

z/VM



# CP Planning and Administration

*Version 5 Release 1*



z/VM



# CP Planning and Administration

*Version 5 Release 1*

**Note:**

Before using this information and the product it supports, read the information under “Notices” on page 781.

**Second Edition (December 2004)**

This edition applies to version 5, release 1, modification 0 of IBM z/VM (product number 5741-A05) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC24-6083-00.

**© Copyright International Business Machines Corporation 1991, 2004. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>About This Book</b> . . . . .	xv
Who Should Read This Book . . . . .	xv
What You Should Know before Reading This Book . . . . .	xv
How to Read Syntax Diagrams . . . . .	xvi
Where to Find More Information . . . . .	xviii
How to Send Your Comments to IBM . . . . .	xviii
<b>Summary of Changes</b> . . . . .	xix
SC24-6083-01, z/VM Version 5 Release 1, Updated Edition . . . . .	xix
Fibre Channel Protocol (FCP) Logical Unit Number (LUN) Access Control (APAR VM63328) . . . . .	xix
Support for the IBM TotalStorage DS8000 (APAR VM63534) . . . . .	xix
SC24-6083-00, z/VM Version 5 Release 1 . . . . .	xix
SCSI FCP Disk Support . . . . .	xix
PCIX Cryptographic Coprocessor (PCIXCC) . . . . .	xix
Enhanced Guest LAN and Virtual Switch Authorization Support . . . . .	xix
Networking Enhancements . . . . .	xix
Hyperswap Support . . . . .	xx
Removal of CP Functions . . . . .	xx
Converting to System Configuration Files . . . . .	xx
Removal of Obsolete 370 References . . . . .	xx
SC24-6043-01, z/VM Version 4 Release 4 . . . . .	xx
Networking Enhancements . . . . .	xx
Hardware Configuration Definition (HCD) . . . . .	xxi
CP Command Response Suppression . . . . .	xxi
Guest IPL from SCSI Disks . . . . .	xxi
Miscellaneous . . . . .	xxi

---

## Part 1. Overview . . . . . 1

<b>Chapter 1. Planning and Administration Overview</b> . . . . .	3
Migration Planning . . . . .	3
Major z/VM Files . . . . .	3
System Planning and Administration . . . . .	4
User Planning and Administration . . . . .	5
Real and Virtual Storage Planning and Administration . . . . .	7
Auxiliary Storage (Expanded Storage and DASD) Planning and Administration . . . . .	7
CP-Owned DASDs . . . . .	7
Dedicated DASDs . . . . .	8
DASDs Used for Minidisks . . . . .	8
DASD Space Used for Shared File Pools . . . . .	9
Expanded Storage . . . . .	9
Virtual Disks in Storage . . . . .	9
Terminals . . . . .	10
Unit Record Devices . . . . .	11
Dedicated Unit Record Devices . . . . .	11
Spooled Unit Record Devices . . . . .	11
Performance Planning and Administration . . . . .	12
I/O Reconfiguration in z/VM . . . . .	12
OpenExtensions Planning . . . . .	13

---

## Part 2. System Planning and Administration. . . . . 15

<b>Chapter 2. Configuring Your System</b> . . . . .	23
Specifying System Configuration Information . . . . .	23
Using Configuration Files . . . . .	23
What You Can Specify in the System Configuration File . . . . .	24
Contents of the System Configuration File . . . . .	25
What You Can Specify in the Logo Configuration File . . . . .	28
Contents of the Logo Configuration File . . . . .	28
<b>Chapter 3. Understanding the CP File System</b> . . . . .	31
Initial State of the CP File System . . . . .	31
Changing the List of Disks Accessed by CP . . . . .	32
Performance Considerations . . . . .	33
Displaying the Contents of CP-Accessed Minidisks . . . . .	33
Changing Information on CP-Accessed Minidisks . . . . .	34
<b>Chapter 4. Using the Stand-Alone Program Loader</b> . . . . .	37
Creating the Stand-Alone Program Loader . . . . .	37
Overriding the Console that CP Will Use . . . . .	39
Overriding Stand-Alone Program Loader Defaults . . . . .	39
To IPL from a SCSI Disk . . . . .	39
The Stand-Alone Program Loader Screen . . . . .	40
Passing IPL Parameters . . . . .	42
<b>Chapter 5. Defining I/O Devices</b> . . . . .	45
Device Support . . . . .	45
Unsupported Devices . . . . .	46
Device Sensing . . . . .	46
DASD . . . . .	46
Tapes . . . . .	47
Printers . . . . .	47
Unit Record Devices . . . . .	48
Displays . . . . .	49
Communication Controllers . . . . .	49
ESCON Devices . . . . .	49
Other Devices . . . . .	50
<b>Chapter 6. The System Configuration File</b> . . . . .	51
The Parm Disk . . . . .	51
Summary of System Configuration File Statements . . . . .	51
General Rules for Coding a System Configuration File . . . . .	55
Format . . . . .	55
Comments . . . . .	55
Continuations . . . . .	55
Case . . . . .	56
Record Qualifiers . . . . .	56
The Order of Statements in the File . . . . .	57
Checking the Syntax of Statements in the File . . . . .	57
ALTERNATE_OPERATORS Statement . . . . .	58
ASSOCIATE EXIT Statement . . . . .	60
ASSOCIATE MESSAGES / MSGS Statement . . . . .	64
CHARACTER_DEFAULTS Statement . . . . .	67
CP_ACCESS Statement . . . . .	70
CP_ADDON_INITIALIZE_ROUTINES Statement . . . . .	72
CP_OWNED Statement . . . . .	73
CPXLOAD Statement . . . . .	76
DEFINE ALIAS Statement . . . . .	81

