HCPCCF6209I INVOKING ICKDSF.
⋮
ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

```
cpfmtxa vdev m02res 30.9

INVOKING ICKDSF.
FORMAT WILL ERASE CYLINDERS 30-38 ON DISK vdev
DO YOU WANT TO CONTINUE? (YES | NO)

yes

HCPCCF6209I INVOKING ICKDSF.
⋮
ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
ENTER ALLOCATION DATA
TYPE CYLINDERS

end

HCPCCF6209I INVOKING ICKDSF.
⋮
ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

4. Mark the SSI ownership of M02RES with the cluster name and system name:

   ```
   cpfmtxa vdev m02res owner clustera.vmsys02
   ```

5. If you are following the suggested practice of using the same real device numbers for DASD across the LPARs, skip to step "6" on page 791.

   However, if the real device number for the VMCOM1 device in LPAR LP02 is different from the real device number for that device in LPAR LP01, write a new copy of the Stand-Alone Program Loader (SAPL) on M02RES and specify the real device number for VMCOM1 in LP02.

   a. Determine what IPL parameters (if any) were specified for SAPL on M01RES so you can specify the same parameters when you write the new copy of SAPL on M02RES. Because SAPL on M02RES is currently a duplicate of SAPL on M02RES, you can get this information by IPLing M02RES in your virtual machine and looking at the SAPL panel.

      i) Do a system-reset-clear on your virtual machine:

      ```
      system clear
      ```

      ii) Change your virtual console mode to 3270:

      ```
      terminal conmode 3270
      ```

      iii) Display the status of your virtual console to get the virtual device number (usually 0009):

      ```
      query virtual console

      CONS vdev ON LDEV  …
      ⋮
      ```

      iv) IPL the M02RES device:

      ```
      ipl M02RES_vdev loadparm console_vdev
      ```

      v) When the SAPL panel is displayed, record the parameters specified in the **IPL PARAMETERS** field.

      vi) Use the PA1 (Program Attention) key, or equivalent, to exit SAPL and get back to CP.

      Note: In an IBM Personal Communications 3270 emulation session, right click on the session screen to display Pad 1 and click on the PA1 button.

      vii) IPL CMS.

   b. Write the new SAPL on M02RES, specifying the IPL parameters that you recorded plus the PDVOL parameter to specify the real device number for VMCOM1 in LPAR LP02:

      ```
      salipl vdev (iplparms IPL_parameters pdvol=rdev
      ```

6. Copy the VMSYS01 member-specific user volume to the corresponding VMSYS02 volume and relabel it.

Attention
Complete this process for one volume before starting the next volume, and do not stop the process until the volume has been relabeled. If system VMSYS01 goes down, it might come back up with the VMSYS02 volume instead of the VMSYS01 volume (if the VMSYS02 volume has a lower real device number).

   For example, to create member-specific user volume M02PV1:

   a. Attach the VMSYS02 volume to your virtual machine:

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

   b. Copy the VMSYS01 volume to the VMSYS02 volume and relabel it:

   ```
   flashcopy 124 0 end to vdev 0 end synchronous label m02pv1
   ```

   Notes:

   i) The SYNCHRONOUS operand is required only for FlashCopy Version 1 hardware.

   ii) Alternatively, you can use the DDR utility to copy the minidisks.

   iii) In this example, WATCHER 191 has been moved to a different location on M01PV1 than on USRVL1.

   Notes:

   - The SYNCHRONOUS operand is required only for FlashCopy Version 1 hardware.
   - Alternatively, you can use the DDR utility to copy the volume.

   Attention: If you use DDR, immediately after the copy completes, issue the CPFMTXA command to change the label of the target volume to the VMSYS02 value:

   ```
   cpfmtxa vdev  m02pv1 label
   ```

   c. Detach the VMSYS02 volume.

   d. Follow the same process for the next volume.

7. If you obtained a new single-server certificate for the SSL server on VMSYS02, replace the certificate copied from VMSYS01 now. For information about managing the certificate database, see *z/VM: TCP/IP User's Guide*.

Task 6: Update the User Directory

Update the source directory to add the target member to the multiconfiguration virtual machine definitions.

DirMaint Alternative
If you have DirMaint installed and enabled in the SSI cluster, skip this task and go to "Task 6A: Update the User Directory Using DirMaint" on page 796.

1. Make a backup copy and then edit the common source directory (USER DIRECT).
2. Update the DIRECTORY statement to add M02RES as the volume where the object directory for VMSYS02 will be created.

```
DIRECTORY SSI 123 3390 M01RES M02RES
```

3. Update the dummy user IDs $ALLOC$, $DIRECT$, $SYSCKP$, $SYSWRM$, $PAGE$ and $SPOOL$ to protect certain areas of the new volumes, such as cylinder 0 and directory, warm start, and checkpoint areas.

   a. For USER $ALLOC$, create new MDISK statements to protect cylinder 0 on each of the RES and MEMBER volumes that you copied. For example, the label for the new volume in our example is M02RES. You would add the following MDISK statement to the directory entry for $ALLOC$:

   ```
   MDISK  B02  3390  000 001 M02RES R
   ```

   b. For USER $DIRECT$, create a new MDISK statement to protect the directory space on the new RES volume. In our example, the label for the new volume is M02RES. You would add the following MDISK statement to the directory entry for $DIRECT$:

   ```
    MDISK  A02  3390  001 020 M02RES R
   ```

   c. For USER $SYSCKP$, create a new MDISK statement to protect the system checkpoint space on the new RES volume. The label for the new RES volume in our example is M02RES. You would add the following MDISK statement to the directory entry for $SYSCKP$:

   ```
    MDISK  A02  3390  021 009 M02RES R
   ```

   d. For USER $SYSWRM$, create a new MDISK statement to protect the system warm start area on the new RES volume. The label for the new RES volume in our example is M02RES. You would add the following MDISK statement to the directory entry for $SYSWRM$:

   ```
   MDISK  A02  3390  030 009 M02RES R
   ```

   e. For USER $PAGE$, create a new MDISK statement (or statements) to protect your entire paging volume (or volumes, if you are adding more than one paging volume). The label for the new paging volume in our example is M02P01. You would add the following MDISK statement to the directory entry for $PAGE$:

   ```
   MDISK  A02  3390  000 END M02P01 R
   ```

   f. For USER $SPOOL$, create a new MDISK statement (or statements) to protect your entire spool volume (or volumes, if you are adding more than one spool volume). The label for the new spool volume in our example is M02S01. You would add the following MDISK statement to the directory entry for $SPOOL$:

   ```
   MDISK  A02  3390  000 END M02S01 R
   ```

   Note: If any of the suggested minidisk addresses are already in use for the user, select any other address that is not in use.

4. Update each IBM-supplied multiconfiguration virtual machine definition to include a subconfiguration entry for VMSYS02:

   a. In the identity entry, uncomment the BUILD statement for member 2 and update the statement to specify system name VMSYS02.

   b. Uncomment the corresponding subconfiguration entry.

   c. If the subconfiguration entry for VMSYS01 contains MDISK statements, add corresponding MDISK statements to the subconfiguration entry for VMSYS02. The MDISK statements in the VMSYS02 subconfiguration entry must specify VMSYS02 volumes. For example, if the VMSYS01 subconfiguration entry defines a 191 minidisk on M01RES, the VMSYS02 subconfiguration entry must define the 191 minidisk on M02RES.

   The following examples show the original and updated definition for the TCPIP user ID.

Example of the original TCPIP definition:

Note: The real TCPIP definition could be different from this example.

```
IDENTITY TCPIP password 128M 256M ABG
 INCLUDE TCPCMSU
 BUILD ON VMSYS01 USING SUBCONFIG TCPIP-1
* BUILD ON @@member2name USING SUBCONFIG TCPIP-2
* BUILD ON @@member3name USING SUBCONFIG TCPIP-3
* BUILD ON @@member4name USING SUBCONFIG TCPIP-4
 OPTION QUICKDSP SVMSTAT MAXCONN 1024 DIAG98 APPLMON
 SHARE RELATIVE 3000
 IUCV ALLOW
 IUCV ANY PRIORITY
 IUCV *CCS PRIORITY MSGLIMIT 255
 IUCV *VSWITCH MSGLIMIT 65535

SUBCONFIG TCPIP-1
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M01RES MR RTCPIP WTCPIP MTCPIP

*SUBCONFIG TCPIP-2
* LINK TCPMAINT 491 491 RR
* LINK TCPMAINT 492 492 RR
* LINK TCPMAINT 591 591 RR
* LINK TCPMAINT 592 592 RR
* LINK TCPMAINT 198 198 RR

*SUBCONFIG TCPIP-3
* LINK TCPMAINT 491 491 RR
* LINK TCPMAINT 492 492 RR
* LINK TCPMAINT 591 591 RR
* LINK TCPMAINT 592 592 RR
* LINK TCPMAINT 198 198 RR

*SUBCONFIG TCPIP-4
* LINK TCPMAINT 491 491 RR
* LINK TCPMAINT 492 492 RR
* LINK TCPMAINT 591 591 RR
* LINK TCPMAINT 592 592 RR
* LINK TCPMAINT 198 198 RR
```

Example of the updated TCPIP definition:

```
IDENTITY TCPIP password 128M 256M ABG
 INCLUDE TCPCMSU
 BUILD ON VMSYS01 USING SUBCONFIG TCPIP-1
 BUILD ON VMSYS02 USING SUBCONFIG TCPIP-2
* BUILD ON @@member3name USING SUBCONFIG TCPIP-3
* BUILD ON @@member4name USING SUBCONFIG TCPIP-4
 OPTION QUICKDSP SVMSTAT MAXCONN 1024 DIAG98 APPLMON
 SHARE RELATIVE 3000
 IUCV ALLOW
 IUCV ANY PRIORITY
 IUCV *CCS PRIORITY MSGLIMIT 255
 IUCV *VSWITCH MSGLIMIT 65535

SUBCONFIG TCPIP-1
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M01RES MR RTCPIP WTCPIP MTCPIP

SUBCONFIG TCPIP-2
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M02RES MR RTCPIP WTCPIP MTCPIP

*SUBCONFIG TCPIP-3
* LINK TCPMAINT 491 491 RR
* LINK TCPMAINT 492 492 RR
* LINK TCPMAINT 591 591 RR
* LINK TCPMAINT 592 592 RR
* LINK TCPMAINT 198 198 RR

*SUBCONFIG TCPIP-4
* LINK TCPMAINT 491 491 RR
* LINK TCPMAINT 492 492 RR
* LINK TCPMAINT 591 591 RR
* LINK TCPMAINT 592 592 RR
* LINK TCPMAINT 198 198 RR
```

5. Update each customer-defined multiconfiguration virtual machine definition to include a subconfiguration entry for VMSYS02:

   a. In the identity entry, add a BUILD statement for VMSYS02:

      i) Copy the BUILD statement for VMSYS01 and change the system name to VMSYS02.

      ii) Change the SUBCONFIG ID to *name-2*.

   b. Add the subconfiguration entry for VMSYS02:

      i) Copy the *name-1* subconfiguration entry and change the ID to *name-2*.

      ii) Change the volume labels on the MDISK statements to the corresponding volumes for VMSYS02.

      iii) Update other statements in the VMSYS02 configuration as required, such as DEDICATE and COMMAND statements.

      Note: NICDEF statements do not need to be changed if the virtual switch is the same on all members of the cluster.

   The following example shows the updated WATCHER definition.

```
IDENTITY WATCHER password 64M 64M G
   BUILD ON VMSYS01 USING SUBCONFIG WATCHR-1
   BUILD ON VMSYS02 USING SUBCONFIG WATCHR-2
   IPL CMS
   LINK MAINT 190 190
   LINK MAINT 19E 19E
   MDISK 192 3390 300 100 USRVL1 RR
⋮

SUBCONFIG WATCHR-1
   MDISK 191 3390 100 100 M01PV1 MR
⋮

SUBCONFIG WATCHR-2
   MDISK 191 3390 100 100 M02PV1 MR
⋮
```

6. After you finish editing the file, check the minidisk assignments:

   ```
   diskmap user direct fm
   ```

   The output file is written to your A disk. If any overlapping minidisks are flagged, edit the user directory file to make any needed adjustments and run DISKMAP again.

7. Create a new object directory for VMSYS01:

   ```
   directxa user direct fm
   ```

8. If the SSI cluster includes other existing members, create a new object directory on each of those members. For example, if you are adding member 4, do the following operations on members 2 and 3.

   a. Log on to the MAINT user ID on that member.

   b. Access the PMAINT 2CC and 551 minidisks.

   c. Create the new object directory:

      ```
      directxa user direct fm
      ```

9. If you are using RACF, edit the source directory again and modify the definition for the AUTOLOG2 user ID to change its password to NOLOG.

   After you IPL the new VMSYS02 system later in this procedure, you need to start the RACFVM virtual machine. When RACFVM is initialized, it issues an XAUTOLOG command for AUTOLOG2, which issues XAUTOLOG commands for the SVMs identified in its profile. However, you need to start RACFVM without logging on those SVMs, so you can do some other operations first. After you complete those operations you will modify the AUTOLOG2 user ID again to restore its password (so keep a record of the original password), and then autolog AUTOLOG2 to start the SVMs.

   Important: To ensure that this change affects only the object directory for the new member of the cluster, you should delay putting any directory changes online for the other members until after this change is rescinded.

10. On system VMSYS01, create the object directory for VMSYS02:

    a. Detach M01RES and attach M02RES as virtual device 123:

       ```
       detach 123
       attach rdev to * as 123
       ```

    b. Create the object directory:

       ```
       directxa user direct fm
       ```

    c. Detach M02RES:

```
detach 123
```

d. Re-link M01RES:

```
link maint 123 123 m
```

Task 6A: Update the User Directory Using DirMaint

If you completed "Task 6: Update the User Directory" on page 791, skip this task and go to "Task 7: Update the System Configuration File" on page 801.

In this task you will update the DirMaint configuration to include the new member, and you will use DirMaint facilities to help you update the user directory.

Note: The DIRMAINT commands used in this task must be issued from a cluster member already set up (VMSYS01, for example). For information about the DIRMAINT commands, see *z/VM: Directory Maintenance Facility Commands Reference*. For more information about the DirMaint configuration files updated in this task, see *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*.

1. Update the DIRECTORY statement to specify M02RES as the volume where the object directory for VMSYS02 will be created:

   ```
   dirmaint directory change 1 ssi 123 3390 m01res m02res
   ```

2. Add the virtual machine definitions for the DIRMSAT and DATAMOVE servers for the new member, DIRMSAT2 and DATAMOV2.

   a. Get a copy of the DIRMSAT virtual machine definition:

   ```
   dirmaint for dirmsat get nolock
   ```

   b. Receive the DIRMSAT DIRECT file from your reader as DIRMSAT2 DIRECT.

   c. Xedit the DIRMSAT2 DIRECT file and change the volume label on each MDISK statement to the corresponding system volume for member 2.

   The following examples shows the DIRMSAT definition and the new DIRMSAT2 definition.

   Note: The real DIRMSAT definition could be different from this example.

```
USER DIRMSAT  AUTOONLY  128M  256M BG
 IPL CMS PARM AUTOCR
 MACHINE ESA
 ACCOUNT SYSTEM SYSPROG
 D8ONECMD FAIL LOCK
 OPTION CONCEAL D84NOPAS IGNMAXU
 IUCV ANY PRIORITY MSGLIMIT 100
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR * CMS system disk
 LINK MAINT 19D 19D RR * help disk
 LINK MAINT 19E 19E RR * Product code disk
 LINK MAINT 123 123 MW * Object directory disk
 LINK DIRMAINT 191 191 RR
 LINK DIRMAINT 192 192 RR
 LINK DIRMAINT 11F 11F RR
 LINK DIRMAINT 21F 21F RR
 LINK DIRMAINT 1DF 1DF RR
 LINK DIRMAINT 15D 15D RR
 LINK DIRMAINT 2DF 2DF RR
 LINK PMAINT 551 551 RR
 MDISK 155 3390 04057 009 M01RES MR
 MDISK 1AA 3390 04066 020 M01RES MR
 MDISK 1DE 3390 04086 020 M01RES MR
 MDISK 1FA 3390 04106 012 M01RES MR
 MDISK 2AA 3390 04118 020 M01RES MR
```

```
USER DIRMSAT2  AUTOONLY  128M  256M BG
 IPL CMS PARM AUTOCR
 MACHINE ESA
 ACCOUNT SYSTEM SYSPROG
 D8ONECMD FAIL LOCK
 OPTION CONCEAL D84NOPAS IGNMAXU
 IUCV ANY PRIORITY MSGLIMIT 100
 CONSOLE 009 3215 T
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR * CMS system disk
 LINK MAINT 19D 19D RR * help disk
 LINK MAINT 19E 19E RR * Product code disk
 LINK MAINT 123 123 MW * Object directory disk
 LINK DIRMAINT 191 191 RR
 LINK DIRMAINT 192 192 RR
 LINK DIRMAINT 11F 11F RR
 LINK DIRMAINT 21F 21F RR
 LINK DIRMAINT 1DF 1DF RR
 LINK DIRMAINT 15D 15D RR
 LINK DIRMAINT 2DF 2DF RR
 LINK PMAINT 551 551 RR
 MDISK 155 3390 04057 009 M02RES MR
 MDISK 1AA 3390 04066 020 M02RES MR
 MDISK 1DE 3390 04086 020 M02RES MR
 MDISK 1FA 3390 04106 012 M02RES MR
 MDISK 2AA 3390 04118 020 M02RES MR
```

d. Add the DIRMSAT2 virtual machine definition to the source directory:

```
dirmaint add dirmsat2
```

e. Go back to step "2.a" on page 796 and repeat the sequence to create the DATAMOV2 DIRECT file from a copy of DATAMOVE DIRECT.

f. Issue the following command to create the DVHPROFA DIRMSAT2 file:

```
dirmaint cms copyfile dvhprofa dirmsat c dvhprofa dirmsat2 c
```

3. Update the CONFIGSS DATADVH file to add the DIRMSAT2 and DATAMOV2 servers.

a. Add a SATELLITE_SERVER statement to define the DIRMSAT2 server. Add a DATAMOVE_MACHINE statement to define the DATAMOV2 server.

```
SATELLITE_SERVER= DIRMSAT VMSYS01
SATELLITE_SERVER= DIRMSAT2 VMSYS02
DATAMOVE_MACHINE= DATAMOVE VMSYS01 *
DATAMOVE_MACHINE= DATAMOV2 VMSYS02 *
```

Note: If you are using an external security manager (ESM), such as RACF, the DIRMSAT2 and DATAMOV2 servers also must be identified to the ESM. See the external security manager considerations in *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*.

b. Reload the data tables from the CONFIG* DATADVH disk files:

```
dirmaint rlddata
```

4. Update the EXTENT CONTROL file to define the minidisk allocation data for the new member. When you add subconfiguration entries to the multiconfiguration virtual machine definitions (in step "6" on page 799), DirMaint will use the information from this file to identify the volumes to be used for minidisks in the new subconfiguration entries.

   Note: The following example uses the same extents on the VMSYS02 volumes as on the VMSYS01 volumes.

   a. Send a copy of the EXTENT CONTROL file to your virtual reader:

   ```
   dirmaint send extent control
   ```

   b. Receive the EXTENT CONTROL file and edit it.

      i) The :REGIONS. section defines the areas to be used for minidisks on the DASD volumes. Entries in this section are not required in order to clone SUBCONFIG directory entries if the entire volumes specified in the :SSI_VOLUMES. section are used for minidisk allocation and the volumes are generic 3390 volumes with 1113 cylinders. However, it is a good practice to complete this section, because entries are required to use the AUTOR method of automatic DASD space allocation provided in the DirMaint DASD allocation commands. Entries are always required for any volumes that are not generic 3390 volumes, so DirMaint knows how many cylinders are available. For each DASD device type and model specified in this section, there should be a corresponding entry in the :DEFAULTS. section.

      The following example shows entries for the system residence volume and local user volume for cluster member 1 and new entries for the corresponding volumes for cluster member 2:

      ```
      :REGIONS.
      RegionA  M01RES  START  END  3390-01
      RegionB  M01PV1  START  END  3390-01
      RegionC  M02RES  START  END  3390-01
      RegionD  M02PV1  START  END  3390-01
      :END.
      ```

      ii) Update the :SSI_VOLUMES. section to specify the user volumes on the members of the SSI cluster to be used for allocating minidisks. A volume set name (such as SYSRES) is used to identify the volumes used for the same purpose on each member.

      ```
      :SSI_VOLUMES.
      SYSRES    VMSYS01   M01RES
      SYSRES    VMSYS02   M02RES
      PRIVATE1  VMSYS01   M01PV1
      PRIVATE1  VMSYS02   M02PV1
      :END.
      ```

iii) Update the :DEFAULTS. section to specify the maximum number of cylinders or blocks for any DASD device type and model identified in the :REGIONS. section. For example:

```
:DEFAULTS.
3390-01     1113
:END.
```

c. Replace the updated EXTENT CONTROL file and reload the data:

```
dirmaint file extent control
dirmaint rldextn
```

5. DirMaint automatically protects the directory space (.DRCT.), paging volumes (.PAGE.), spool volumes (.SPOOL.), and TDISK space (.TDISK.) based on their allocations as returned by the CP QUERY ALLOC command. However, the system checkpoint space and the system warm start areas must be protected as follows:

   a. For USER $SYSCKP$, create a new MDISK statement to protect the system checkpoint space on the new RES volume. The label for the new RES volume in our example is M02RES. To protect the checkpoint space on the new RES volume (in this example, it begins at cylinder 21 for 9 cylinders), issue a DirMaint command like the following example:

   ```
   DIRM FOR $SYSCKP$ AMDISK A02 3390 021 009 M02RES R
   ```

   b. For USER $SYSWRM$, create a new MDISK statement to protect the system warm start area on the new RES volume. The label for the new RES volume in our example is M02RES. To protect the warm start area on the new RES volume (in this example, it begins at cylinder 30 for 9 cylinders), issue a DirMaint command like the following example:

   ```
   DIRM FOR $SYSWRM$ AMDISK A02 3390 030 009 M02RES R
   ```

   Note: If any of the minidisk addresses suggested in these examples are already in use for the user, select any other address that is not in use.

6. Update all of the multiconfiguration virtual machine definitions (IBM-supplied and customer-defined) to add a subconfiguration entry for VMSYS02.

   a. Search the directory for all of the BUILD ON statements for member VMSYS01:

   ```
   dirmaint scan build on vmsys01
   ```

   The matching records are sent to you in a file. For example:

   ```
   Userid:  <=== Qualifying Record ======>
   MAINT    BUILD ON VMSYS01 USING SUBCONFIG MAINT-1
   AVSVM    BUILD ON VMSYS01 USING SUBCONFIG AVSVM-1
   TSAFVM   BUILD ON VMSYS01 USING SUBCONFIG TSAFVM-1
   GCS      BUILD ON VMSYS01 USING SUBCONFIG GCS-1
   AUDITOR  BUILD ON VMSYS01 USING SUBCONFIG AUDITR-1
   AUTOLOG1 BUILD ON VMSYS01 USING SUBCONFIG AUTLG1-1
   CMSBATCH BUILD ON VMSYS01 USING SUBCONFIG CMSBAT-1
   ⋮
   TCPIP    BUILD ON VMSYS01 USING SUBCONFIG TCPIP-1
   ⋮
   ```

   b. For each user ID listed in the search results, issue the following command to add a new BUILD statement and subconfiguration entry for VMSYS02:

   ```
   dirmaint add name-2 like name-1 build on vmsys02 in userid
   ```

For example, the following command will make the changes shown in the example of the virtual machine definition for the TCPIP user ID:

```
dirmaint add tcpip-2 like tcpip-1 build on vmsys02 in tcpip
```

Notes:

- The real TCPIP definition could be different from this example.
- The DIRMAINT ADD command adds new statements to the virtual machine definition. It does not uncomment any sample BUILD statement or subconfiguration entry that might be included in the definition.
- The :SSI_VOLUMES. section of the EXTENT CONTROL file identifies the volumes used in the new subconfiguration entry.

```
IDENTITY TCPIP password 128M 256M ABG
 INCLUDE TCPCMSU
 BUILD ON VMSYS01 USING SUBCONFIG TCPIP-1
 BUILD ON VMSYS02 USING SUBCONFIG TCPIP-2
* BUILD ON @@member2name USING SUBCONFIG TCPIP-2
* BUILD ON @@member3name USING SUBCONFIG TCPIP-3
* BUILD ON @@member4name USING SUBCONFIG TCPIP-4
 OPTION QUICKDSP SVMSTAT MAXCONN 1024 DIAG98 APPLMON
 SHARE RELATIVE 3000
 IUCV ALLOW
 IUCV ANY PRIORITY
 IUCV *CCS PRIORITY MSGLIMIT 255
 IUCV *VSWITCH MSGLIMIT 65535

SUBCONFIG TCPIP-1
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M01RES MR RTCPIP WTCPIP MTCPIP

SUBCONFIG TCPIP-2
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M02RES MR RTCPIP WTCPIP MTCPIP

*SUBCONFIG TCPIP-2
* LINK TCPMAINT 491 491 RR
* LINK TCPMAINT 492 492 RR
* LINK TCPMAINT 591 591 RR
* LINK TCPMAINT 592 592 RR
* LINK TCPMAINT 198 198 RR

*SUBCONFIG TCPIP-3
* LINK TCPMAINT 491 491 RR
* LINK TCPMAINT 492 492 RR
* LINK TCPMAINT 591 591 RR
* LINK TCPMAINT 592 592 RR
* LINK TCPMAINT 198 198 RR

*SUBCONFIG TCPIP-4
* LINK TCPMAINT 491 491 RR
* LINK TCPMAINT 492 492 RR
* LINK TCPMAINT 591 591 RR
* LINK TCPMAINT 592 592 RR
* LINK TCPMAINT 198 198 RR
```

7. Send a copy of the source directory (including passwords) to your virtual reader:

```
dirmaint user withpass
```

8. Receive the USER WITHPASS file to the PMAINT 2CC minidisk as USER DIRECT.

9. Create a new object directory on VMSYS01:

   ```
   dirmaint direct
   ```

   If the SSI cluster includes other existing members that are already using their DirMaint satellite service machines (for example, if you are adding member 4, and DIRMSAT2 and DIRMSAT3 are already active), this command will cause DirMaint to update the directory on those satellite systems as well as the DIRMAINT system.
10. If the SSI cluster includes other existing members and the DirMaint satellite service machines are not functional on those systems, create a new object directory on each of those members:
    a. Log on to the MAINT user ID on that member.
    b. Access the PMAINT 2CC and 551 minidisks.
    c. Create the new object directory:

       ```
       directxa user direct fm
       ```
11. If you are using RACF, modify the definition for the AUTOLOG2 user ID to change its password to NOLOG.

    After you IPL the new VMSYS02 system later in this procedure, you need to start the RACFVM virtual machine. When RACFVM is initialized, it issues an XAUTOLOG command for AUTOLOG2, which issues XAUTOLOG commands for the SVMs identified in its profile. However, you need to start RACFVM without logging on those SVMs, so you can do some other operations first. After you complete those operations you will modify the AUTOLOG2 user ID again to restore its password (so keep a record of the original password), and then autolog AUTOLOG2 to start the SVMs.

    Issue the following DIRMAINT commands from MAINT*vrm* on VMSYS01.

    a. Issue the following command to prevent the updated directory from being placed online. You should delay putting any directory changes online for the other cluster members until after the AUTOLOG2 change is rescinded. If you need to put changes online for a particular member, use DIRECTXA, not DIRMAINT DIRECT.

       ```
       dirmaint offline
       ```
    b. Issue the following command to modify the AUTOLOG2 definition:

       ```
       dirmaint foruser autolog2 setpw nolog
       ```
12. On system VMSYS01, create the object directory for VMSYS02:
    a. Detach M01RES and attach M02RES:

       ```
       detach 123
       attach rdev to * as 123
       ```
    b. Create the object directory:

       ```
       directxa user direct fm
       ```
    c. Detach M02RES:

       ```
       detach 123
       ```
    d. Re-link M01RES:

       ```
       link maint 123 123 m
       ```

Task 7: Update the System Configuration File

Update the common system configuration file to add the configuration information for the target member.

1. Access the PMAINT CF0 minidisk:

```
access cf0 fm
```

2. Make a copy of the SYSTEM CONFIG file named TEMP CONFIG on the CF0 minidisk and edit the copy.
3. Uncomment the SYSTEM_IDENTIFIER statement for member 2 and update the statement to specify LPAR name LP02 and system name VMSYS02.

```
/**********************************************************************/
/*                   System_Identifier Information                    */
/**********************************************************************/

   System_Identifier LPAR LP01 VMSYS01
   System_Identifier LPAR LP02 VMSYS02
 /*     System_Identifier LPAR @@LU-3 @@MEMSL3   */
 /*     System_Identifier LPAR @@LU-4 @@MEMSL4   */
```

4. In the SSI statement, uncomment the definition for member 2 and update it to specify system name VMSYS02.

   Note: When your SSI cluster is made up of 3 or more systems, it might be useful to organize the systems into relocation domains. These domains should be defined dynamically using the DEFINE RELODOMAIN command and added to the system configuration file using RELOCATION_DOMAIN statements. For more information, see "Using Relocation Domains" on page 754.

```
/**********************************************************************/
/*         SSI Statement                                              */
/**********************************************************************/

   SSI CLUSTERA PDR_Volume VMCOM1 ,
         Slot 1 VMSYS01,
         Slot 2 VMSYS02
/*       Slot 3 @@MEMSLOT3,  */
/*       Slot 4 @@MEMSLOT4   */
```

5. Uncomment the SYSTEM_RESIDENCE statement for member 2 and update it to specify system name VMSYS02 as the qualifier and M02RES as the volume label. Ensure that the checkpoint and warm start values for M02RES are identical to M01RES.

```
/**********************************************************************/
/*                Checkpoint and Warmstart Information                */
/**********************************************************************/

      VMSYS01:  System_Residence,
                Checkpoint  Volid M01RES   From CYL 21  For 9 ,
                Warmstart   Volid M01RES   From CYL 30  For 9
      VMSYS02:  System_Residence,
                Checkpoint  Volid M02RES   From CYL 21  For 9 ,
                Warmstart   Volid M02RES   From CYL 30  For 9
 /*@@MEMSLOT3:  System_Residence,                                  */
 /*             Checkpoint  Volid M03RES   From CYL 21  For 9 ,    */
 /*             Warmstart   Volid M03RES   From CYL 30  For 9      */
 /*@@MEMSLOT4:  System_Residence,                                  */
 /*             Checkpoint  Volid M04RES   From CYL 21  For 9 ,    */
 /*             Warmstart   Volid M04RES   From CYL 30  For 9      */
```

6. Update the CP_OWNED statements:

   a. In the system residence volume section, uncomment the statement for member 2 and update it to specify system name VMSYS02 as the qualifier and M02RES as the volume label.

```
/**********************************************************************/
/*                     CP_Owned Volume Statements                     */
/**********************************************************************/
/*                                                 RES   VOLUME      */
/**********************************************************************/

       VMSYS01: CP_Owned   Slot   1  M01RES
       VMSYS02: CP_Owned   Slot   1  M02RES
  /*@@MEMSLOT3: CP_Owned   Slot   1  M03RES  */
  /*@@MEMSLOT4: CP_Owned   Slot   1  M04RES  */

/**********************************************************************/
/*                                                COMMON VOLUME      */
/**********************************************************************/

                CP_Owned   Slot   5  VMCOM1
```

b. In the dump and spool volumes section, uncomment the statement for member 2 (slot 11 in this example) and specify M02S01 as the volume label.

   Important: Spool volumes must be common, not qualified.

```
/**********************************************************************/
/*                                          DUMP & SPOOL VOLUMES     */
/**********************************************************************/

                CP_Owned   Slot  10  M01S01
                CP_Owned   Slot  11  M02S01
  /*            CP_Owned   Slot  12  M03S01   */
  /*            CP_Owned   Slot  13  M04S01   */
```

c. In the paging volumes section, uncomment the section for member 2 and update it to specify system name VMSYS02 as the qualifier and M02P01 as the label of the paging volume.

```
/**********************************************************************/
/*                                           PAGE & TDISK VOLUMES     */
/**********************************************************************/

/**********************************************************************/
/* Page and Tdisk volumes for Member 1                                */
/**********************************************************************/

         VMSYS01: BEGIN
                CP_Owned   Slot 255  M01P01
         VMSYS01: END

/**********************************************************************/
/* Page and Tdisk volumes for Member 2                                */
/**********************************************************************/

         VMSYS02: BEGIN
                CP_Owned   Slot 255  M02P01
         VMSYS02: END

/**********************************************************************/
/* Page and Tdisk volumes for Member 3                                */
/**********************************************************************/

  /*@@MEMSLOT3: BEGIN                        */
  /*            CP_Owned   Slot 255  M03P01   */
  /*@@MEMSLOT3: END                          */

/**********************************************************************/
/* Page and Tdisk volumes for Member 4                                */
/**********************************************************************/

  /*@@MEMSLOT4: BEGIN                        */
  /*            CP_Owned   Slot 255  M04P01   */
  /*@@MEMSLOT4: END                          */
```

7. Add one or more USER_VOLUME_LIST statements to identify the work volumes and member-specific volumes for member 2 and qualify the statements for system name VMSYS02.

```
/**********************************************************************/
/*                           User_Volume_List                         */
/**********************************************************************/
/* COMMON user volumes                                                */
/**********************************************************************/

   User_Volume_List vrmRL1 vrmRL2 USRVL1

/**********************************************************************/
/* User volumes for Member 1                                          */
/**********************************************************************/

   VMSYS01: User_Volume_List M01PV1

/**********************************************************************/
/* User volumes for Member 2                                          */
/**********************************************************************/

   VMSYS02: User_Volume_List M02PV1
```

8. Add ACTIVATE ISLINK statements for the target member, and update the statements for the existing members, to identify the ISFC links from each member to the other members. Every member must have one direct ISFC logical link (consisting of multiple CTCA devices) defined to each of the other members. For example, if you are adding member 3, it must have links to members 1 and 2, member 1 must have links to members 2 and 3, and member 2 must have links to members 1 and 3. Use qualifiers to define the set of links for each member.

Notes:

- If the target member is being added to a second-level SSI cluster , the ACTIVATE ISLINK statements must specify virtual CTCAs defined by DEFINE and COUPLE statements in the profiles for the first-level user IDs.
- If the target member is being created second-level but the existing members are first-level, the ACTIVATE ISLINK statements for the target member must identify dedicated real CTCA devices, not virtual CTCAs, to enable the target member to communicate with the first-level members.