DEFINE COMMAND / CMD Statement . . . . .	84
DEFINE DIAGNOSE Statement. . . . .	90
DEFINE EXIT Statement . . . . .	94
DEFINE LAN Statement . . . . .	97
DEFINE VSWITCH Statement. . . . .	100
DEVICES Statement . . . . .	105
DISABLE COMMAND / CMD Statement . . . . .	110
DISABLE DIAGNOSE Statement. . . . .	112
DISABLE EXITS Statement. . . . .	114
DISTRIBUTE Statement . . . . .	116
DRAIN (Disk) Statement . . . . .	118
EDEVICE Statement . . . . .	121
EMERGENCY_MESSAGE_CONSOLES Statement . . . . .	124
ENABLE COMMAND / CMD Statement . . . . .	126
ENABLE DIAGNOSE Statement . . . . .	128
ENABLE EXITS Statement . . . . .	130
ENFORCE_BY_VOLid Statement . . . . .	132
EQUATE Statement . . . . .	133
EXTERNAL_SYNTAX Statement. . . . .	135
FEATURES Statement . . . . .	136
FORM_DEFAULT Statement . . . . .	150
HOT_IO_RATE Statement . . . . .	152
IMBED Statement . . . . .	155
INIT_MITIME Statement . . . . .	157
IODF Statement . . . . .	158
JOURNALING Statement . . . . .	160
LOGO_CONFIG Statement. . . . .	164
MODIFY COMMAND / CMD Statement . . . . .	165
MODIFY DIAGNOSE Statement . . . . .	169
MODIFY EXIT Statement. . . . .	172
MODIFY LAN Statement. . . . .	175
MODIFY PRIV_CLASSES Statement . . . . .	176
MODIFY VSWITCH Statement . . . . .	178
OPERATOR_CONSOLES Statement . . . . .	180
PRINTER_TITLE Statement . . . . .	182
PRIV_CLASSES Statement. . . . .	184
PRODUCT Statement . . . . .	186
RDEVICE Statement . . . . .	188
RDEVICE Statement (Advanced Function Printers) . . . . .	189
RDEVICE Statement (Card Punches) . . . . .	190
RDEVICE Statement (Card Readers) . . . . .	192
RDEVICE Statement (Communication Controllers) . . . . .	193
RDEVICE Statement (DASD) . . . . .	195
RDEVICE Statement (Graphic Display Devices) . . . . .	197
RDEVICE Statement (Impact Printers). . . . .	201
RDEVICE Statement (Special Devices) . . . . .	205
RDEVICE Statement (Tape Units) . . . . .	207
RDEVICE Statement (Unsupported Devices) . . . . .	209
RDEVICE Statement (3800 Printers) . . . . .	212
SAY Statement . . . . .	217
SET Statement . . . . .	219
START (Disk) Statement . . . . .	220
STORAGE Statement . . . . .	222
SYSTEM_DATEFORMAT Statement . . . . .	225
SYSTEM_IDENTIFIER Statement . . . . .	226
SYSTEM_IDENTIFIER_DEFAULT Statement . . . . .	228

SYSTEM_RESIDENCE Statement . . . . .	230
SYSTEM_USERIDS Statement . . . . .	232
THROTTLE Statement . . . . .	236
TIMEZONE_BOUNDARY Statement . . . . .	237
TIMEZONE_DEFINITION Statement . . . . .	239
TOLERATE_CONFIG_ERRORS Statement . . . . .	241
TRANSLATE_TABLE Statement . . . . .	243
USERFORM Statement . . . . .	247
USER_DEFAULTS System Configuration File Statement . . . . .	248
USER_VOLUME_EXCLUDE Statement . . . . .	251
USER_VOLUME_INCLUDE Statement . . . . .	253
USER_VOLUME_LIST Statement . . . . .	255
VMLAN Statement . . . . .	257
XLINK_DEVICE_DEFAULTS Statement . . . . .	259
XLINK_SYSTEM_EXCLUDE Statement . . . . .	263
XLINK_SYSTEM_INCLUDE Statement . . . . .	264
XLINK_VOLUME_EXCLUDE Statement . . . . .	266
XLINK_VOLUME_INCLUDE Statement . . . . .	268
XSPPOOL_SYSTEM Statement. . . . .	271
XSPPOOL_TRACE Statement . . . . .	273
XSPPOOL_XLIST_INPUT Statement . . . . .	274
XSPPOOL_XLIST_OUTPUT Statement . . . . .	276
<b>Chapter 7. The Logo Configuration File . . . . .</b>	<b>279</b>
Using a Logo Configuration File . . . . .	279
Summary of Logo Configuration File Statements . . . . .	279
General Rules for Coding a Logo Configuration File . . . . .	279
Format . . . . .	279
Comments . . . . .	279
Continuations . . . . .	280
Case . . . . .	280
Categories . . . . .	281
Precedence . . . . .	281
Selection . . . . .	282
Creating Logo Screens . . . . .	282
Special Considerations When Creating Logos . . . . .	283
CHOOSE_LOGO Statement . . . . .	285
INPUT_AREA Statement. . . . .	291
ONLINE_MESSAGE Statement . . . . .	292
STATUS Statement. . . . .	293
<b>Chapter 8. Setting Up Service Virtual Machines . . . . .</b>	<b>295</b>
Setting Up Virtual Machines for Accounting . . . . .	295
Setting Up Virtual Machines to Collect Accounting Records . . . . .	297
Specifying a New Accounting Virtual Machine . . . . .	297
Starting Manual Retrieval of Accounting Records . . . . .	297
Disassociating a User ID from the Retrieval of Accounting Records . . . . .	298
Accounting Record Formats. . . . .	299
Adding Your Own Accounting Records and Source Code . . . . .	310
Setting Up Virtual Machines for Error Recording . . . . .	314
Setting Up Virtual Machines to Collect EREP Records . . . . .	315
Specifying a New EREP Virtual Machine . . . . .	316
Starting EREP Record Retrieval Manually . . . . .	316
Disassociating a User ID from the Retrieval of EREP Records . . . . .	317
Setting Up Virtual Machines for Symptom Record Recording . . . . .	318
Setting Up Virtual Machines to Collect Symptom Records. . . . .	319



Specifying a New Symptom Record Recording Virtual Machine. . . . .	319
Starting Manual Retrieval of Symptom Records . . . . .	320
Disassociating a User ID from the Retrieval of Symptom Records. . . . .	320
Setting Up a Virtual Machine for Communication Controller Support for Emulator Program (EP) . . . . .	321
PVM Example. . . . .	323
Setting Up Service Pool Virtual Machines. . . . .	325
Setting Up Print Services Facility/VM Virtual Machines . . . . .	326
Setting Up Virtual Machines for Data Storage Management . . . . .	327
<b>Chapter 9. Planning for SNA Console Communication Services (SNA/CCS) . . . . .</b>	<b>329</b>
Structure of the SNA Environment . . . . .	329
Establishing the SNA/CCS Terminal Environment. . . . .	330
Defining the VSM to z/VM . . . . .	330
Defining Logos Used by VSMS . . . . .	330
Defining SNA/CCS to VCNA . . . . .	330
Defining SNA/CCS to VSCS . . . . .	331
Enabling SNA Communication. . . . .	331
Starting the VTAM Service Machine. . . . .	332
Improving SNA/CCS Performance . . . . .	332
VSM Termination. . . . .	332
<b>Chapter 10. Setting Up Cross System Extensions (CSE) . . . . .</b>	<b>333</b>
Overview . . . . .	333
CSE Capabilities. . . . .	333
Users Supported. . . . .	334
Setting Up a CSE Complex. . . . .	334
Planning for CSE . . . . .	334
Requirements . . . . .	334
Restrictions. . . . .	335
Supported Features. . . . .	335
Sharing DASD Volumes . . . . .	336
Spool File Directory Statements . . . . .	341
Preparing for CSE . . . . .	341
Enabling CSE . . . . .	345
Installing the VM/Pass-Through Facility and Its Modifications . . . . .	345
Defining CP for Each System . . . . .	346
Initializing the Volumes for Cross System Link . . . . .	349
Preparing the Communication Virtual Machine (CVM) . . . . .	350
Verifying Your CSE Configuration. . . . .	353
Verifying Cross System Link . . . . .	353
Enabling Cross System Spool . . . . .	353
Phasing CSE into the Production Environment. . . . .	356
Administering CSE . . . . .	358
How Linking Works in z/VM. . . . .	358
How CSE Extends Linking . . . . .	359
CSE Area . . . . .	359
VM I/O Performance Measurement for Shared DASD . . . . .	365
What Is Cross System Spool? . . . . .	366
How CSE Extends Spooling . . . . .	367
VM/Pass-Through Facility Functions for Cross System Spool . . . . .	371
Multiple Concurrent Logon Sessions . . . . .	371
Preparing to Deal with Outages . . . . .	372
Returning to Non-CSE Operation. . . . .	372