```
/**********************************************************************/
/*         Activate ISLINK statements                                 */
/**********************************************************************/
/**********************************************************************/
/* ISFC links for Member 1                                            */
/**********************************************************************/

   VMSYS01: ACTIVATE ISLINK rdev… NODE VMSYS02

/**********************************************************************/
/* ISFC links for Member 2                                            */
/**********************************************************************/

   VMSYS02: ACTIVATE ISLINK rdev… NODE VMSYS01
```

9. Adjust the real device numbers of the OSA devices on DEFINE VSWITCH statements, if needed.

```
/**********************************************************************/
/*         DEFINE VSWITCH statements                                  */
/**********************************************************************/

   DEFINE VSWITCH switchname RDEV rdev…
```

10. If the file includes a VMLAN statement with the MACPREFIX operand for each of the existing members to specify the prefix for system-assigned locally-administered MAC addresses, add a qualified statement for the target member. The MACPREFIX value must be different for each member. If the USERPREFIX option is included on the existing statements to specify the prefix for customer-assigned locally administered MAC addresses, include the USERPREFIX option on the new statement. The USERPREFIX value must be identical for all members.

```
VMSYS01: VMLAN MACPREFIX xxxxxx USERPREFIX yyyyyy
VMSYS02: VMLAN MACPREFIX zzzzzz USERPREFIX yyyyyy
```

11. After you finish editing the file, use the CPSYNTAX utility (available on the MAINT 193 minidisk) to check the syntax of the configuration statements for member VMSYS02.

```
cpsyntax temp config * (lpar lp02
```

Note: If z/VM is installed second level, the LPAR name field specifies the user ID of the virtual machine in which this system is running. The value of the LPAR operand must match the value specified on the SYSTEM_IDENTIFIER statement that was added above.

If there are other existing members in the cluster, repeat the command to check the syntax of the statements for each of those members. Correct all errors before proceeding to the next step. For the example in this chapter, do the following:

```
cpsyntax temp config * (lpar lp01
```

12. Prepare to use the revised system configuration file:

   a. Rename the current SYSTEM CONFIG file to BACKUP CONFIG.
   b. Rename the TEMP CONFIG file to SYSTEM CONFIG.

Task 8: Enable the Existing Members to Access the Target Member

All of the following commands can be issued on the current member (VMSYS01 in this example).

1. Add system VMSYS02 to the member list for the SSI cluster:

   ```
   set ssi slot 2 vmsys02
   ```

   Note: VMSYS02 will be in the Down state until the system is IPLed and joins the cluster.
2. Activate the ISFC links from the existing members to VMSYS02:
   a. Activate the links from VMSYS01 to VMSYS02:

      ```
      activate islink rdev [rdev…] node vmsys02
      ```

   b. If there are other active existing members in the cluster, activate the links from each member to VMSYS02:

      ```
      at sysname cmd activate islink rdev [rdev…] node vmsys02
      ```

3. Define the VMSYS02 spool volumes to CP (add the volumes to the CP-owned list) on the existing members and attach the volumes:
   a. Enable each device on member VMSYS01:

      ```
      vary online rdev
      ```

   b. If there are other active existing members in the cluster, enable each device on each member:

      ```
      at sysname cmd vary online rdev
      ```

   c. Define each volume to CP on member VMSYS01 and attach the volume. For example, to define volume M02S01 as slot 11 and attach it:

      ```
      define cpowned slot 11 m02s01
      attach rdev to system
      ```

      The attach process will automatically add the volume to the CP-owned list on the other active existing members and attach the volume to those members.
4. Log off MAINT*vrm*.

Task 9: IPL the Target Member

When you IPL the target member, the ACTIVATE ISLINK statements in the configuration file will trigger the process that joins the target member to the cluster.

1. Use the HMC to IPL the M02RES device.

   Note: To create a second level system, log on to the virtual machine in which VMSYS02 will be initialized and IPL the M02RES minidisk.
2. During the IPL, when prompted, specify options CLEAN NOAUTOLOG to bring up VMSYS02 without autologging any users except the system operator.

Task 10: Start the ESM Service Virtual Machine

If an ESM is installed, start the ESM service virtual machine.

1. For example, if you are using RACF:

   ```
   xautolog racfvm
   ```

   Note: RACFVM issues an XAUTOLOG command for the AUTOLOG2 user ID. However, you previously changed the password for AUTOLOG2 to NOLOG to prevent AUTOLOG2 from logging on and executing its profile to autolog other SVMs.

Task 11: Update the VMSES/E System-Level Product Inventory Table

Update the VMSES/E System-Level Product Inventory table (VM SYSPINV) to add VMSYS02 to the product records.

1. Log on to the MAINT*vrm* user ID.
2. Issue the following command:

   ```
   vmfupdat syspinv system vmsys02 vmsys01
   ```

Task 12: Update the CRR Server LU Name

Update the recovery SFS server with the new system name.

1. Log on to the recovery SFS server, VMSERVR. The default password is WD5JU8QP.

   To stop the server, enter:

   ```
   STOP
   ```

2. XEDIT the VMSERVR DMSPARMS file (located on the VMSERVR 191 minidisk). Change the LUNAME to match the target system name. Save the updated file.
3. Update the CRR log minidisk with the target system name:

   ```
   fileserv crrlog 306 307
   ```

   Note: 306 and 307 are the default minidisks for the recovery server defined in the VMSYSR POOLDEF file (the VDEVS assigned on the DDNAME=CRR*n* statements) on the VMSERVR 191 minidisk. If your recovery server is using minidisks other than the defaults, adjust the FILESERV command accordingly.
4. Restart the server and disconnect:

   ```
   profile
   #cp disc
   ```

Task 13: Build the Saved Segments and Named Saved Systems

Build the saved segments and named saved systems.

1. If you had previously configured the SYSTEM NETID file on VMSYS01, update the file on VMSYS02:

   a. Link and access the MAINT 190 minidisk:

      ```
      link to maint 190 as 190 m
      access 190 fm
      ```

   b. Edit the SYSTEM NETID file and add a record for the LPAR/processor ID and system name for this member. Make sure that file contains a record for every current member.

      ```
      cpuid1 VMSYS01 RSCS
      cpuid2 VMSYS02 RSCS
      ```

2. Start the VMSERVS, VMSERVU, and VMSERVR file pool servers on VMSYS02:

```
xautolog vmservs
⋮
xautolog vmservu
⋮
xautolog vmservr
```

3. Ensure that the product service file pool (VMPSFS, managed by server VMSERVP) is running. Ordinarily, this server should be running on one existing member of the cluster (for example, VMSYS01). Issue the following command:

```
xautolog vmservp
```

   You will receive either a message or a response:

   - If you receive the following message, the server is already logged on to another member of the cluster:

   ```
   HCP054E  Already logged on SYSTEM sysname
   ```

   - If you receive the following response, the server has been started on VMSYS02:

   ```
   Command accepted
   AUTO LOGON   ***     VMSERVP USERS = nnnnn
   ```

4. Save the CMS named saved system:

```
put2prod savecms
```

5. Build the segments defined in the system segments build list:

```
put2prod segments all
```

6. Build the GCS segment and named saved system:

```
service gcs bldnuc
put2prod
```

Task 14: Start the Service Virtual Machines

1. Update any remaining SVM configuration files that need to provide member-specific information for VMSYS02.
2. If you are using RACF, modify the definition for the AUTOLOG2 user ID to restore its password:
   - If you are using DirMaint:
     a. Start the DIRMSAT2 server:

     ```
     xautolog dirmsat2
     ```

     b. Change the password on the AUTOLOG2 definition:

     ```
     dirmaint foruser autolog2 setpw password
     ```

     c. Issue the following command to cancel the previous DIRMAINT OFFLINE command and put the updated directory online immediately for all the active members:

     ```
     dirmaint online immed
     ```

   - If you are not using DirMaint:
     a. Make a backup copy and then edit the common source directory (USER DIRECT).
     b. Change the password on the AUTOLOG2 definition.
     c. Create the new object directory:

```
directxa user direct fm
```

d. If directory changes for any other members of the cluster have been put online since you changed the AUTOLOG2 password to NOLOG, you need to create new object directories for those members as well to include this change:

   i) Log on to the MAINT user ID on that member.

   ii) Access the PMAINT 2CC and 551 minidisks.

   iii) Create the new object directory:

   ```
   directxa user direct fm
   ```

3. Update the SSL server certificate, if necessary.

   If you obtained a new single server certificate for VMSYS02, see *z/VM: TCP/IP User's Guide* for information on how to update the key database on VMSYS02 with the new certificate information.

4. Start the system startup virtual machine to autolog the SVMs:

   - If you are using RACF:

     ```
     xautolog autolog2
     ```

   - If you are not using RACF:

     ```
     xautolog autolog1
     ```

Task 15: Test and Verify

1. Verify that everything is working as it should on the new member.
2. After you verify that everything is working, it is safe to make updates to the user directory.

   Reminder: Whenever you change the common source directory, create a new object directory for each member of the cluster.

3. The cloning procedure is now completed and you can apply service to the members of the cluster, if needed.

Chapter 35. Adding Members to a 4-Member SSI Cluster

This topic includes information about the additional planning that is needed to migrate to an SSI cluster with more than 4 members. For information about defining a new member, see Chapter 34, "Adding a Member to a z/VM SSI Cluster by Cloning an Existing Member," on page 785.

Before adding 1 to 4 more members to a 4-member SSI cluster, you will need to perform the following tasks:

- If you plan to follow the migration procedure described in "Installing this support and preparing for more than 4 members" on page 811, verify that the PTF for APAR VM66462 was previously installed on all current members of the SSI cluster. This APAR is not required on all members at this time if the support code is being installed now, but no members are being changed to an 8-member cluster (SSI_CONTROLS SPOOL_MEMBERS is set to 8) at this time. This APAR is required on all members when the change to an 8-member cluster is being enabled on any member.
- Verify that no user has more than 1,249 spool files.
- Use the PUT2PROD EXEC to install this support on each existing member.
- Add the new SSI_CONTROLS system configuration statement when it is time to change to a greater number of members.
- Update the DIRECTORY statement in the CP directory.
- IPL each member after the changes to the SYSTEM CONFIG file are made.

Installing this support without preparing for more than 4 members

This support can be installed using the usual methods in a non-SSI environment or if there is no intention to grow the SSI cluster to more than 4 members in the near term. However, another IPL of each member will be required in the future to bring up a 5th member or more. If the plan is to run more than 4 members in the cluster soon, use the procedure described in "Installing this support and preparing for more than 4 members" on page 811 to avoid unnecessary IPLs. Installing without preparing for an 8-member SSI cluster does not require pre-installation of APAR VM66462 on all current members. It is a prerequisite on all of the members only when one of the members will be changed over to using the 8-member spool ID assignment algorithm.

1. Install the z/VM 7.3 release code.
2. Do not make any changes to the CP directory or the SYSTEM CONFIG file at this time.
3. This service can be installed one member at a time.

Sometime in the future, when more than 4 members are needed and all existing members have the 8-member SSI support installed, continue with step "4" on page 812 in "Installing this support and preparing for more than 4 members" on page 811. After this support is running on the system, you will notice that the output of the QUERY SSI command displays 8-member slots.

Installing this support and preparing for more than 4 members

1. Verify that the APAR VM66462 was installed previously and is running on all current members of the SSI cluster.
2. Run the SFCOUNTR utility with the default parameter on each member to verify that there are no users with more than the allowed number of spool files when in an 8-member SSI. The default value is 1124 (90% of the 1,249 file limit) to allow a safety margin for the migration scenario. However, you can specify any whole number as input. Any users that are multiconfiguration virtual machines (IDENTITY) can be ignored. SFCOUNTR automatically skips any IDENTITY users that are currently

logged on. Prune spool files from USER virtual machines with too many files so that the number they own is below the limit of 1,249.

3. Install the z/VM 7.3 release code.
4. To prepare for IPLing each member:
   a. Run the SFCOUNTR utility as described in step “2” on page 811. Prune spool files from USER virtual machines with too many files so that the number they own is below the limit of 1,249. If this is not done, a prompt will be received during the IPL (step “5” on page 812) asking whether excess files can be purged.
   b. Add the new SSI_CONTROLS statement to SYSTEM CONFIG to set the SPID assignment switch to the 8-member SPID assignment algorithm. If you are not upgrading all existing members now, use statement qualifiers or BEGIN and END statements to make this statement applicable only to the member or members that are being upgraded now or that are upgraded already.

      Note: Do not add slots 5 to 8 to the SSI statement at this point. This should be done only after all members have been upgraded because if this is done now, the IPL of any member that does not have the new support installed and the new SSI_CONTROLS statement set to 8 members for all members will fail.
5. After pruning spool files from USERs and re-running the SFCOUNTR utility if needed, re-IPL the member that is being upgraded. During the IPL, this member's SPIDs will be adjusted to the requirements of the new SPID assignment algorithm and new SPIDs assigned will follow those requirements. A WARM start will work and no files will be lost unless permission to purge excess files from USERs is given, as described in step “4” on page 812.
6. Repeat this process (steps “2” on page 811 to “5” on page 812) for all of the other existing members of the SSI cluster. Members that have been upgraded and members that have not been upgraded can coexist in the cluster. However, a disabled wait state 9052 is loaded if a member name is added to slot 5, 6, 7, or 8 before all existing members have been upgraded to the new support and switched to the 8-member SPID assignment algorithm.
7. A member can be IPLed back and forth between CPLOAD modules with and without the support until a member is defined in slot 5, 6, 7, or 8. Doing this might require changing the statement qualifiers or BEGIN and END statements on the SSI_CONTROLS statement in SYSTEM CONFIG because a module without this support will flag these statements as an error. Changing between the 4- and 8-member SPID assignment algorithms is also allowed until a member is defined in slot 5, 6, 7, or 8.
8. After all existing members are enabled for an 8-member cluster, use the following process for adding a new member to the SSI cluster. This includes the following steps:
   a. Add whichever members are needed in slots 5 to 8 on the SSI statement in the SYSTEM CONFIG file. To add the new members to the configuration dynamically, issue:

      ```
      SET SSI SLOT n sys_name
      ```

      You will receive error message HCP6650E from this command if all existing members of the cluster have not yet been switched over to the 8-member SPID assignment algorithm.
   b. Add the necessary volids to one or more DIRECTORY SSI statements in the CP directory and run the DIRECTXA utility.
   c. IPL the new members as desired.

      Note: After any of the members in slots 5 to 8 are defined as other than AVAILABLE, none of the members can be backed off to use the 4-member SPID assignment algorithm. If that is attempted, message HCP6654E is displayed and disabled wait state 1682 is loaded on the system being IPLed using the 4-member algorithm. This means all members must be running the new support and have the 8-member SPID assignment switch set.

   For more detailed steps of what must be done to add an SSI member, see Chapter 34, “Adding a Member to a z/VM SSI Cluster by Cloning an Existing Member,” on page 785.

Migration error cases

The following errors could be encountered during the process of setting up a 5th member of an SSI cluster. These errors are caused by doing migration steps in the wrong order. SSI statement syntax errors (for example) are not included here.

Old CPLOAD finds an SSI SLOT 5|6|7|8 statement

When existing code encounters an SSI statement with slots 5 to 8 coded, it will display some error messages and load a wait 1682. For example:

```
HCPZPM6700E File SYSTEM CONFIG, records 71 through 72:
HCPZPM002E Invalid operand - 5
HCPZSF1682W Errors found while processing SSI statement.
```

For a second-level z/VM system, the following message is also displayed along with the wait state.

```
HCPGIR450W CP entered; disabled wait ... 00001682
```

Old CPLOAD finds a new SSI_CONTROLS statement

When existing code encounters an SSI_CONTROLS statement, it will react two different ways depending on what was specified on the last TOLERATE_CONFIG_ERRORS statement that was processed. For example:

```
If no TOLERATE_CONFIG_ERRORS or TOLERATE_CONFIG_ERRORS YES:
10:11:31 HCPZPM6700E File SYSTEM CONFIG, record 173:
10:11:31 HCPZPM6701E Invalid system configuration file statement - SSI_CONTROLS
...
and CP initialization continues.