<b>Chapter 11. Customizing the CP Message Function . . . . .</b>	<b>373</b>
HCPMSU Module . . . . .	373
Entry Point HCPMSUEX . . . . .	374
Changing and Adding to the CP Message Function . . . . .	377
<b>Chapter 12. Security and Integrity in z/VM . . . . .</b>	<b>379</b>
Security-Enhancing Products . . . . .	379
Security Considerations and Guidelines . . . . .	380
z/VM Security Facilities . . . . .	383
Using an External Security Manager for Auditing and Protecting . . . . .	384
Verifying Storage Access . . . . .	384
Clearing Temporary Disk Space . . . . .	384
Permitting Bypassing of Directory Password Authorization . . . . .	385
Journaling the LOGON, AUTOLOG, XAUTOLOG, and LINK Commands . . . . .	385
Automatic Deactivation of Restricted Passwords . . . . .	386
Using Link Access Control Options . . . . .	387
Suppressing Passwords Entered on the Command Line . . . . .	387
Using an Integrated Cryptographic Facility . . . . .	387
Maintaining System Integrity . . . . .	389
z/VM System Integrity Requirements . . . . .	390
z/VM Integrity and CP Function . . . . .	392
Data Structures That Can Enhance System Integrity . . . . .	395
z/VM Options That Can Enhance System Integrity . . . . .	397
Storage Handling . . . . .	397
Program Stack, Security, and Integrity . . . . .	398
Reporting z/VM Integrity Problems . . . . .	400
<b>Chapter 13. Using the Stand-Alone Dump Utility . . . . .</b>	<b>403</b>
Overview . . . . .	403
Creating the Stand-Alone Dump Utility . . . . .	403
Before You Begin . . . . .	404
Devices You Can Use to IPL a Stand-Alone Dump . . . . .	405
Devices to Which You Can Send Dump Output . . . . .	406
Example for Generating the Stand-Alone Dump Utility . . . . .	406
Taking a Stand-Alone Dump . . . . .	408
Processing the Stand-Alone Dump Data on Tape . . . . .	409
The HCPSADMP EXEC . . . . .	409
Usage Notes . . . . .	409
<b>Chapter 14. Creating and Modifying Image Libraries for Printers. . . . .</b>	<b>411</b>
Creating Text Decks . . . . .	411
Creating Text Decks for the 3800 . . . . .	411
Creating Text Decks for Impact Printers . . . . .	412
Using VMFHASM or VMFHLASM to Create a Text Deck . . . . .	422
Image Libraries . . . . .	422
Default Image Library Names . . . . .	422
Installing the Image Library That IBM Provides . . . . .	423
Installing Your Own Image Library . . . . .	423
Modifying Image Libraries . . . . .	424
Displaying Information about Image Libraries . . . . .	424
Purging Image Libraries . . . . .	425
Keeping Backup Copies of Image Libraries . . . . .	425
Where to Find More Information about Printers . . . . .	425
<b>Chapter 15. CCW Translation . . . . .</b>	<b>427</b>
Using CCW Translation . . . . .	427

Coding the Device Class Macroinstruction . . . . .	427
Unsupported Devices Tables . . . . .	428
Device Macroinstruction . . . . .	429

---

**Part 3. User Planning and Administration . . . . . 431**

<b>Chapter 16. Redefining Command Privilege Classes . . . . .</b>	<b>433</b>
IBM-Defined Privilege Classes. . . . .	433
Planning a New User Class Structure . . . . .	434
Defining Users' Needs. . . . .	434
Assigning Commands to Types of Users . . . . .	435
Associating Privilege Classes with Users and Commands. . . . .	436
Further Considerations . . . . .	437
Preparing and Activating the Override File . . . . .	438
Creating a Class Override File. . . . .	438
Class Override File Example . . . . .	439
Verifying and Activating the Override File . . . . .	441
Querying the User Class Restructure File. . . . .	442
Changing Back to the IBM-Defined User Classes. . . . .	442
Another Way of Changing the Privilege Class of Certain CP Functions	443
Changing the Directory . . . . .	443
Defining Privilege Classes for a Virtual Machine . . . . .	444
How Users Can Find Which Commands They Can Enter . . . . .	446
<b>Chapter 17. Creating and Updating a User Directory . . . . .</b>	<b>447</b>
Overview . . . . .	447
Creating the User Directory . . . . .	451
Continued Directory Control Statements . . . . .	452
Quoted String Operands . . . . .	452
Running the User Directory Program . . . . .	453
Changing the Directory . . . . .	454
Checking a Directory for Errors . . . . .	454
Specifying a Directory That Contains VM/SP Control Statements . . . . .	455
Creating Directory Profiles . . . . .	455
Determining How Much Space the Directory Needs . . . . .	455
ACCOUNT Directory Control Statement (General) . . . . .	456
ACIGROUP Directory Control Statement (General) . . . . .	458
APPCPASS Directory Control Statement (General) . . . . .	459
AUTOLOG Directory Control Statement (General) . . . . .	461
CLASS Directory Control Statement (General) . . . . .	462
CONSOLE Directory Control Statement (Device) . . . . .	463
CPU Directory Control Statement (General) . . . . .	465
CRYPTO Directory Control Statement (General) . . . . .	468
DASDOPT Directory Control Statement (Device) . . . . .	472
DATEFORMAT Directory Control Statement (General) . . . . .	475
DEDICATE Directory Control Statement (Device) . . . . .	477
DIRECTORY Directory Control Statement (Control) . . . . .	480
D8ONECMD Directory Control Statement (General) . . . . .	483
GLOBALDEFS Directory Control Statement (Control) . . . . .	485
GLOBALOPTS Directory Control Statement (Control) . . . . .	486
INCLUDE Directory Control Statement (General) . . . . .	487
IOPRIORITY Directory Control Statement (General) . . . . .	488
IPL Directory Control Statement (General) . . . . .	490
IUCV Directory Control Statement (General) . . . . .	493
LINK Directory Control Statement (Device) . . . . .	500
LOAD Directory Control Statement (Control) . . . . .	504