If TOLERATE_CONFIG_ERRORS NO:
10:09:31 HCPZPM6700E File SYSTEM CONFIG, record 169:
10:09:31 HCPZPM6701E Invalid system configuration file statement - SSI_CONTROLS
...
10:09:31 HCPASK6717A
10:09:31 HCPASK6717A One or more errors were encountered in processing
10:09:31 HCPASK6717A sections of the system configuration file that were
10:09:31 HCPASK6717A marked not to tolerate errors.
10:09:31 HCPASK6717A
10:09:31 HCPASK6717A To ignore the errors and continue normally, enter GO.
10:09:31 HCPASK6717A To continue with the IPL without autologging any users,
10:09:31 HCPASK6717A enter NOAUTOLOG.
10:09:31 HCPASK6717A To abort the IPL, enter STOP.
```

SET SSI SLOT 5|6|7|8 command

The SET SSI SLOT command cannot be used to define slot 5, 6, 7, or 8 until all joined members are using the 8-member SPID assignment algorithm. The SET SSI command rejects this case when one of the following is true:

1. The member on which the command was issued is using the 4-member SPID assignment algorithm. The command is rejected with error message HCP6650E if this is attempted.
2. A remote member votes NO to the SET SSI command synchronization if it is using the 4-member SPID assignment algorithm. The command is rejected with existing message HCP1649E in this case.

Member tries to join with slot 5, 6, 7, or 8 defined, but its SSI_CONTROLS SPOOL_MEMBERS value is too low

SSI_CONTROLS SPOOL_MEMBERS must be set to 8 if any slots higher than slot number 4 are to be defined. This error leads to message HCP6654E followed by wait state 1682.

Member 5, 6, 7, or 8 tries to join (IPLs) and one or more of the already-joined members are still using the 4-member SPID assignment algorithm

Members in slots 5 to 8 cannot be IPLed until all joined members are using the 8-member SPID assignment algorithm. This will be rejected when an attempt is made to define slot 5 using the SET SSI command. If an attempt is made to IPL the member in slot 5, the other members will vote NO to the join, the IPL will fail, and message HCP9052W (`PDR and defined configuration do not match`) is displayed, and wait state 9052 is loaded.

IPL with SSI_CONTROLS SPOOL_MEMBERS 8 and not all members have APAR VM66462

To avoid problems created by an open-for-read COPY SPFBK whose SPID needs to be changed, APAR VM66462 must be on all members of the SSI cluster before any member is IPLed with SSI_CONTROLS SPOOL_MEMBERS set to 8. Also, a member without APAR VM66462 cannot join a cluster that has one or more members who have previously been IPLed with SSI_CONTROLS SPOOL_MEMBERS set to 8. If this requirement is not met, message HCP1922W is issued and wait state 1682 is loaded and IPL does not complete.

Back-off to 4-member SPID algorithm

After a member is defined in slot 5, 6, 7, or 8 via the SET SSI SLOT command or the SYSTEM CONFIG SSI statement, no member can be IPLed using the 4-member SPID assignment algorithm. Any attempt to do this results in that IPLing member going into disabled wait state 9052. The remedy is to update the configuration file from another active member to indicate the 8-member SPID algorithm for this member, and then try to IPL.

Chapter 36. Moving a Second-Level z/VM SSI Cluster to First-Level

Use this procedure to move a second-level z/VM SSI cluster (where the members are IPLed in virtual machines) to first-level (where the members are IPLed in logical partitions).

Before you begin the procedure, see "Assumptions for Using This Procedure" on page 815.

In this procedure you will complete the following tasks:

"Task 1: Update the System Configuration File" on page 815
"Task 2: Shut Down the Second-Level SSI Cluster and IPL the Members First-Level" on page 817
"Task 3: Update the z/VM System Where the SSI Cluster Was Installed Second-Level" on page 818

Assumptions for Using This Procedure

This procedure is based upon the following assumptions:

- The SSI cluster was installed second-level by following one of the SSI installation procedures documented in *z/VM: Installation Guide*.
- The second-level cluster was installed to real DASD volumes.
- Any member-specific minidisks for customer-defined service machines and servers have been defined on additional real DASD volumes.
- DASD volume labels and addresses will not be changed.
- System names of the members and other cluster data will not be changed.
- The logical partitions in which the members will be IPLed have been defined and configured with access to the same devices that are available in the second-level environment.
- A common system configuration file is used for all members of the cluster.

The examples in this procedure show a cluster configuration consisting of four members. The following values are used:

- System names of the members: VMSYS01, VMSYS02, VMSYS03, and VMSYS04.
- Virtual machines in which the members are IPLed: USER1, USER2, USER3, and USER4.
- Corresponding logical partitions: LP1, LP2, LP3, and LP4.

Task 1: Update the System Configuration File

Update the system configuration file (SYSTEM CONFIG) to specify the information for first-level execution of the SSI cluster.

Note: For simplicity, this procedure does not show the system responses.

1. Log on to a MAINT*vrm* user ID on any member of the cluster.
2. Access the PMAINT CF0 minidisk:

```
access cf0 fm
```

3. Make a copy of the system configuration file with a different name (such as 1STLVL CONFIG) on the CF0 minidisk and make the changes to the copy.
4. Define SYSTEM_IDENTIFIER statements that map the members of the cluster to the logical partitions.

   In the existing statement for each member, the LPAR name field specifies the first-level user ID of the virtual machine in which the member is IPLed.

- If you will never run the SSI cluster second-level again, update the LPAR name field to specify the name of the logical partition in which the member will be IPLed.

  Before (Second-Level):

  ```
  /**********************************************************************/
  /*                    System_Identifier Information                   */
  /**********************************************************************/

   System_Identifier LPAR USER1 VMSYS01
   System_Identifier LPAR USER2 VMSYS02
   System_Identifier LPAR USER3 VMSYS03
   System_Identifier LPAR USER4 VMSYS04
  ```

  After (First-Level):

  ```
  /**********************************************************************/
  /*                    System_Identifier Information                   */
  /**********************************************************************/

   System_Identifier LPAR LP1 VMSYS01
   System_Identifier LPAR LP2 VMSYS02
   System_Identifier LPAR LP3 VMSYS03
   System_Identifier LPAR LP4 VMSYS04
  ```

- If you intend to be able to switch between first-level and second-level execution of the SSI cluster (such as for disaster recovery), keep the second-level statements and add a new set of statements for first-level.

  First-Level or Second-Level Execution:

  ```
  /**********************************************************************/
  /*                    System_Identifier Information                   */
  /**********************************************************************/

   System_Identifier LPAR USER1 VMSYS01
   System_Identifier LPAR USER2 VMSYS02
   System_Identifier LPAR USER3 VMSYS03
   System_Identifier LPAR USER4 VMSYS04
   System_Identifier LPAR LP1 VMSYS01
   System_Identifier LPAR LP2 VMSYS02
   System_Identifier LPAR LP3 VMSYS03
   System_Identifier LPAR LP4 VMSYS04
  ```

5. Update the ACTIVATE ISLINK statements to specify the real CTCA devices for the ISFC links from each member to the other members.

   The existing ACTIVATE ISLINK statements specify the virtual CTCAs defined by DEFINE and COUPLE statements in the profiles for the first-level user IDs.

   The following example shows devices for two ISFC links between members (therefore six devices for each member) and shows the default virtual device numbers defined by the second-level installation procedure. That is, A2B1 and A2B2 are the devices for the links from VMSYS01 to VMSYS02, A2C1 and A2C2 are the devices for the links from VMSYS01 to VMSYS03, and so on.

   Change the virtual device numbers to the real device numbers. That is, *rdev1* and *rdev2* are the real devices for the links from VMSYS01 to VMSYS02, *rdev3* and *rdev4* are the real devices for the links from VMSYS01 to VMSYS03, and so on.

   Before:

```
/**********************************************************************/
/* Activate ISLINK statements */
/**********************************************************************/

VMSYS01: ACTIVATE ISLINK A2B1 A2B2 NODE VMSYS02
VMSYS01: ACTIVATE ISLINK A2C1 A2C2 NODE VMSYS03
VMSYS01: ACTIVATE ISLINK A2D1 A2D2 NODE VMSYS04
VMSYS02: ACTIVATE ISLINK B2A1 B2A2 NODE VMSYS01
VMSYS02: ACTIVATE ISLINK B2C1 B2C2 NODE VMSYS03
VMSYS02: ACTIVATE ISLINK B2D1 B2D2 NODE VMSYS04
VMSYS03: ACTIVATE ISLINK C2A1 C2A2 NODE VMSYS01
VMSYS03: ACTIVATE ISLINK C2B1 C2B2 NODE VMSYS02
VMSYS03: ACTIVATE ISLINK C2D1 C2D2 NODE VMSYS04
VMSYS04: ACTIVATE ISLINK D2A1 D2A2 NODE VMSYS01
VMSYS04: ACTIVATE ISLINK D2B1 D2B2 NODE VMSYS02
VMSYS04: ACTIVATE ISLINK D2C1 D2C2 NODE VMSYS03
```

After:

```
/**********************************************************************/
/* Activate ISLINK statements */
/**********************************************************************/

VMSYS01: ACTIVATE ISLINK rdev1 rdev2 NODE VMSYS02
VMSYS01: ACTIVATE ISLINK rdev3 rdev4 NODE VMSYS03
VMSYS01: ACTIVATE ISLINK rdev5 rdev6 NODE VMSYS04
VMSYS02: ACTIVATE ISLINK rdev7 rdev8 NODE VMSYS01
VMSYS02: ACTIVATE ISLINK rdev9 rdev10 NODE VMSYS03
VMSYS02: ACTIVATE ISLINK rdev11 rdev12 NODE VMSYS04
VMSYS03: ACTIVATE ISLINK rdev13 rdev14 NODE VMSYS01
VMSYS03: ACTIVATE ISLINK rdev15 rdev16 NODE VMSYS02
VMSYS03: ACTIVATE ISLINK rdev17 rdev18 NODE VMSYS04
VMSYS04: ACTIVATE ISLINK rdev19 rdev20 NODE VMSYS01
VMSYS04: ACTIVATE ISLINK rdev21 rdev22 NODE VMSYS02
VMSYS04: ACTIVATE ISLINK rdev23 rdev24 NODE VMSYS03
```

6. If the hardware features and architectural enhancements available to the members of the SSI cluster at first level differ from those available to the members at second level, you might want to adjust the relocation domain definitions and the assignment of guests to those domains. For example, a guest might need a particular type of cryptographic feature or High Performance FICON for IBM Z (zHPF). A feature is available for use by a guest only if all members of its relocation domain have the feature and support it. For additional details see "Using Relocation Domains" on page 754.
7. File your changes to 1STLVL CONFIG.
8. Use the CPSYNTAX command to check the syntax of the configuration statements in 1STLVL CONFIG.

Task 2: Shut Down the Second-Level SSI Cluster and IPL the Members First-Level

You are now ready to change the execution of the SSI cluster from second-level to first-level.

1. Shut down all the members of the cluster except the member where you are logged on as MAINT*vrm*.
2. Rename the current SYSTEM CONFIG file to a different name (such as 2NDLVL CONFIG) and rename the 1STLVL CONFIG file to SYSTEM CONFIG.
3. Shut down the member where you are logged on.
4. IPL each member of the cluster in its corresponding logical partition.

   If you encounter a problem bringing up the cluster first-level, you can use the old system configuration file (currently named 2NDLVL CONFIG) to bring up the cluster second-level. Once the cluster is successfully running first-level, you can discard the old file.

Attention: The members of the SSI cluster must be run either all first-level or all second-level. Do not attempt to run some members first-level and some second-level.

Task 3: Update the z/VM System Where the SSI Cluster Was Installed Second-Level

When you are satisfied that the SSI cluster is successfully running first-level, perform one of the following tasks on the z/VM system where the SSI cluster was installed second-level.

- If you will never run the SSI cluster second-level again:
  1. If USER1, USER2, USER3, and USER4 are still logged on, log them off.
  2. Remove the virtual machine definitions for those user IDs from the source directory.
- If you intend to be able to switch between first-level and second-level execution of the SSI cluster, update the profiles for USER1, USER2, USER3, and USER4 and change the virtual device numbers of the CTCAs on the DEFINE and COUPLE statements to match the first-level real device numbers (*vdev1* = *rdev1*, and so on).

  The following example shows the changes in the profile for USER1.

Before:

```
'DEFINE CTCA A2B1'
'DEFINE CTCA A2B2'
'DEFINE CTCA A2C1'
'DEFINE CTCA A2C2'
'DEFINE CTCA A2D1'
'DEFINE CTCA A2D2'
'COUPLE A2B1 TO USER2 B2A1'
'COUPLE A2B2 TO USER2 B2A2'
'COUPLE A2C1 TO USER3 C2A1'
'COUPLE A2C2 TO USER3 C2A2'
'COUPLE A2D1 TO USER4 D2A1'
'COUPLE A2D2 TO USER4 D2A2'
```

After:

```
'DEFINE CTCA vdev1'
'DEFINE CTCA vdev2'
'DEFINE CTCA vdev3'
'DEFINE CTCA vdev4'
'DEFINE CTCA vdev5'
'DEFINE CTCA vdev6'
'COUPLE vdev1 TO USER2 vdev7'
'COUPLE vdev2 TO USER2 vdev8'
'COUPLE vdev3 TO USER3 vdev13'
'COUPLE vdev4 TO USER3 vdev14'
'COUPLE vdev5 TO USER4 vdev19'
'COUPLE vdev6 TO USER4 vdev20'
```

Note: Additional changes to the virtual machine configuration or profile might be necessary if any of the required virtual device numbers are already in use on this user ID.

Chapter 37. Decommissioning a Member of a z/VM SSI Cluster

Use this procedure to remove a member from a z/VM SSI cluster and permanently shut down that z/VM system. This process is referred to as *decommissioning* the member.

Note: If a member of the cluster is simply shut down, it can be reIPLed and can rejoin the cluster. A decommissioned member cannot be reIPLed and cannot rejoin the cluster; it is permanently removed.

See "Before You Begin the Decommissioning Procedure" on page 821 for assumptions, requirements, and preparations.

In this procedure you will complete the following tasks:

Before You Begin the Decommissioning Procedure

Note the assumptions for this procedure. Make sure that you meet all of the requirements and complete the preparations.

Assumptions:

- The member being decommissioned is **not** the last member of the SSI cluster. This procedure includes examples for decommissioning member 3 of a three-member SSI cluster, but the tasks apply to decommissioning any member except the last member of the cluster.

  To decommission the last member of an SSI cluster, bring up a z/VM starter system on that member and format all of the member's DASD with new labels. For information about using a starter system, see *z/VM: Installation Guide*.
- A common system configuration file and a common source directory are used for the SSI cluster.
- For the member being decommissioned, object directory extents are allocated only on the system residence volume.

Requirements:

- When decommissioned, the z/VM system must be permanently shut down.

  Note: To reuse the LPAR for another z/VM system, reinstall z/VM and use different DASD volume labels.
- If you plan to relocate guests from the member being decommissioned to other members of the cluster, ensure that the relocation candidates and the target members meet the eligibility requirements. For more information, see Chapter 31, "Preparing for Guest Relocations in a z/VM SSI Cluster," on page 753.

Preparations:

- Prepare a worksheet with the DASD volume labels and other information about the member being decommissioned. The following values are used in this procedure:
  - SSI cluster name = CLUSTERA
  - Logical partition name = LP03

    Note: At second level, this is the user ID of the virtual machine in which the member system is running.
  - System ID of the member being decommissioned = VMSYS03
  - Labels of the DASD volumes associated with this system.

    The examples in this procedure use the default volume labels created by the installation procedure for a multimember SSI cluster. The naming convention is M0*mvnn* for most volumes, where *m* is the number of this member in the SSI member list (on the SSI configuration statement) and *vnn* identifies the volume type and number (for example, P01 for the first paging volume).
    - System residence volume = M03RES
    - Volume for paging = M03P01
    - Volume for spool files = M03S01
    - Volume for temporary disks (if used) = M03T01
    - Member-specific user volume = M03PV1

Note: In an SSI cluster in which members have different releases installed, if the member being decommissioned is the last member on which the highest release level product is installed, the cross-system highest release level programs residing on the SSI system common disk (PMAINT 551, by default) will remain at that highest release level.

Task 1: Log on to the Member Being Decommissioned

Note: For simplicity, this procedure does not show all of the system responses.

1. Log on to the MAINT*vrm* user ID for the release level on VMSYS03.

Task 2: Update the DirMaint Configuration (If Required)

If DirMaint is not being used in the SSI cluster, skip this task and go to "Task 3: Move the Workload to Other Members" on page 823.

If DirMaint is installed and enabled in the SSI cluster, remove the member being decommissioned from the DirMaint configuration:

- If the DIRMAINT server is logged on to the member being decommissioned, log off DIRMAINT on that member and log on DIRMAINT on another member.

  Attention: Do not use the VMRELOCATE command to relocate the DIRMAINT virtual machine.
- Remove the DIRMSAT and DATAMOVE servers for the member being decommissioned (DIRMSAT*n* and DATAMOV*n*). Also, if you moved the DIRMAINT server, update the configuration to reflect the new location.

1. If DIRMAINT is running on VMSYS03, shut down the server:

```
dirmaint shutdown

DVHxxx2193I A shutdown command has been issued by MAINTvrm from VMSYS03
DVHxxx2198A The DIRMAINT service machine is logging off
```

2. On another member of the SSI cluster, log on the DIRMAINT server and update the DirMaint configuration:

Note: In this example, DIRMAINT is being moved from VMSYS03 to VMSYS01.

a. Log on to the MAINT user ID.

b. Log on the DIRMAINT virtual machine:

```
xautolog dirmaint
```

c. Update the CONFIGSS DATADVH file:

i) Remove the SATELLITE_SERVER statement for VMSYS03 (for example, DIRMSAT3).

Note: If you are using an external security manager, such as RACF, the satellite for VMSYS03 also must be removed from the ESM configuration. See the external security manager considerations in *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*.

ii) Remove the DATAMOVE_MACHINE statement for VMSYS03 (for example, DATAMOV3).

iii) Reload the data tables from the CONFIG* DATADVH disk files:

```
dirmaint rlddata
```

3. Update the EXTENT CONTROL file to remove the minidisk allocation data for member VMSYS03.

a. Send a copy of the EXTENT CONTROL file to your virtual reader:

```
dirmaint send extent control
```

b. Receive the EXTENT CONTROL file and edit it.

i) If the :REGIONS. section includes any entries for VMSYS03 volumes, remove them.

ii) Update the :SSI_VOLUMES. section to remove the entries for VMSYS03 volumes.

iii) If the :DEFAULTS. section includes an entry for a DASD device type and model used only on VMSYS03, remove the entry.

c. Replace the updated EXTENT CONTROL file and reload the data:

```
dirmaint file extent control
dirmaint rldextn
```

Task 3: Move the Workload to Other Members

Move all workload from the member being decommissioned to the other members of the cluster.

1. Make sure all users are logged off VMSYS03.
2. If there are servers and service machines on VMSYS03 that you will need after VMSYS03 is decommissioned, and those virtual machines can be shut down, log them off VMSYS03 and log them on to another member of the cluster.
3. If there are guests running on VMSYS03 that you will need after VMSYS03 is decommissioned, and you do not want to shut down those guests, use the VMRELOCATE command to relocate the virtual machines to other members of the cluster. For example, to relocate the LINX7 virtual machine from member VMSYS03 to member VMSYS02:

```
vmrelocate move user linx7 to vmsys02

Relocation of LINX7 from VMSYS03 to VMSYS02 started
⋮
User LINX7 has been relocated from VMSYS03 to VMSYS02
```

Note: To relocate a virtual machine to another member of the SSI cluster, the relocation candidate and the destination system must meet the eligibility requirements. For more information, see Chapter 31, "Preparing for Guest Relocations in a z/VM SSI Cluster," on page 753.

Task 4: Purge the Decommissioned Member's Spool Files

Purge the standard spool files on the member being decommissioned. This will also remove the shadowed spool files on the other members.

1. (Optional) If you want to save the spool files, you can use the SPXTAPE command to dump them to tape and purge them after they are dumped:

   ```
   vary online rdev
   attach rdev to * as vdev
   ⋮
   spxtape dump vdev std all purge
   ```

2. Purge the spool files:

   ```
   purge force system ur all
   ```

Task 5: Update the Profile for the System Startup Virtual Machine

If the profile for the system startup virtual machine (AUTOLOG1) contains any logic to conditionally process certain XAUTOLOG commands for the member being decommissioned, remove those statements and processing logic.

1. Link (multiple-write) and access the AUTOLOG1 191 minidisk:

   ```
   link to autolog1 191 as vdev m
   access vdev fm
   ```

   Note: If you are using RACF, make the changes to the AUTOLOG2 virtual machine instead of AUTOLOG1.

2. Edit the PROFILE EXEC file and remove the XAUTOLOG commands and processing logic that apply only to member VMSYS03.

Task 6: Shut Down the Decommissioned Member and Log on to Another Member

1. Shut down system VMSYS03:

   ```
   shutdown
   ```

2. Log on to the MAINT*vrm* user ID on another member of the SSI cluster, for example VMSYS02.

Task 7: Remove the Member from the Member List

Remove the decommissioned member from the member list for the SSI cluster.

1. To remove system VMSYS03 from the member list, mark slot 3 as available:

   ```
   set ssi slot 3 available
   ```

Task 8: Update the System Configuration File

Update the common system configuration file to remove configuration information for the decommissioned member.

1. Access the PMAINT CF0 minidisk:

   ```
   access cf0 fm
   ```

2. Make a copy of the SYSTEM CONFIG file called TEMP CONFIG on the CF0 minidisk and edit the copy.

3. Remove the SYSTEM_IDENTIFIER statement for VMSYS03..

   Before:

```
/**********************************************************************/
/*                  System_Identifier Information                     */
/**********************************************************************/

   System_Identifier LPAR LP01 VMSYS01
   System_Identifier LPAR LP02 VMSYS02
   System_Identifier LPAR LP03 VMSYS03
```

   After:

```
/**********************************************************************/
/*                  System_Identifier Information                     */
/**********************************************************************/

   System_Identifier LPAR LP01 VMSYS01
   System_Identifier LPAR LP02 VMSYS02
```

4. Update the SSI statement to remove VMSYS03 from slot 3 and mark the slot as AVAILABLE.

   Before:

```
/**********************************************************************/
/*         SSI Statement                                              */
/**********************************************************************/

   SSI CLUSTERA PDR_Volume VMCOM1 ,
         Slot 1 VMSYS01,
         Slot 2 VMSYS02,
         Slot 3 VMSYS03,
         Slot 4 AVAILABLE
```

   After:

```
/**********************************************************************/
/*         SSI Statement                                              */
/**********************************************************************/

   SSI CLUSTERA PDR_Volume VMCOM1 ,
         Slot 1 VMSYS01,
         Slot 2 VMSYS02,
         Slot 3 AVAILABLE,
         Slot 4 AVAILABLE
```

5. Remove VMSYS03 from any RELOCATION_DOMAIN statements.

   Before:

```
/**********************************************************************/
/*                  Relocation Domains                                */
/**********************************************************************/

   Relocation_Domain DOMAIN1 Members VMSYS01 VMSYS02
   Relocation_Domain DOMAIN2 Members VMSYS02
VMSYS03
```

After:

```
/**********************************************************************/
/*                     Relocation Domains                             */
/**********************************************************************/

   Relocation_Domain DOMAIN1 Members VMSYS01 VMSYS02
   Relocation_Domain DOMAIN2 Members
VMSYS02
```

6. Remove the SYSTEM_RESIDENCE statement for VMSYS03.

   Before:

```
/**********************************************************************/
/*                  Checkpoint and Warmstart Information              */
/**********************************************************************/

   VMSYS01:  System_Residence,
             Checkpoint  Volid M01RES   From CYL 21  For 9 ,
             Warmstart   Volid M01RES   From CYL 30  For 9
   VMSYS02:  System_Residence,
             Checkpoint  Volid M02RES   From CYL 21  For 9 ,
             Warmstart   Volid M02RES   From CYL 30  For 9
   VMSYS03:  System_Residence,
             Checkpoint  Volid M03RES   From CYL 21  For 9 ,
             Warmstart   Volid M03RES   From CYL 30  For 9
```

   After:

```
/**********************************************************************/
/*                  Checkpoint and Warmstart Information              */
/**********************************************************************/

   VMSYS01:  System_Residence,
             Checkpoint  Volid M01RES   From CYL 21  For 9 ,
             Warmstart   Volid M01RES   From CYL 30  For 9
   VMSYS02:  System_Residence,
             Checkpoint  Volid M02RES   From CYL 21  For 9 ,
             Warmstart   Volid M02RES   From CYL 30  For 9
```

7. Update the CP_OWNED statements to remove VMSYS03.

   a. System residence volume.

      Before:

```
/**********************************************************************/
/*                     CP_Owned Volume Statements                     */
/**********************************************************************/
/*                                                  RES   VOLUME      */
/**********************************************************************/

   VMSYS01: CP_Owned   Slot   001  M01RES
   VMSYS02: CP_Owned   Slot   001  M02RES
   VMSYS03: CP_Owned   Slot   001  M03RES

/**********************************************************************/
/*                                                 COMMON VOLUME      */
/**********************************************************************/

   CP_Owned   Slot   005  VMCOM1
```

After:

```
/**********************************************************************/
/*                      CP_Owned Volume Statements                    */
/**********************************************************************/
/*                                                     RES   VOLUME   */
/**********************************************************************/

   VMSYS01: CP_Owned   Slot   001  M01RES
   VMSYS02: CP_Owned   Slot   001  M02RES

/**********************************************************************/
/*                                                    COMMON VOLUME   */
/**********************************************************************/

   CP_Owned   Slot   005  VMCOM1
```

b. Dump and spool volumes.

Record the slot numbers and labels of the spool volumes. You will need this information in "Task 10: Free the Decommissioned Member's DASD Volumes" on page 831.

Attention: Do not delete the spool volume slots; mark them as RESERVED.

Before:

```
/**********************************************************************/
/*                                             DUMP & SPOOL VOLUMES   */
/**********************************************************************/

   CP_Owned   Slot  010  M01S01
   CP_Owned   Slot  011  M02S01
   CP_Owned   Slot  012  M03S01
```

After:

```
/**********************************************************************/
/*                                             DUMP & SPOOL VOLUMES   */
/**********************************************************************/

   CP_Owned   Slot  010  M01S01
   CP_Owned   Slot  011  M02S01
   CP_Owned   Slot  012  RESERVED
```

c. Paging and temporary disk volumes.

Note: Although this example shows a temporary disk volume, this volume is not included in the default z/VM installation.

Before:

```
/**********************************************************************/
/*                                            PAGE & TDISK VOLUMES     */
/**********************************************************************/

/**********************************************************************/
/* Page and Tdisk volumes for Member 1                                */
/**********************************************************************/

   VMSYS01: BEGIN
            CP_Owned   Slot 254  M01T01
            CP_Owned   Slot 255  M01P01
   VMSYS01: END

/**********************************************************************/
/* Page and Tdisk volumes for Member 2                                */
/**********************************************************************/

   VMSYS02: BEGIN
            CP_Owned   Slot 254  M02T01
            CP_Owned   Slot 255  M02P01
   VMSYS02: END

/**********************************************************************/
/* Page and Tdisk volumes for Member 3                                */
/**********************************************************************/

   VMSYS03: BEGIN
            CP_Owned   Slot 254  M03T01
            CP_Owned   Slot 255  M03P01
   VMSYS03: END
```

After:

```
/**********************************************************************/
/*                                            PAGE & TDISK VOLUMES     */
/**********************************************************************/

/**********************************************************************/
/* Page and Tdisk volumes for Member 1                                */
/**********************************************************************/

   VMSYS01: BEGIN
            CP_Owned   Slot 254  M01T01
            CP_Owned   Slot 255  M01P01
   VMSYS01: END

/**********************************************************************/
/* Page and Tdisk volumes for Member 2                                */
/**********************************************************************/

   VMSYS02: BEGIN
            CP_Owned   Slot 254  M02T01
            CP_Owned   Slot 255  M02P01
   VMSYS02: END
```

8. Update the USER_VOLUME_LIST statements to remove the VMSYS03 work volume and member-specific user volume.

   Before:

```
/**********************************************************************/
/*                          User_Volume_List                         */
/**********************************************************************/
/* COMMON user volumes                                                */
/**********************************************************************/

   User_Volume_List vrmRL1 vrmRL2 USRVL1

/**********************************************************************/
/* User volumes for Member 1                                          */
/**********************************************************************/

   VMSYS01: User_Volume_List M01PV1

/**********************************************************************/
/* User volumes for Member 2                                          */
/**********************************************************************/

   VMSYS02: User_Volume_List M02PV1

/**********************************************************************/
/* User volumes for Member 3                                          */
/**********************************************************************/

   VMSYS03: User_Volume_List M03PV1
```

After:

```
/**********************************************************************/
/*                          User_Volume_List                         */
/**********************************************************************/
/* COMMON user volumes                                                */
/**********************************************************************/

   User_Volume_List vrmRL1 vrmRL2 USRVL1

/**********************************************************************/
/* User volumes for Member 1                                          */
/**********************************************************************/

   VMSYS01: User_Volume_List M01PV1

/**********************************************************************/
/* User volumes for Member 2                                          */
/**********************************************************************/

   VMSYS02: User_Volume_List M02PV1
```

9. Remove the ACTIVATE ISLINK statements for ISFC links from and to VMSYS03.

   Record the device numbers of the links from each member to VMSYS03. You will deactivate these devices in "Task 9: Deactivate the ISFC Links to the Decommissioned Member" on page 831.

   Before:

```
/**********************************************************************/
/*         Activate ISLINK statements                                 */
/**********************************************************************/
/**********************************************************************/
/* ISFC links for Member 1                                            */
/**********************************************************************/

   VMSYS01: Activate ISLINK rdev… Node VMSYS02
   VMSYS01: Activate ISLINK rdev… Node VMSYS03

/**********************************************************************/
/* ISFC links for Member 2                                            */
/**********************************************************************/

   VMSYS02: Activate ISLINK rdev… Node VMSYS01
   VMSYS02: Activate ISLINK rdev… Node VMSYS03

/**********************************************************************/
/* ISFC links for Member 3                                            */
/**********************************************************************/

   VMSYS03: Activate ISLINK rdev… Node VMSYS01
   VMSYS03: Activate ISLINK rdev… Node VMSYS02
```

After:

```
/**********************************************************************/
/*         Activate ISLINK statements                                 */
/**********************************************************************/
/**********************************************************************/
/* ISFC links for Member 1                                            */
/**********************************************************************/

   VMSYS01: Activate ISLINK rdev… Node VMSYS02

/**********************************************************************/
/* ISFC links for Member 2                                            */
/**********************************************************************/

   VMSYS02: Activate ISLINK rdev… Node VMSYS01
```

10. Adjust the real device numbers of the OSA devices on DEFINE VSWITCH statements, if needed.

```
/**********************************************************************/
/*         Define VSWITCH statements                                  */
/**********************************************************************/

   Define VSWITCH switchname RDEV rdev…
```

11. Remove all other statements or entries qualified (tagged) for system VMSYS03.
12. Issue the CPSYNTAX command to check the syntax of the configuration statements for each remaining member of the cluster to make sure you have not inadvertently caused syntax errors when removing the statements for VMSYS03. Correct all errors before proceeding to the next step.

```
cpsyntax temp config * (lpar lp01
```

```
cpsyntax temp config * (lpar lp02
```

Note: If z/VM is installed second level, the LPAR name field specifies the user ID of the virtual machine in which this system is running. The value of the LPAR operand must match the value specified on the SYSTEM_IDENTIFIER statement that was added above.

13. Prepare to use the revised system configuration file:

    a. Rename the current SYSTEM CONFIG file to BACKUP CONFIG.

    b. Rename the TEMP CONFIG file to SYSTEM CONFIG.

    c. Make the PMAINT CF0 minidisk available to CP as file mode L:

    ```
    cpaccess pmaint cf0 l
    ```

Task 9: Deactivate the ISFC Links to the Decommissioned Member

Deactivate the ISFC links from the remaining members of the SSI cluster to the decommissioned member. Do this for each remaining member. You can issue all of the necessary commands from the current member.

1. For example, to deactivate the links from the current member (VMSYS02 in this example) to VMSYS03, issue a DEACTIVE ISLINK command for the devices:

   ```
   deactive islink rdev [rdev…]

   Link rdev deactivated.
   ```

2. To deactivate the links from a remote member (for example, VMSYS01) to VMSYS03, use the AT command to issue the DEACTIVE ISLINK command:

   ```
   at vmsys01 cmd deactive islink rdev [rdev…]

   Link rdev deactivated.
   ```

Task 10: Free the Decommissioned Member's DASD Volumes

Free the DASD volumes used by the decommissioned member.

1. Drain the VMSYS03 spool volumes. You need to issue the DRAIN command only once for each volume. For example, to drain volume M03S01:

   ```
   drain volid ms3s01 all
   ```

2. Detach the VMSYS03 spool volumes from the remaining members. Do this for each remaining member. You can issue all of the necessary commands from the current member.

   a. For example, to detach volume M03S01 from the current member (VMSYS02 in this example):

   ```
   detach volid ms3s01 from system
   ```

   b. To detach volume M03S01 from a remote member (for example, VMSYS01), use the AT command to issue the DETACH command:

   ```
   at vmsys01 cmd detach volid ms3s01 from system
   ```

3. Dynamically remove the VMSYS03 spool volumes from the CP-owned lists for the remaining members by defining the slots as reserved. Use the information you recorded in "Task 8: Update the System Configuration File" on page 824. For example, to remove volume M03S01, define slot 12 as reserved. Do this for each remaining member. You can issue all of the necessary commands from the current member.

   a. For example, to update the CP-owned list for the current member (VMSYS02 in this example):

   ```
   define cpowned slot 012 reserved
   ```

   b. To update the CP-owned list for a remote member (for example, VMSYS01), use the AT command to issue the DEFINE CPOWNED command:

   ```
   at vmsys01 cmd define cpowned slot 012 reserved
   ```

4. Delete all data from all of the VMSYS03 CP-owned and member-specific volumes.

   a. Attach the volumes:

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

   b. Format each volume with a new label:

   Note: Formatting the volume also removes the ownership information.

   ```
   cpfmtxa vdev volid

   CPFMTXA:
   FORMAT WILL ERASE CYLINDERS 00000-nnnnn ON DISK vdev
   DO YOU WANT TO CONTINUE? (YES | NO)

   yes

   HCPCCF6209I INVOKING ICKDSF.
   ⋮
   ```

Task 11: Update the User Directory

Update the common source directory to remove the decommissioned member from the multiconfiguration virtual machine definitions.

DirMaint Alternative

If DirMaint is installed and enabled in the SSI cluster, use DirMaint facilities to make the changes described in this task. Use the procedure for making multiple updates to a directory that is documented in *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*.

1. Make a backup copy and then edit the USER DIRECT file.
2. Update the DIRECTORY statement to remove the directory volume for VMSYS03 (M03RES).

   Before:

   ```
   DIRECTORY SSI 123 3390 M01RES M02RES M03RES
   ```

   After:

   ```
   DIRECTORY SSI 123 3390 M01RES M02RES
   ```

3. Update the multiconfiguration virtual machine definitions to remove the VMSYS03 configurations. In each definition, remove the BUILD statement for VMSYS03 from the identity entry and remove the associated subconfiguration entry. For example, to remove the configuration for TCPIP on VMSYS03:

   Before:

```
IDENTITY TCPIP password 128M 256M ABG
 INCLUDE TCPCMSU
 BUILD ON VMSYS01 USING SUBCONFIG TCPIP-1
 BUILD ON VMSYS02 USING SUBCONFIG TCPIP-2
 BUILD ON VMSYS03 USING SUBCONFIG TCPIP-3
 OPTION QUICKDSP SVMSTAT MAXCONN 1024 DIAG98 APPLMON
 SHARE RELATIVE 3000
 IUCV ALLOW
 IUCV ANY PRIORITY
 IUCV *CCS PRIORITY MSGLIMIT 255
 IUCV *VSWITCH MSGLIMIT 65535

SUBCONFIG TCPIP-1
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M01RES MR RTCPIP WTCPIP MTCPIP

SUBCONFIG TCPIP-2
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M02RES MR RTCPIP WTCPIP MTCPIP

SUBCONFIG TCPIP-3
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M03RES MR RTCPIP WTCPIP MTCPIP
```

After:

```
IDENTITY TCPIP password 128M 256M ABG
 INCLUDE TCPCMSU
 BUILD ON VMSYS01 USING SUBCONFIG TCPIP-1
 BUILD ON VMSYS02 USING SUBCONFIG TCPIP-2
 OPTION QUICKDSP SVMSTAT MAXCONN 1024 DIAG98 APPLMON
 SHARE RELATIVE 3000
 IUCV ALLOW
 IUCV ANY PRIORITY
 IUCV *CCS PRIORITY MSGLIMIT 255
 IUCV *VSWITCH MSGLIMIT 65535

SUBCONFIG TCPIP-1
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M01RES MR RTCPIP WTCPIP MTCPIP

SUBCONFIG TCPIP-2
 LINK TCPMAINT 491 491 RR
 LINK TCPMAINT 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 05179 005 M02RES MR RTCPIP WTCPIP MTCPIP
```

4. Remove the virtual machine definitions for single-configuration virtual machines that run only on VMSYS03 and will not be relocated to another member of the SSI cluster. If DirMaint is installed in the cluster, remove the virtual machine definitions for the DIRMSAT server (for example, DIRMSAT3) and DATAMOVE server (for example, DATAMOV3).
5. Create a new object directory for each of the remaining members of the cluster:

a. If you are using DirMaint, issue the following command (which will cause the directory to be created on each member):

```
dirmaint direct
```

b. If you are not using DirMaint:

   i) On the current member, issue the following command:

   ```
   directxa user direct fm
   ```

   ii) If there are other existing members:

      a) Log on to the MAINT user ID on that member.

      b) Access the PMAINT 2CC and 551 minidisks.

      c) Create the new object directory:

      ```
      directxa user direct fm
      ```

Task 12: Update the VMSES/E System-Level Product Inventory Table

Update the VMSES/E System-Level Product Inventory table (VM SYSPINV) to remove the decommissioned member from the product records.

1. Issue the following command:

```
vmfupdat syspinv remove system vmsys03
```

Task 13: Update the TCP/IP Configuration

Reconfigure the TCP/IP stack to remove the decommissioned member.

1. Link (multiple-write) and access the TCPMAINT 198 minidisk:

```
link to tcpmaint 198 as 198 m
access 198 fm
```

   a. Erase the TCPIP server configuration file for VMSYS03 (VMSYS03 TCPIP).

   b. Remove VMSYS03 from other multinode-capable files on this minidisk. For example:

      - Remove VMSYS03-specific logic in the global profile exit (TCPRUNXT EXEC).
      - Remove VMSYS03-specific configurations for other TCP/IP servers; for example, erase the VMSYS03 DTCPARMS file.

2. Link (multiple-write) and access the TCPMAINT 592 minidisk:

```
link to tcpmaint 592 as 592 m
access 592 fm
```

   a. Edit the TCPIP DATA file and add remove the HOSTNAME statement for VMSYS03.

Task 14: Update the Configuration Files for Other Service Virtual Machines

1. If needed, update the configuration files for other facilities, features, and products to remove configuration information for the decommissioned member. For example, if you are using the Performance Toolkit for centralized monitoring in your SSI cluster, see the preparation information in *z/VM: Performance Toolkit Guide*.
2. Decommissioning the SSI cluster member is now complete.

Chapter 38. Converting a CSE Complex to a z/VM SSI Cluster

z/VM 6.2 was the last release to support the cross system extensions (CSE) environment. If you want to upgrade the z/VM systems in a CSE complex, you must convert the CSE complex to a z/VM SSI cluster. A z/VM SSI cluster offers a more comprehensive clustering solution than a CSE complex and adds functionality not available in CSE, such as multisystem installation, single point of maintenance, autonomic minidisk cache management, and virtual server mobility.

Note: The cross-system link (XLINK) function that was included in CSE is still supported for non-SSI systems. See Appendix C, “Using Cross-System Link (XLINK),” on page 843.

The conversion of a CSE environment to an SSI environment is complex. Therefore this topic does not provide a step by step procedure, but offers an overview of a possible approach.

A Possible Conversion Approach

Use one of the SSI installation procedures documented in *z/VM: Installation Guide* to install a separate SSI cluster, either second level or in LPARs, using full-pack minidisks or dedicated DASD volumes. At some point the systems in the new SSI cluster will need to reside in real LPARs, so understanding how they will be defined is required regardless of whether you use the second level approach.

After the new SSI cluster is created, gradually and with caution move your workload to the SSI cluster. It is likely that you will want to copy your infrastructure virtual machines (such as servers and SVMs) before you copy your guest virtual machines and general users.

SYSAFFIN directory statements are not supported in an SSI-enabled directory. If you have virtual machines defined with SYSAFFIN statements to provide different logon configurations for the systems in the CSE complex, you need to convert them to multiconfiguration virtual machine definitions (IDENTITY definitions) to provide different logon configurations for the members of the SSI cluster.

The following example shows the definition of a virtual machine called WATCHER that has a different 191 disk for each CSE system.

```
USER WATCHER password 64M
   IPL CMS
   LINK MAINT 190 190
   LINK MAINT 19E 19E
   MDISK 192 3390 300 100 CSECM1 RR
   SYSAFFIN VMA
   MDISK 191 3390 100 100 VMAPV1 MR
   SYSAFFIN VMB
   MDISK 191 3390 100 100 VMBPV1 MR
   SYSAFFIN VMC
   MDISK 191 3390 100 100 VMCPV1 MR
   SYSAFFIN VMD
   MDISK 191 3390 100 100 VMDPV1 MR
```

The following example shows the converted definition for WATCHER. The following changes were made:

- Changed "USER" to "IDENTITY".
- Added a BUILD ON statement to the identity entry for each cluster member, specifying the ID of the SUBCONFIG statement that begins the subconfiguration entry for that member.
- Changed each SYSAFFIN statement to a SUBCONFIG statement with an ID that matches the corresponding BUILD statement. Note that in this example the volumes for the minidisks have also been changed.

```
IDENTITY WATCHER password 64M
   BUILD ON VMSYS01 USING SUBCONFIG WATCHR-1
   BUILD ON VMSYS02 USING SUBCONFIG WATCHR-2
   BUILD ON VMSYS03 USING SUBCONFIG WATCHR-3
   BUILD ON VMSYS04 USING SUBCONFIG WATCHR-4
   IPL CMS
   LINK MAINT 190 190
   LINK MAINT 19E 19E
   MDISK 192 3390 300 100 USRVL1 RR

SUBCONFIG WATCHR-1
   MDISK 191 3390 100 100 M01PV1 MR

SUBCONFIG WATCHR-2
   MDISK 191 3390 100 100 M02PV1 MR

SUBCONFIG WATCHR-3
   MDISK 191 3390 100 100 M03PV1 MR

SUBCONFIG WATCHR-4
   MDISK 191 3390 100 100 M04PV1 MR
```

It is also important to reflect any user modifications to the directory from your current CSE environment into the directory for the SSI cluster.

In addition, you need to address the handling of minidisks:

- Copy the minidisk to a new volume in the SSI cluster.
- Define the SSI minidisk on the same physical volume as the CSE minidisk. You need to ensure that the volume is defined as SHARED YES, so CP knows the device is shared outside the SSI cluster. However, there is no LINK protection, as CP cannot analyze link conflicts between an SSI cluster and a CSE complex. You should consider deleting or commenting-out the MDISK definition on the CSE system, if appropriate.

Move over to the SSI cluster any spool files needed by the workload, such as NSSs and DCSSs.

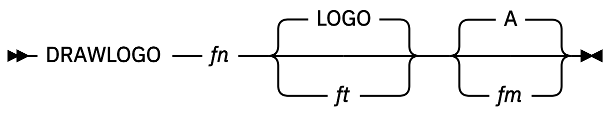
Prior to allowing the virtual machines to be used in the new SSI environment, it would be wise to ensure they can no longer run on the CSE systems (for example, by making them NOLOG), to avoid the virtual machines running in both environments at the same time.

Appendix A. Sample Utility Programs

This appendix describes a sample (unsupported) utility program, DRAWLOGO, which is supplied with z/VM. A sample XEDIT macro, X$DRWL$X, which is used by DRAWLOGO, is also supplied. The DRAWLOGO sample program is shipped with a file type of SAMPEXEC. The X$DRWL$X sample XEDIT macro has a file type of SAMPXEDI. By default these files are loaded onto the CP samples disk, 2C2, owned by user MAINT*vrm* (where *vrm* is the z/VM version, release, and modification level) if z/VM was loaded to minidisk. It is on `VMPSFS:MAINTvrm.CPDV.SAMPLE`, if z/VM was loaded to filepool.

To use the DRAWLOGO utility, you must change its file type from SAMPEXEC to EXEC and change the file type of X$DRWL$X from SAMPXEDI to XEDIT.

DRAWLOGO



Purpose

Use DRAWLOGO to create logo screens for your z/VM system. DRAWLOGO lets you edit the text of a logo file and modify the 3270 screen attributes in a logo file.

For information about how to use this utility, see "Creating Logo Screens" on page 340.

Operands

fn
is the file name of the logo file.

ft
is the file type of the logo file. The default file type is LOGO.

fm
is the file mode of the logo file. The default file mode is A.

Usage Notes

1. When you enter DRAWLOGO, you are placed in an XEDIT session. If the DRAWLOGO utility does not recognize the type of logo (online message, body of logo, input area, or printer separator page), DRAWLOGO prompts you for the type of logo.
2. DRAWLOGO invokes the X$DRWL$X XEDIT macro, which must be available.
3. To make your new logo screen active, target the LOGO CONFIG file after updating it to include your new logo screen.

Return Codes

In addition to the return code listed, DRAWLOGO might return a return code from the CMS LISTFILE and CMS XEDIT commands. For a list of return codes that the CMS LISTFILE and CMS XEDIT commands generate, see *z/VM: CMS Commands and Utilities Reference*.

Code	Meaning
4	No minidisk accessed in R/W mode

Appendix B. Device Class and Type Codes

This appendix is the HCPDVTYP copy file and can be found in macro library HCPOM1.

Device Class Definitions

```
DEVCLAS BIT DEFINITIONS     DEVICE CLASSES

CLASAIF  EQU   X'FF'     Adapter-Interrupt-Facility (AIF) Class
CLASTERM EQU   X'80'     TERMINAL DEVICE CLASS
CLASGRAF EQU   X'40'     GRAPHIC DISPLAY DEVICE CLASS
CLASGRFR EQU   X'41'     GRAPHIC DISPLAY DEVICE CLASS (REMOTE)
CLASPOOL EQU   X'20'     UNIT RECORD SPOOLING DEVICE CLASS
CLASTAPE EQU   X'08'     MAGNETIC TAPE DEVICE CLASS
CLASDASD EQU   X'04'     DIRECT ACCESS STORAGE DEVICE CLASS
CLASSPEC EQU   X'02'     SPECIAL DEVICE CLASS
CLASSVCM EQU   X'10'     SIMULATED DEVICE CLASS
CLASSWCH EQU   X'01'     SWITCH DEVICE CLASS
```

Device Type Definitions within Device Classes

```
DEVTYPE BIT DEFINITIONS     DEVICE TYPES BY DEVICE CLASSES

TYP2700  EQU   X'80'     TERM - 2700 BISYNC LINE
TYPBSC   EQU   X'88'     TERM - BISYNC LINE FOR 3270
                                REMOTE STATION
TYPCONS  EQU   X'40'     TERM - CONSOLE DEVICE
TYP3215  EQU   X'40'     TERM - 3215 CONSOLE
TYP1052  EQU   X'40'     TERM - 1052 CONSOLE
TYPTTY   EQU   X'20'     TERM - USASCII-8 TELEGRAPH TERMINAL
TYPIBM1  EQU   X'10'     TERM - IBM TERMINAL CONTROL TYPE 1
TYPUNDEF EQU   X'1C'     TERM - TERMINAL TYPE UNDEFINED
TYP2741  EQU   X'18'     TERM - 2741 COMMUNICATIONS TERMINAL
TYP3767  EQU   X'18'     TERM - 3767 IN 2741 COMPATIBILITY MODE
TYP1050  EQU   X'14'     TERM - 1050 COMMUNICATIONS TERMINAL
TYPSDLC  EQU   X'08'     TERM - SDLC INTEGRATED ADAPTER
TYPTELE2 EQU   X'20'     TERM - TELE2 INTEGRATED ADAPTER
TYPHDLC  EQU   X'24'     TERM - HDLC INTEGRATED ADAPTER
TYPILAN  EQU   X'28'     TERM - ILAN INTEGRATED ADAPTER
TYPELAN  EQU   X'28'     TERM - ELAN INTEGRATED ADAPTER
TYPIC    EQU   X'04'     TERM - Integrated console
```

```
TYP3270  EQU   X'40'     GRAF - 3270 GENERIC DISPLAY STATION
TYP3277  EQU   X'80'     GRAF - 3277 DISPLAY STATION
TYP3278  EQU   X'40'     GRAF - 3278 DISPLAY STATION
TYP3178  EQU   X'40'     GRAF - 3178 DISPLAY STATION
TYP3279  EQU   X'40'     GRAF - 3279 DISPLAY STATION
TYP3179  EQU   X'40'     GRAF - 3179 DISPLAY STATION
TYP3180  EQU   X'40'     GRAF - 3180 DISPLAY STATION
TYP3290  EQU   X'40'     GRAF - 3290 DISPLAY STATION
TYP3190  EQU   X'40'     GRAF - 3190 DISPLAY STATION
TYP3271  EQU   X'20'     GRAF - 3271 CONTROLLER (REMOTE)
*YP3274  EQU   X'21'     GRAF - 3274 CONTROLLER (REMOTE)
TYP3275  EQU   X'10'     GRAF - 3275 DISPLAY STATION
*YP3276  EQU   X'11'     GRAF - 3276 DISPLAY STATION
TYP3284  EQU   X'08'     GRAF - 3284 PRINTER
TYP3286  EQU   X'08'     GRAF - 3286 PRINTER
TYP3287  EQU   X'09'     GRAF - 3287 PRINTER
TYP3288  EQU   X'08'     GRAF - 3288 PRINTER
TYP3289  EQU   X'09'     GRAF - 3289 PRINTER
TYP3287L EQU   X'08'     GRAF - 3287 LOGICAL PRINTER
TYP3289L EQU   X'08'     GRAF - 3289 LOGICAL PRINTER
TYP2250  EQU   X'04'     GRAF - 2250 DISPLAY UNIT
TYP3250  EQU   X'04'     GRAF - 3250 DISPLAY UNIT
TYP5080  EQU   X'06'     GRAF - 5080 DISPLAY UNIT
TYPCLUST EQU   X'30'     GRAF - CLUSTER CTLR (3271 OR 3275)
```

```
TYPRDR   EQU   X'80'     SPOL - CARD READER DEVICE
TYP2501  EQU   X'81'     SPOL - 2501 CARD READER
```

```
TYP2540R EQU   X'82'     SPOL - 2540 CARD READER
TYP3505  EQU   X'84'     SPOL - 3505 CARD READER
TYPPUN   EQU   X'40'     SPOL - CARD PUNCH DEVICE
TYP2540P EQU   X'42'     SPOL - 2540 CARD PUNCH
TYP3525  EQU   X'44'     SPOL - 3525 CARD PUNCH
```

```
TYPPRT   EQU   X'20'     SPOL - PRINTER TYPE DEVICE
TYP1403  EQU   X'21'     SPOL - 1403 PRINTER
TYP32XX  EQU   X'22'     SPOL - 3203 OR 3211 PRINTER
TYP3203  EQU   X'26'     SPOL - 3203 PRINTER
TYP3211  EQU   X'22'     SPOL - 3211 PRINTER
TYP3800  EQU   X'28'     SPOL - 3800 PRINTER
TYP3262  EQU   X'23'     SPOL - 3262 PRINTER
TYP4245  EQU   X'24'     SPOL - 4245 PRINTER
TYP4248  EQU   X'29'     SPOL - 4248 PRINTER
TYP3820  EQU   X'25'     SPOL - 3820 PRINTER - DEDICATED ONLY
TYPAFP1  EQU   X'27'     SPOL - AFP1 PRINTER - DEDICATED ONLY
TYPVAFP  EQU   X'2A'     SPOL - VAFP PRINTER - SIMULATED ONLY
TYPSYS   EQU   X'10'     SPOL - SYSTEM VIRT DEVICE FOR DUMPS
```