LOADDEV Directory Control Statement (General)	507
LOGONBY Directory Control Statement (General)	509
MACHINE Directory Control Statement (General)	510
MAXSTORAGE Directory Control Statement (General)	512
MDISK Directory Control Statement (Device)	513
MINIOPT Directory Control Statement (Device)	524
NAMESAVE Directory Control Statement (General)	527
NICDEF Directory Control Statement (Device)	529
NOPDATA Directory Control Statement (General)	532
OPTION Directory Control Statement (General)	533
POOL Directory Control Statement (General)	540
POSIXGLIST Directory Control Statement (General)	541
POSIXGROUP Directory Control Statement (Control)	543
POSIXINFO Directory Control Statement (General)	545
POSIXOPT Directory Control Statement (General)	548
PROFILE Directory Control Statement (Control)	551
SCREEN Directory Control Statement (General)	553
SHARE Directory Control Statement (General)	555
SPECIAL Directory Control Statement (Device)	557
SPOOL Directory Control Statement (Device)	562
SPOOLFILE Directory Control Statement (General)	565
STDEVOPT Directory Control Statement (General)	566
STORAGE Directory Control Statement (General)	568
SYSAFFIN Directory Control Statement (Control)	569
USER Directory Control Statement (Control)	572
XAUTOLOG Directory Control Statement (General)	578
XCONFIG Directory Control Statement (General)	580
XCONFIG ACCESSLIST Operand	581
XCONFIG ADDRSPACE Operand	583
XSTORE Directory Control Statement (General)	585

---

## **Part 4. Storage Planning and Administration** . . . . . 587

<b>Chapter 18. Real Storage Planning and Administration</b>	589
Planning Real Storage Needs	589
Real and Virtual Storage Requirements for Generating a CP Module	589
Determining the Size of the Dynamic Paging Area	590
Virtual Machine Considerations	590
<b>Chapter 19. Expanded Storage Planning and Administration</b>	591
Overview	591
Expanded Storage Considerations with Shared CMS Minidisks	592
Attaching Expanded Storage to a Virtual Machine	592
Mapping Expanded Storage	593
Fragmentation	593
Discontinuous Expanded Storage	593
Allocating Expanded Storage to CP	596
Allocating Retained Expanded Storage	596
Pending Retained Expanded Storage	597
Using Expanded Storage as a Minidisk Cache	597
What Devices Can Be Cached	600
Turning Caching Off	600
Planning for and Using Minidisk Caching	600
<b>Chapter 20. Allocating DASD Space</b>	603
Direct Access Storage Requirements	603

CKD Device Geometry . . . . .	603
FBA Device Geometry . . . . .	604
Storage Calculations . . . . .	604
CP Module . . . . .	605
Warm Start Data . . . . .	606
Example . . . . .	607
Checkpoint Data . . . . .	607
Directory Space . . . . .	608
Example . . . . .	609
Directory Size Constraints . . . . .	610
Paging Space . . . . .	610
Example . . . . .	611
Spooling Space . . . . .	613
Allocating Space for CP Abend Dumps . . . . .	614
Named Saved System . . . . .	615
Spool Space Example . . . . .	615
Switching Operating Modes for 3390 Devices . . . . .	615
Mode Switch Procedure . . . . .	616
Migration Considerations for the 3390 Model 9 . . . . .	617
<b>Chapter 21. DASD Sharing . . . . .</b>	<b>619</b>
Sharing DASD among Multiple Virtual Machines by Using Virtual Reserve/Release . . . . .	619
When to Use Virtual Reserve/Release . . . . .	621
Sharing DASD without Using Virtual Reserve/Release . . . . .	622
Sharing DASD Using the CMS Shared File System . . . . .	622
Sharing DASD between One Virtual Machine and Other Systems Using Real Reserve/Release . . . . .	622
When to Use Real Reserve/Release . . . . .	624
Sharing DASD among Multiple Virtual Machines and Other Systems Using Concurrent Virtual and Real Reserve/Release . . . . .	624
When to Use Concurrent Virtual and Real Reserve/Release . . . . .	626
Restrictions for Using Reserve/Release . . . . .	626
Reserve/Release Summary . . . . .	626
Sharing DASD using the Multi-Path Lock Facility . . . . .	627
Cached DASD . . . . .	628
Cache Control . . . . .	628
Defining a Minidisk on a Cached DASD . . . . .	629
Defining a Cached DASD as a Dedicated Device . . . . .	630
Using ESS Parallel Access Volumes . . . . .	631
Using PAV for Workload Balancing . . . . .	631
Using PAV for DASD Sharing . . . . .	632
z/VM Restrictions on Using PAV . . . . .	633
<b>Chapter 22. Defining and Managing SCSI FCP Disks . . . . .</b>	<b>635</b>
Overview . . . . .	635
Defining SCSI Devices . . . . .	637
Emulated FBA Disks on SCSI Disks . . . . .	637
Real SCSI Disks . . . . .	638
Additional Considerations . . . . .	639
<b>Appendix A. Sample Utility Programs . . . . .</b>	<b>641</b>
DRAWLOGO . . . . .	642
<b>Appendix B. Defining Your System (HCPSYS Macroinstructions) . . . . .</b>	<b>643</b>
Coding HCPSYS ASSEMBLE . . . . .	644