```
TYP3420  EQU   X'10'     TAPE - 3420 TAPE DRIVE
TYP3422  EQU   X'80'     TAPE - 3422 TAPE DRIVE
TYP3424  EQU   X'42'     TAPE - 3424 TAPE DRIVE
TYP3430  EQU   X'20'     TAPE - 3430 TAPE DRIVE
TYP3480  EQU   X'40'     TAPE - 3480 TAPE DRIVE
TYP3490  EQU   X'81'     TAPE - 3490 TAPE DRIVE
TYP3590  EQU   X'83'     TAPE - 3590 TAPE DRIVE
TYP9348  EQU   X'44'     TAPE - 9348 TAPE DRIVE
```

```
TYP3330  EQU   X'40'     DASD - 3330 DISK STORAGE FACILITY
TYP3340  EQU   X'20'     DASD - 3340 DISK STORAGE FACILITY
TYP3350  EQU   X'10'     DASD - 3350 DISK STORAGE FACILITY
TYP3350C EQU   X'11'     DASD - 3350 4 X 8 PAGING STORAGE
TYP3350D EQU   X'12'     DASD - 3350 4 X 4 PAGING STORAGE
TYP2305  EQU   X'08'     DASD - 2305 FIXED HEAD STORAGE FACILITY
TYP3380  EQU   X'04'     DASD - 3380 DISK STORAGE FACILITY
TYP3390  EQU   X'82'     DASD - 3390 DISK STORAGE FACILITY
TYP3375  EQU   X'80'     DASD - 3375 DISK STORAGE FACILITY
TYP3370  EQU   X'02'     DASD - 3370 DISK STORAGE FACILITY
TYP9345  EQU   X'81'     DASD - 9345 DISK STORAGE FACILITY
TYP9332  EQU   X'21'     DASD - 9332 FBA DASD
TYP9335  EQU   X'22'     DASD - 9335 FBA DASD
TYP9336  EQU   X'24'     DASD - 9336 FBA DASD
TYPFBA   EQU   X'28'     DASD - GENERIC FBA DASD
```

```
TYP9033  EQU   X'40'     SWCH - 9033 SWITCH DEVICE
TYP9032  EQU   X'80'     SWCH - 9032 SWITCH DEVICE
TYP2032  EQU   X'20'     SWCH - 2032 FICON Switch device
```

```
TYPCTCA  EQU   X'80'     SPEC - CHANNEL TO CHANNEL ADAPTER
TYP3088  EQU   X'80'     SPEC - 3088 MULTISYSTEM CHANNEL
                                COMMUNICATION UNIT
TYP3704  EQU   X'40'     SPEC - 3704 PROGRAMMABLE
                                COMMUNICATION CONTROL UNIT
TYP3705  EQU   X'40'     SPEC - 3705 PROGRAMMABLE
                                COMMUNICATION CONTROL UNIT
TYPOSA   EQU   X'20'     SPEC - OPEN SYSTEMS ADAPTER DEVICE
TYP3851  EQU   X'02'     SPEC - MSS MASS STORAGE COMMUNICATOR
TYP3890  EQU   X'08'     SPEC - 3890 DOCUMENT PROCESSOR
TYPUNSUP EQU   X'01'     SPEC - DEVICE UNSUPPORTED BY
                                THE VM/370 MIGRATION AID
TYPFCP   EQU   X'10'     SPEC - Open FCP Adapter (FCP)
```

```
TYPDYNIO EQU   X'02'     VCM  - Dynamic I/O
TYPMSGF  EQU   X'01'     VCM  - Message Facility
```

```
TYPAIF0  EQU   X'00'     AIF  - AIF Adapter Type 0
```

Device Features by Class and Type

```
DEVFEAT BIT DEFINITIONS   FEATURES OF DEVICES BY CLASS/TYPE

FTROPRDR EQU   X'80'     GRAF - OPERATOR ID CARD READER
FTRDIAL  EQU   X'01'     GRAF - 3275 WITH SWITCHED LINE SUPPORT
```

```
FTRUCS   EQU   X'01'     SPOL - UCS FEATURE
FTR4WCGM EQU   X'80'     SPOL - 3800 WITH FOUR WRITEABLE
                                CHARACTER GENERATION MODULES
```

```
FTR7TRK  EQU   X'80'     TAPE - 7-TRACK FEATURE
FTRDUAL  EQU   X'40'     TAPE - DUAL DENSITY FEATURE
FTRTRAN  EQU   X'20'     TAPE - TRANSLATE FEATURE
FTRCONV  EQU   X'10'     TAPE - DATA CONVERSION FEATURE
FTRCMPCT EQU   X'08'     TAPE - DATA COMPACTION FEATURE
```

```
FTRRPS   EQU   X'80'     DASD - ROTATIONAL POSITIONAL SENSING
FTRDYNP  EQU   X'40'     DASD/TAPE - DYNAMIC PATHING FEATURE
FTRVUA   EQU   X'20'     DASD - 3330V THAT MAY BE DEDICATED
                                TO A VIRTUAL MACHINE
*TRSYSV  EQU   X'10'     DASD - 3330V THAT MAY BE USED BY
                                VM FOR MOUNTING MSS SYSTEM
                                VOLUMES
FTR35MB  EQU   X'08'     DASD - 35 MB DATA MODULE (3340)
FTR70MB  EQU   X'04'     DASD - 70 MB DATA MODULE (3340)
FTRRSRL  EQU   X'02'     DASD/TAPE/SPEC - RESERVE/RELEASE
                                CCW FEATURE
FTRCOMP  EQU   X'01'     DASD - 3350 IN 3330 COMPAT. MODE
```

```
FTRTERM  EQU   X'80'     SPEC - UNSUPPORTED TERMINAL DEVICE
FTRGRAF  EQU   X'40'     SPEC - UNSUPPORTED GRAPHIC DISPLAY
                                DEVICE
FTRSPOOL EQU   X'20'     SPEC - UNSUPPORTED UNIT RECORD
                                SPOOLING DEVICE
FTRTAPE  EQU   X'08'     SPEC - UNSUPPORTED MAGNETIC TAPE
                                DEVICE
FTRDASD  EQU   X'04'     SPEC - UNSUPPORTED DIRECT ACCESS
                                DEVICE
FTRSWCH  EQU   X'01'     SPEC - UNSUPPORTED DYNAMIC SWITCH
```

```
FTRTYP1  EQU   X'10'     SPEC - TYPE ONE CHANNEL ADAPTER
FTRTYP4  EQU   X'40'     SPEC - TYPE FOUR CHANNEL ADAPTER
```

Device Model Definitions

Note: This list contains only commonly compared model numbers. It is NOT a complete list of supported devices or models.

```
DEVMODEL CODE DEFINITIONS    RDEVDVMN DEVICE MODEL NUMBERS

M230502  EQU   X'02'     DASD - 2305 MODEL 2
M333001  EQU   X'01'     DASD - 3330 MODEL 1 (404 CYLINDERS)
M333002  EQU   X'01'     DASD - 3330 MODEL 2 (404 CYLINDERS)
M333011  EQU   X'11'     DASD - 3330 MODEL 11(808 CYLINDERS)
M3380E   EQU   X'0A'     DASD - 3380 MODEL E
M3380K   EQU   X'1E'     DASD - 3380 MODEL K
```

```
M3390N   EQU   X'02'     DASD - 3390 MODEL 1 NATIVE MODE
M3390E   EQU   X'96'     DASD - 3390 MODEL 1 EMULATION MODE
M3391N   EQU   X'06'     DASD - 3390 MODEL 2 NATIVE MODE
M3391E   EQU   X'8A'     DASD - 3390 MODEL 2 EMULATION MODE
M3393N   EQU   X'0A'     DASD - 3390 MODEL 3 NATIVE MODE
M3393E   EQU   X'9E'     DASD - 3390 MODEL 3 EMULATION MODE
M3399N   EQU   X'0C'     DASD - 3390 MODEL 9 NATIVE MODE
M934500  EQU   X'00'     DASD - 9345 MODEL 1
M934504  EQU   X'04'     DASD - 9345 MODEL 2
M380001  EQU   X'01'     SPOL - 3800 PRINTER MODEL 1
M380003  EQU   X'03'     SPOL - 3800 PRINTER MODEL 3
```

```
M380008  EQU   X'08'     SPOL - 3800 PRINTER MODEL 8

M327702  EQU   X'02'     GRAF - 3277 DISPLAY STATION MODEL 2
```

```
M308861  EQU   X'61'     SPEC - 3088 CTCA PCA adaptor card
                                with CLAW support.
M308862  EQU   X'62'     SPEC - 3088 OSA card
CUT1731  EQU   X'1731'   SPEC - OSA-Express Control Unit Type
DVT1731  EQU   X'1731'   SPEC - OSA Direct Express Agent DEV
DVT1732  EQU   X'1732'   SPEC - OSA Direct Express Device Type
```

Appendix C. Using Cross-System Link (XLINK)

Cross-system link (XLINK) extends CP link protocols to control normal, stable, and exclusive read or read/write access across multiple VM systems for minidisks on shared DASD.

Note: Support for the cross system extensions (CSE) environment has been removed. z/VM single system image (SSI) clusters provide the new technology for clustering z/VM systems. The XLINK function that was included in CSE is still supported for non-SSI systems, and "CSE" is still used in some function and object names, command responses, and messages related to XLINK.

Planning for Cross-System Link

Before using the cross-system link function, make sure that you understand the capabilities, requirements, and restrictions.

Shared DASD

Only count-key-data (CKD) and extended-count-key-data (ECKD) DASD are supported for cross-system link. Cross-system link does not work with fixed-block architecture (FBA) DASD. XLINK is limited to mapping real cylinders up to and including cylinder 65520 on an Extended Address Volume (EAV).

For cross-system link to control links between multiple VM systems, the real volumes on which the minidisks reside must be physically and logically shared by all systems. Physical sharing is provided by having the disk drives accessible to all systems. Logical sharing is assured by corresponding I/O device specifications (in the system configuration files and IOCPs if needed) and identical volume lists (in XLINK_VOLUME_INCLUDE and XLINK_VOLUME_EXCLUDE configuration statements) on all of the systems. Cross-system link does not use real reserve/release.

MDISK Statements in the CP User Directory

The CP user directory defines the location and extent of all minidisks. For two or more systems to link to the same minidisks, the MDISK directory entries for all shared volumes must be identical on every system participating in the XLINK cross-system link function. CP and the directory program do not prevent you from defining minidisks that overlap. A CP system using the XLINK cross-system link function is completely unable to recognize overlapping minidisks on other systems because the minidisks are known solely by their starting cylinders. If you define such overlap, you assume responsibility for data integrity.

Maximum Number of Systems

Only VM systems that are not members of an SSI cluster can participate in cross-system link. The list of participating systems is defined by XLINK_SYSTEM_INCLUDE and XLINK_SYSTEM_EXCLUDE configuration statements on all of the systems.

The maximum number of systems supported for cross-system link is 56 if only ECKD devices are used. If any shared CKD devices are used, then the maximum number of systems supported depends on the device type. See Table 54 on page 852 for the maximum number of systems (XLINK map records) supported for various DASD types.

Performance Considerations

When DASD are shared, the chance of having high contention volumes is increased because potentially more users have access to one or more minidisks on the same volume. To achieve better I/O performance:

- Use 4 KB block formatting on all CMS minidisks.
- Ensure that DASD cache is enabled for a volume that contains heavily used *read-only* minidisks. This can dramatically reduce the volume's busy time.

- If possible, put heavily used minidisks on separate volumes and assign users to these minidisks so that the load is primarily from a single system.

Minidisk Cache (MDC) Feature

Because the VM minidisk cache (MDC) feature does not address cross-cache invalidation, the scope of MDC is limited to one system only.

Attention: Using MDC with cross-system link might cause data integrity problems.

Because CACHING=ON is the default for the real device, you can identify a device on which shared minidisks reside by specifying SHARED YES on the RDEVICE statement in the system configuration file. This specification prevents MDC for the device. Alternatively, you can suppress MDC at a more granular level by specifying NOMDC on the MINIOPT directory statement for shared minidisks, or by using the SET MDCACHE command.

Placing of the CSE Area

CP keeps control information about the state of cross-system links to minidisks on a special area (called the *CSE area*) of the real volume where the minidisks reside. The CSE area must be initialized with the XLINK FORMAT utility. CP updates the information in the CSE area only when the first link to that minidisk occurs, when a link higher than any other links to that minidisk occurs, or when the last link of that type disappears. It is important that this area not be inadvertently updated by any program other than CP.

If you do not specify a particular location for the CSE area, CP initializes the CSE area starting on cylinder 0, track 1, when you issue the XLINK FORMAT command. Because CP keeps the volume label and allocation maps on cylinder 0, track 0, the default location for the CSE area should be available. Allocate the location you choose for the CSE area as PERM space with the CPFMTXA utility.

Notes:

1. For some device types, the CSE area resides on multiple cylinders. See Table 54 on page 852. When reserving space for the CSE area, these cylinders must also be considered.
2. Do not define user minidisks in the space reserved for the CSE area, except for the purpose described under “Protecting the CSE Area” on page 848.

Synchronization of Directories

To protect your system's minidisks when they are shared with other systems, the directory statements for shared DASD volumes must be kept exactly synchronized.

To enable multiple systems to link to the same minidisks, the MDISK directory statements for all shared volumes must be identical on all of the systems. Directory entries for user IDs that exist on multiple systems (and therefore might be logged on simultaneously) should not contain MDISK or LINK statements with an access mode of multiwrite (M/W) if these minidisks are used by CMS. When a hardware path from each system to the proper device is defined, and identical minidisk extents in the directory are also defined, minidisks become available to every system.

You can use the OPTION, MDISK, or LINK directory statements to give a specified user sole write access to a minidisk for a limited period of time, thus offering increased data integrity in cases where the stability of a given minidisk is required. The LNKSTABL and LNKEXCLU operands of the OPTION directory statement authorize a user to use the stable and exclusive link modes of the LINK command and DIAGNOSE code X'E4'. This global authority allows a virtual machine to perform a stable link to any minidisk to which it has password-level authorization. You can also specify stable and exclusive authority to a specific minidisk using the mode suffix letter (S or E) on the MDISK or LINK directory statements. While stable or exclusive links to a minidisk are in effect, access to that minidisk by any other user will be restricted or denied. For more information about the LNKSTABL and LNKEXCLU options, see the “OPTION Directory Statement” on page 572. For more information about link accesses, see “LINK Directory Statement” on page 533 or “MDISK Directory Statement” on page 550.

Setting Up Cross-System Link

To set up and enable the cross-system link function, you need to do the following on each system:

1. Define the systems and volumes on XLINK statements in the system configuration file (see "Defining the Systems and Volumes for Cross-System Link" on page 845).
2. Initialize the volumes for sharing (see "Initializing the Volumes for Cross-System Link" on page 845).
3. IPL the systems to establish the cross-system link function.
4. Verify the cross-system links (see "Verifying Cross-System Link" on page 846).

Defining the Systems and Volumes for Cross-System Link

Define the participating systems and shared DASD volumes with XLINK statements in the system configuration file. The lists of systems and volumes must be identical on each participating system.

XLINK_SYSTEM_INCLUDE
This statement defines a system to be included in the cross-system link function. The statement includes a SLOT *n* operand that defines the position of the system in the list of participating systems. Create a statement for each system. For more information, see "XLINK_SYSTEM_INCLUDE Statement" on page 329.

XLINK_SYSTEM_EXCLUDE
This statement defines a system to be excluded from the cross-system link function. For more information, see "XLINK_SYSTEM_EXCLUDE Statement" on page 328.

XLINK_VOLUME_INCLUDE
This statement defines a DASD volume to be included in the cross-system link function. Create a statement for each shared volume. For more information, see "XLINK_VOLUME_INCLUDE Statement" on page 333.

XLINK_VOLUME_EXCLUDE
This statement defines a DASD volume to be excluded from the cross-system link function. For more information, see "XLINK_VOLUME_EXCLUDE Statement" on page 331.

XLINK_DEVICE_DEFAULTS
This statement can be used to change the defaults for the CSE area location and format for specific DASD types. For more information, see "XLINK_DEVICE_DEFAULTS Statement" on page 324.

Note: Changing the defaults is not recommended. This statement allows you to provide link support for new devices with new geometries, should that become necessary in the future.

Initializing the Volumes for Cross-System Link

The CSE area on each DASD volume contains link information about the minidisks on that volume. To prepare the volume for sharing, you need to format the CSE area with the XLINK FORMAT utility.

Generating the XLINK Utility Program

The XLINK utility program (XLINK MODULE) contains the XLINK FORMAT and XLINK DISPLAY utilities. The DASD parameters used by these XLINK utilities must be the same as those used by CP. Therefore, if you changed any of the CSE area defaults on XLINK_DEVICE_DEFAULTS or XLINK_VOLUME_INCLUDE statements, you need to generate an updated version of the XLINK utility program before you use XLINK FORMAT to initialize the CSE area. Use the VMFBLD EXEC to generate the XLINK MODULE on each system:

```
VMFBLD PPF ppfname compname HCPXLOAD XLINK (ALL
```

where:

ppfname
is the name of the product parameter file

compnmame
is the name of the component

For information about the VMFBLD EXEC, see *z/VM: VMSES/E Introduction and Reference*.

Initializing the CSE Area on the Shared Minidisk Volumes

Use the XLINK FORMAT utility to format the CSE area on each DASD volume that cross-system link will control. You must format the CSE areas before the cross-system link function can be activated.

Note: If you changed the defaults for the CSE area, you must IPL the system prior to running XLINK FORMAT.

To initialize the CSE area on a shared minidisk volume:

1. Connect the volume to be formatted to the virtual machine by one of the following methods:
   a. Use the LINK command, in R/W mode, to link to a minidisk that defines a portion of the volume from cylinder 0 to where the CSE area is assigned. The default assignment for the CSE area is on cylinder 0. This method is preferred because it does not expose any data beyond the CSE area.

      Note: The CSE area on some devices will reside on multiple cylinders. If you are using one of those devices, your minidisk must include all of those cylinders. See Table 54 on page 852 for devices that use multiple cylinders for the CSE area.
   b. Use the LINK command, in R/W mode, to link to a minidisk that defines the entire real volume (a full-pack minidisk).
   c. Use the ATTACH command to attach the real volume to the virtual machine.

   For information about the LINK and ATTACH commands, see *z/VM: CP Commands and Utilities Reference*.
2. Enter an XLINK FORMAT command that specifies the virtual address and the real volume identification of the volume to be formatted:

   ```
   XLINK FORMAT vaddr volid
   ```

   The volume label on the disk must match the volume name given on the command. This command is run under CMS. Therefore, it does not require that you have any special CP privilege class. However, the command does require write access to the specified device.

You should run the XLINK FORMAT utility before you load the new CP module. However, if you wish to initialize the CSE area to something other than the defaults, you must IPL the system prior to running XLINK FORMAT.

For more information about the XLINK FORMAT utility, see *z/VM: CP Commands and Utilities Reference*.

Verifying Cross-System Link

You can use the XLINK CHECK command to determine whether one or more volumes are under cross-system link control:

```
XLINK CHECK volid1...volidn
```

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

Controlling Cross-System Link

The lists of systems participating in cross-system link and the shared DASD volumes under cross-system link protection are controlled by the XLINK statements in the system configuration file (see "Defining the Systems and Volumes for Cross-System Link" on page 845). The lists of systems and volumes must be identical on all of the participating systems. CP cannot coordinate these definitions across the systems. Therefore, if you modify or add a statement on one system, you need to make the same change on the other systems.

The XLINK RESET command is provided to recover links and prevent lockouts if one of the systems participating in cross-system link fails.

Attention: Incorrect use of this command might cause a loss of minidisk data. For more information, see *z/VM: CP Commands and Utilities Reference*.

CSE Area

Information about the state of current cross-system links is kept on the CSE area on each DASD volume being shared. To see these link indicators, use the XLINK DISPLAY utility. For information about the XLINK DISPLAY response, see *z/VM: CP Commands and Utilities Reference*.

The CKD cross-system link support uses a single track on each real DASD volume to hold information about the status of links to each of the minidisks contained on that volume. This information is contained in map records. There is one map record for each system in the complex and one byte in each map record for each cylinder on the volume. Because CKD minidisks always begin on a cylinder boundary, a particular byte in a particular map record can represent the status of a particular system's links to a particular minidisk on that real DASD volume. A single channel program is used to interrogate and update the map records for each link mode. See Figure 51 on page 848 for an example of the format of a CKD-formatted CSE area, and see Figure 52 on page 849 for an example of map records for a CKD-formatted CSE area.

The ECKD cross-system link support uses multiple tracks on each real DASD volume to hold information about the status of links to each of the minidisks contained on that volume. This information is contained in link records. There is one link record for each system in the complex on each track and one half byte in each link record for each cylinder on the volume. Each link record is 256 bytes long and contains link information for 512 cylinders. Multiple cylinders will be used if the number of tracks needed for the CSE area is greater than the number of tracks per cylinder. A series of channel programs are used to interrogate and update the link flag. See Figure 53 on page 850 for an example of an ECKD-formatted CSE area, and Figure 54 on page 851 for an example of link records for an ECKD-formatted CSE area.

Link states are signified by the following flags:

NOLINK
No links exist for the minidisk.

READ
Read links exist for the minidisk, but write links do not.

WRITE
At least one write link, and possibly one or more read links, exist for the minidisk.

STABLE READ
At least one stable read link, and possibly one or more read links exist for the minidisk. No write links will be granted while the STABLE READ flag exists for the minidisk.

STABLE WRITE
One stable write link, and possibly one or more read links exist for the minidisk. No write links will be granted while the STABLE WRITE flag exists for the minidisk.

EXCLUSIVE READ
One exclusive read link exists for the minidisk. No links will be granted while the EXCLUSIVE READ flag exists for the minidisk.

EXCLUSIVE WRITE
One exclusive write link exists for the minidisk. No links will be granted while the EXCLUSIVE WRITE flag exists for the minidisk.

For cross-system link to operate, the CSE area must be formatted properly. You can use cross-system link only with volumes that are currently attached to the system by CP and that have been formatted by the XLINK FORMAT utility. Cross-system link cannot support DASD that have not been formatted by this utility or that are attached to virtual machines.

The XLINK FORMAT utility writes the map records and other control information on the CSE area of each volume. The cylinder containing the CSE area must have been allocated as PERM space. Make sure that no user minidisks are allocated on that cylinder. To prevent inadvertent changes, no program other than CP should have write access to the CSE area of any volume attached to the system.

Because the CSE area for each shared volume is on that volume, any damage to the volume results in a loss of access to that volume only.

Location of the CSE Area

The CSE area may be placed at any unused location on a volume. If no other area was specified, CP uses the default location of cylinder 0, track 1. CP keeps the volume label and allocation maps on cylinder 0, track 0. Therefore, the remaining tracks on cylinder 0 should be available. Regardless of the location of the CSE area, it must be allocated as PERM space. For some device types, the CSE area resides on multiple cylinders. Table 54 on page 852 indicates the number of cylinders used for each device.

CP needs to update the link information only when the first link to that minidisk occurs, when a link higher than any other links to that minidisk occurs, or the last link of that type is detached. Thus, the first link to a minidisk causes the disk arm to move to the CSE area, but subsequent links might not.

To reduce the need for, and frequency of, I/O activity to CSE areas, the primary read-only (RR) link mode should be used in LINK commands and directory statements whenever possible. Because RR links are always granted when any nonexclusive link exists on a given system, CSE is not concerned with the information on the CSE area.

If the disk arm does move, it moves to the CSE area, which defaults to track 1 of cylinder 0. If a minidisk is frequently linked (more than once a second) in a mode other than RR, or if the type of access changes frequently from R/O to R/W, it might be useful to move that minidisk as close to the CSE area as possible. This avoids long seeks on every LINK command. Alternatively, the CSE area could be moved to a different position on the volume, closer to the high-usage minidisk.

Protecting the CSE Area

To protect the CSE areas from inadvertent assignment as a user's minidisk space, it might be helpful to assign a NOLOG user ID to the cylinder on which the area resides. Access to the cylinder containing the CSE area must be strictly controlled.

The Format of the CSE Area on a CKD Volume

Figure 51 on page 848 illustrates the general format of a CKD-formatted CSE area.

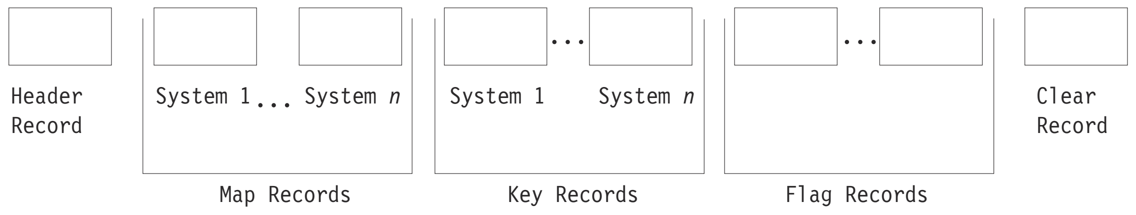


Figure 51. The Format of a CKD-Formatted CSE Area

A CKD-formatted CSE area contains the following records:

1 Header Record
Contains 240 bytes of general cross-system link information.

***n* Map Records**
One record per system, each containing link information for a cylinder on the volume. The length of the data area is greater than or equal to the number of cylinders on the volume.

***n* Key Records**
One record per system, each containing 1 byte of irrelevant information. The key area is used as a comparison argument.

7 Flag Records
Each record contains a byte representing NOLINK, READ, WRITE, STABLE READ, STABLE WRITE, EXCLUSIVE READ, or EXCLUSIVE WRITE link flags.

1 Clear Record
Contains NOLINK link flags in all bytes. The length is the same as a map record.

n
is the number of map records on this track. The number of sharing systems cannot exceed this value. The recommended value for *n* is equal to the maximum track capacity. If it is necessary to change this value, use the MAP_RECORDS operand on the XLINK_DEVICE_DEFAULTS statement (see "XLINK_DEVICE_DEFAULTS Statement" on page 324) or the XLINK_VOLUME_INCLUDE statement (see "XLINK_VOLUME_INCLUDE Statement" on page 333).

The number of map records used should equal the number of systems that are participating in cross-system link.

The map records show the state of the existing links on a CKD DASD for the systems participating in cross-system link. Figure 52 on page 849 is an example.

```
                  Byte      0                  |1                  |2
                  Position  0 1 2 3 4 5 6 7 8 9|0 1 2 3 4 5 6 7 8 9|0
                            | | | | | | | | | | | | | | | | | | | | |

Map record 1, for system1:  1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 ...

Map record 2, for system2:  1 1 1 1 1 1 5 1 1 1 1 1 1 1 1 3 1 1 1 1 1 ...

Map record 3, for system3:  0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 ...

Map record 4, for system4:  4 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 6 1 ...
```

Figure 52. Sample Map Records for Four Systems in a CKD-Formatted CSE Area

The following values are used in the map records to indicate the link state flags:

Value	Link State Flag
1	NOLINK
0	READ
2	WRITE
3	STABLE READ
4	STABLE WRITE
5	EXCLUSIVE READ
6	EXCLUSIVE WRITE

The map records shown in Figure 52 on page 849 indicate the following links:

- System1 has write links to a minidisk beginning at cylinder 3 and stable read links to a minidisk beginning at cylinder 15.
- System2 has an exclusive read link to a minidisk beginning at cylinder 6 and stable read links to a minidisk beginning at cylinder 15.
- System3 has read links to minidisks beginning at cylinders 0, 3, and 11.
- System4 has a stable write link to a minidisk beginning at cylinder 0, write links to minidisks beginning at cylinder 11, and an exclusive write link to a minidisk beginning at cylinder 19.

Format of the CSE Area on an ECKD Volume

Figure 53 on page 850 illustrates the general format of an ECKD-formatted CSE area.

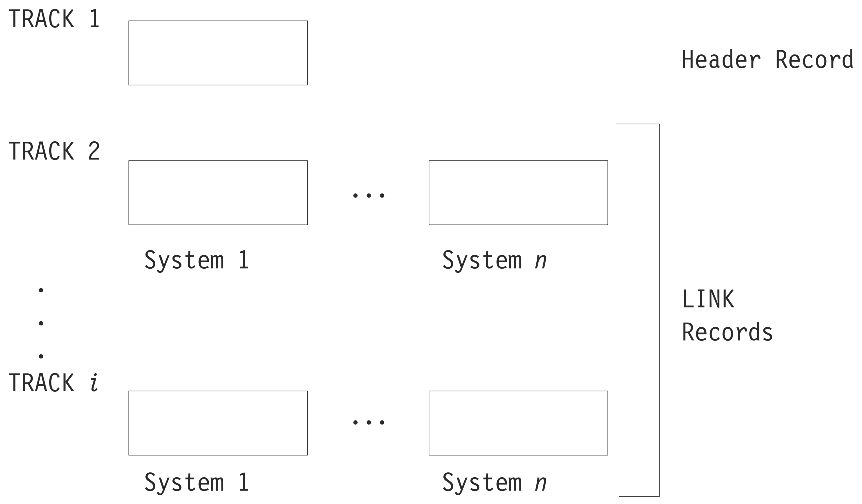


Figure 53. Format of an ECKD-Formatted CSE Area

An ECKD-formatted CSE area contains the following records:

1 Header Record
Contains 56 bytes of general cross-system link information.

***n* × *i* Link Records**
Each record is 256 bytes long and contains link information for 512 cylinders on the volume. Record X on track 2 represents cylinders 0–511 for system X, record X on track 3 represents cylinders 512–1023 for system X, and so on.

n
is the number of link records on each track, which is the number of systems defined as sharing minidisks with cross-system link. If it is necessary to change this value, use the MAP_RECORDS operand on the XLINK_DEVICE_DEFAULTS statement (see "XLINK_DEVICE_DEFAULTS Statement" on page 324) or the XLINK_VOLUME_INCLUDE statement (see "XLINK_VOLUME_INCLUDE Statement" on page 333).

Note: For some devices, the CSE area does not fit on one cylinder, so multiple cylinders are used. Table 54 on page 852 indicates the number of cylinders used for each device.

i
is the number of tracks needed to hold link records with link information for at least the number of cylinders on this volume. If it is necessary to change this value, use the MAP_RECORD_LENGTH operand on the XLINK_DEVICE_DEFAULTS statement (see "XLINK_DEVICE_DEFAULTS Statement" on page 324) or the XLINK_VOLUME_INCLUDE statement (see "XLINK_VOLUME_INCLUDE Statement" on page 333).

The link records show the state of the existing links on an ECKD DASD for the systems participating in cross-system link. Figure 54 on page 851 is an example.

```
                    Byte      0
                    Position  0   1   2   3   4   5   6   7   8   9

Link record 1, for system1:  0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 ...

Link record 2, for system2:  0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 2 0 0 0 0 ...

Link record 3, for system3:  1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 2 0 0 0 0 ...

Link record 4, for system4:  3 0 0 0 0 0 0 0 0 0 0 4 0 0 0 2 0 0 0 6 ...
```

Figure 54. Sample Link Records for Four Systems in Track 2 of an ECKD-Formatted CSE Area

The following values are used in the link records to indicate the link state flags:

Value	**Link State Flag**
0	NOLINK
1	READ
4	WRITE
2	STABLE READ
3	STABLE WRITE
5	EXCLUSIVE READ
6	EXCLUSIVE WRITE

The link records shown in Figure 54 on page 851 indicate the following links:

- System1 has write links to a minidisk beginning at cylinder 3 and stable read links to a minidisk beginning at cylinder 15.
- System2 has an exclusive read link to a minidisk beginning at cylinder 6 and stable read links to a minidisk beginning at cylinder 15.
- System3 has read links to minidisks beginning at cylinders 0, 3, and 11, and stable read links to a minidisk beginning at cylinder 15.
- System4 has a stable write link to a minidisk beginning at cylinder 0, write links to a minidisk beginning at cylinder 11, stable read links to a minidisk beginning at cylinder 15, and an exclusive write link to a minidisk beginning at cylinder 19.

Default Values for the CSE Area

By default, the XLINK FORMAT utility formats each CKD DASD volume with the maximum number of map records that can be recorded on one track. The same defaults are used for ECKD DASD, although the maximum number allowed is higher. Table 54 on page 852 shows, by device type, the default number of map records (which is also the number of systems that can be supported for cross-system link), and the default map record lengths.

Table 54. Defaults for the CSE Area

Device Type	Number of Map Records	Map Record Length	Number of cylinders needed for CSE area
3330 (3350 in 3330 compatibility mode)	8	808	1
3340	4	696	1
3350	15	555	1
3375	12	959	1
3380 type K	8	2655	1
3380 non-type K	11	1770	1
3390-1	12	1113	1
3390-2	8	2226	1
3390-3	8	3339	1
3390-9	56	65520	9
9345-1	12	1440	1
9345-2	8	2156	1
Other	8	1024	1

Note:

1. These defaults support CKD and ECKD DASD. FBA DASD are not supported.
2. The length of each map record is the maximum number of cylinders on any currently existing model of that device type.
3. The "Number of cylinders needed for CSE area" value is for ECKD DASD only. If the last cylinder on the volume is specified for the CSE area and multiple cylinders are needed, the CSE area will not wrap to the beginning of the volume for the other CSE area cylinders.
4. Eight records are defined for other devices, but the actual number of records used is limited to the number that fit on one track on such a device.

Notices

Programming Interface Information

This publication primarily documents information that is NOT intended to be used as Programming Interfaces of z/VM.

This publication also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/VM. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

PI

<...Programming Interface information...>

PI end

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

Environmental Record Editing and Printing Program

- Environmental Record Editing and Printing Program (EREP): Reference (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ifc2000_v2r5.pdf), GC35-0152
- Environmental Record Editing and Printing Program (EREP): User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ifc1000_v2r5.pdf), GC35-0151

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)

Additional Publications

- *System z Input/Output Configuration Program User's Guide for ICP IOCP (publibz.boulder.ibm.com/epubs/pdf/b107037b.pdf)*, SB10-7177

Index

Product Number: 5741-A09

Printed in USA

SC24-6271-73