CSELDEV (Optional)	646
CSELVOL EXCLUDE (Optional)	649
CSELVOL INCLUDE (Optional)	650
CSESYS (Optional)	653
CSETRACE (Optional)	658
CSEUSER (Optional)	659
SYSACNT (Optional)	661
SYSADDIN (Optional)	663
SYSCPVOL (Required)	664
SYSDUMP (Optional)	667
SYSEND (Required)	668
YSEREP (Optional)	669
SYSEXCL (Optional)	671
SYSFCN (Optional)	673
SYSFORM (Optional)	675
SYSID (Optional)	677
SYSINCL (Optional)	680
SYSJRL (Optional)	682
SYSMAXU (Optional)	686
SYSOPR (Required)	687
SYSOPTS (Optional)	689
SYSPCLAS (Optional)	691
SYSRES (Required)	693
SYSSTORE (Required)	697
SYSSYMP (Optional)	699
SYSTIME (Required)	701
SYSUVOL (Optional)	703
<b>Appendix C. Defining I/O Devices Using HCPRIO.</b>	<b>705</b>
Coding HCPRIO ASSEMBLE	705
RDEVICE Macroinstructions	706
RDEVICE—General Usage Notes	707
Dedicated AFP* Printers	707
Common Control Unit (CCU) Printers	708
3820 Printer	710
Printers Attached to a Display Control Unit	712
Display Printers	713
CP System-Managed Impact Printers	715
Impact Printers	716
CP System-Managed 3800 Printers	720
3800 Printer	721
Unit Record Devices	726
3505 Card Reader	727
3525 Card Punch	729
3890 Document Processor	731
Display Terminals and Adapters	732
3250 Display	733
3270-Family Displays	734
Teleprocessing Integrated Adapters	738
5080 and 6090 Graphics Display	740
Tape Units	742
3422, 3480, 3490 and 3590 Tape Drives	743
DASD	745
DASD Types	746
FBA DASD	749
Communication Devices	751

2701 Data Adapter . . . . .	752
3705, 3725, and 3745 Line Adapters, 2741 Terminal, and 3151, 3161, 3162, 3163, and 3167 Display Terminals . . . . .	754
37xx Communications Controllers and 3x74 Controllers . . . . .	757
CTCA, 3088 Multisystem Channel Communication Unit, and 3737 Remote Channel-to-Channel Unit . . . . .	760
Dynamic Switching Devices. . . . .	761
9032 and 9033 ES Connection Directors (ESCDs) . . . . .	762
Unsupported Devices . . . . .	763
RIOGEN. . . . .	766
<b>Appendix D. Configuration Guide for Printers . . . . .</b>	<b>769</b>
<b>Appendix E. Device Class and Type Codes . . . . .</b>	<b>773</b>
Device Class Definitions . . . . .	773
Device Type Definitions within Device Classes . . . . .	773
Device Features by Class and Type. . . . .	775
Device Model Definitions . . . . .	775
<b>Appendix F. Stand-Alone Dump Formats . . . . .</b>	<b>777</b>
Tape Format . . . . .	777
DASD Format. . . . .	779
Printer Format. . . . .	780
<b>Notices . . . . .</b>	<b>781</b>
Programming Interface Information . . . . .	782
Trademarks. . . . .	783
<b>Glossary . . . . .</b>	<b>785</b>
<b>Bibliography . . . . .</b>	<b>787</b>
Where to Get z/VM Books . . . . .	787
z/VM Base Library . . . . .	787
System Overview . . . . .	787
Installation and Service . . . . .	787
Planning and Administration. . . . .	787
Customization. . . . .	787
Operation . . . . .	787
Application Programming. . . . .	787
End Use . . . . .	788
Diagnosis . . . . .	788
Books for z/VM Optional Features . . . . .	788
Data Facility Storage Management Subsystem for VM . . . . .	788
Directory Maintenance Facility . . . . .	788
Performance Toolkit for VM . . . . .	789
Resource Access Control Facility. . . . .	789
<b>Index . . . . .</b>	<b>791</b>





---

## About This Book

This book tells you how to plan and administer your IBM® z/VM® system. It describes the following tasks:

- System planning and administration
- Dynamic I/O configuration planning
- User planning and administration
- Storage planning and administration
- Saved Segment planning and administration
- CMS planning and administration.

System planning and administration are tasks that overlap. Before installation of 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 for the system. 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 control 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 publication are meant to help installations run operating systems efficiently under z/VM.

---

## Who Should Read This Book

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

---

## What You Should Know before Reading This Book

The reader is expected to have a general understanding of data processing and teleprocessing techniques. This book 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 what sort of environment they will be running their applications under.

---

## How to Read Syntax Diagrams


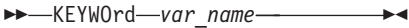

This book uses syntax diagrams to show the operands and options of external interfaces and statements.

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

- The  $\blacktriangleright$ — symbol indicates the beginning of a syntax diagram.
- The — $\blacktriangleright$  symbol, at the end of a line, indicates that the syntax diagram continues on the next line.
- The  $\blacktriangleright$ — symbol, at the beginning of a line, indicates that a syntax diagram continues from the previous line.
- The — $\blacktriangleleft$  symbol indicates the end of a syntax diagram.

Syntax items (for example, a keyword or variable) may be:

- Directly on the line (required)
- Above the line (default)
- Below the line (optional).

Syntax Diagram Description	Example
<p><b>Abbreviations:</b></p> <p>Uppercase letters denote the shortest acceptable abbreviation. If an item appears entirely in uppercase letters, it cannot be abbreviated.</p> <p>You can type the item in uppercase letters, lowercase letters, or any combination.</p> <p>In this example, you can enter KEYWO, KEYWOR, or KEYWORD in any combination of uppercase and lowercase letters.</p>	
<p><b>Symbols:</b></p> <p>You must code these symbols exactly as they appear in the syntax diagram.</p>	<ul style="list-style-type: none"> <li>* Asterisk</li> <li>:</li> <li>,</li> <li>= Equal Sign</li> <li>- Hyphen</li> <li>() Parentheses</li> <li>.</li> </ul>
<p><b>Variables:</b></p> <p>Highlighted lowercase items (<i>like this</i>) denote variables.</p> <p>In this example, <i>var_name</i> represents a variable you must specify when you code the KEYWORD instruction.</p>	
<p><b>Repetition:</b></p> <p>An arrow returning to the left means that the item can be repeated.</p>	

**Syntax Diagram Description**

**Example**

A character within the arrow means you must separate repeated items with that character.



A footnote (1) by the arrow references a limit that tells how many times the item can be repeated.



**Notes:**

1 Specify *repeat* up to 5 times.

**Required Choices:**

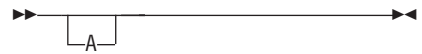
When two or more items are in a stack and one of them is on the line, you *must* specify one item.



In this example, you must choose A, B, or C.

**Optional Choice:**

When an item is below the line, the item 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:**

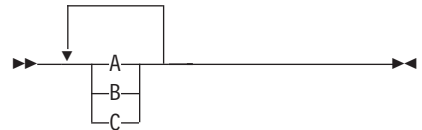
Defaults are above the line. The system uses the default unless you override it. You can override the default by coding an option from the stack below the line.



In this example, A is the default. You can override A by choosing B or C.

**Repeatable Choices:**

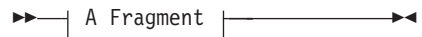
A stack of items followed by an arrow returning to the left means that you can select more than one item or, in some cases, repeat a single item.



In this example, you can choose any combination of A, B, or C.

**Syntax Fragments:**

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.



**A Fragment:**



In this example, the fragment is named "A Fragment."

---

## Where to Find More Information

For more information about z/VM functions, see the books listed in the “Bibliography” on page 787.

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

### Links to Other Online Books

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

---

## How to Send Your Comments to IBM

IBM welcomes your comments. You can send us comments about this book or other VM documentation using any of the following methods:

- Complete and mail the Readers' Comments form (if one is provided at the back of this book) or send your comments to the following address:

IBM Corporation  
Department 55JA, Mail Station P384  
2455 South Road  
Poughkeepsie, New York 12601-5400  
U.S.A.

FAX (United States and Canada): 1-845-432-9405  
FAX (Other Countries): +1 845 432 9405

- Send your comments by electronic mail to one of the following addresses:
  - Internet: [mhvrdfs@us.ibm.com](mailto:mhvrdfs@us.ibm.com)
  - IBMLink™ (US customers only): [IBMUSM10\(MHVRDFS\)](mailto:IBMUSM10(MHVRDFS)@us.ibm.com)
- Submit your comments through the VM Feedback page (“Contact z/VM”) on the z/VM Web site at [www.ibm.com/eserver/zseries/zvm/forms/](http://www.ibm.com/eserver/zseries/zvm/forms/).

Please provide the following information in your comment or note:

- Title and complete publication number of the book (including the suffix)
- Page number, section title, or topic you are commenting on

If you would like a reply, be sure to include your name, postal or email address, and telephone or FAX number.

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

---

## Summary of Changes

This book contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

---

### SC24-6083-01, z/VM Version 5 Release 1, Updated Edition

This edition includes support announced after the announcement of z/VM Version 5 Release 1 (z/VM V5R1) or programming enhancements provided after the general availability of z/VM V5R1. These programming enhancements may be provided through z/VM service by program temporary fixes (PTFs) for authorized program analysis reports (APARs), as indicated.

### Fibre Channel Protocol (FCP) Logical Unit Number (LUN) Access Control (APAR VM63328)

Updates in this book include:

- In Chapter 17, “Creating and Updating a User Directory,” on page 447, the DEDICATE statement was updated.

### Support for the IBM TotalStorage® DS8000 (APAR VM63534)

z/VM support for the high-capacity DS8000 allows the definition of SCSI FCP disks up to 2,147,483,640 512-byte blocks (1 terabyte minus 1 page) in size. A new 2107 attribute has been added to the EDEVICE system configuration statement.

---

### SC24-6083-00, z/VM Version 5 Release 1

This edition supports the general availability of z/VM Version 5 Release 1 (z/VM V5R1)

### SCSI FCP Disk Support

Updates in this book include:

- A new chapter, Chapter 22, “Defining and Managing SCSI FCP Disks,” on page 635, was added.
- In Chapter 6, “The System Configuration File,” on page 51, the EDEVICE statement was added.

### PCIX Cryptographic Coprocessor (PCIXCC)

Updates in this book include:

- In Chapter 17, “Creating and Updating a User Directory,” on page 447, the CRYPTO statement was updated.

### Enhanced Guest LAN and Virtual Switch Authorization Support

Updates in this book include:

- In Chapter 6, “The System Configuration File,” on page 51, the MODIFY LAN and MODIFY VSWITCH statements were updated.

### Networking Enhancements

Updates in this book include:

- In Chapter 6, “The System Configuration File,” on page 51, the DEFINE VSWITCH statement was updated.

## Hyperswap Support

Updates in this book include:

- In Chapter 6, “The System Configuration File,” on page 51, the new ENFORCE\_BY\_VOLID statement was added to the System Configuration File Statements.

## Removal of CP Functions

z/VM V5R1 is designed to operate only on IBM zSeries®, or equivalent servers that support IBM z/Architecture™ (64-bit). As a result, certain functions are not provided by z/VM V5R1:

- IPL from a 31-bit image of the CP nucleus
- Preferred (V=R and V=F) virtual machines
- Paging of the CP nucleus

Several system configuration statements and directory control statements have been updated. For information about the affected external interfaces, see the *z/VM: Migration Guide*.

## Converting to System Configuration Files

IBM assumes that you are using configuration files to define the characteristics of your z/VM system. Accordingly, most references to using the system control file (HCPSYS ASSEMBLE), system real I/O configuration file (HCPRIO ASSEMBLE), and system logo definition file (HCPBOX ASSEMBLE) have been removed. Although the HCPSYS and HCPRIO macroinstructions are still documented in the appendixes of this book, IBM discourages their use. If you are still using these macros to define your system, IBM strongly recommends that you convert to using configuration files. Using the ASSEMBLE files is more difficult and error-prone, requires knowledge of the Assembler H Version 2 Licensed Program, does not support recent CP enhancements, and requires rebuilding the CP module after making changes.

The “Converting to System Configuration Files” chapter has been removed from this book. See the *z/VM: Migration Guide* for information that can assist you in migrating your system definition data to configuration files.

## Removal of Obsolete 370 References

System/370™ architecture (370 mode) virtual machines are not supported on z/VM Version 4 or later. Obsolete 370 references have been removed. However, 370 Accommodation is still supported.

---

## SC24-6043-01, z/VM Version 4 Release 4

This edition supports the general availability of z/VM Version 4 Release 4 (z/VM V4R4).

## Networking Enhancements

Updates in this book include:

- In Chapter 6, “The System Configuration File,” on page 51, three new statements, DEFINE VSWITCH, MODIFY VSWITCH, and MODIFY LAN, were added to the System Configuration File Statements. Also, the VMLAN statement was updated.
- In Chapter 8, “Setting Up Service Virtual Machines,” on page 295, columns 35–42 and 43–50, under Accounting Records Network Data Transmissions (Record Type C), were updated.
- In Chapter 17, “Creating and Updating a User Directory,” on page 447, the IUCV Directory Control Statement and the SPECIAL Directory Control Statement were updated. A new statement, NICDEF Directory Control Statement, was added.

## Hardware Configuration Definition (HCD)

Hardware Configuration Definition (HCD), along with Hardware Configuration Manager (HCM), are components of z/VM that work together to create and manage your z/VM I/O configuration. Updates in this book include:

- Additional material in Chapter 1, “Planning and Administration Overview,” on page 3, Chapter 2, “Configuring Your System,” on page 23, and Chapter 5, “Defining I/O Devices,” on page 45 to reflect the new HCD support. While the original command-based dynamic I/O configuration capabilities of z/VM are still valid, the use of HCM and HCD is incompatible with those commands. You should choose one method or the other and stay with it for the duration of any given IPL of your z/VM system.
- A new system configuration file statement, IODF, which indicates that HCD will be used to control the I/O hardware and/or software configuration. See “IODF Statement” on page 158.
- In Chapter 4, “Using the Stand-Alone Program Loader,” on page 37, a new IPL option, NOHCD, which specifies that CP should ignore any IODF statements in the system configuration file.

For more information on HCD and HCM, see *z/VM: I/O Configuration*.

## CP Command Response Suppression

The SILENT operand has been added to the DEFINE COMMAND / CMD and MODIFY COMMAND / CMD system configuration statements to enable response suppression for the CP command being defined or redefined.

## Guest IPL from SCSI Disks

The directory control statement, LOADDEV, was added to Chapter 17, “Creating and Updating a User Directory,” on page 447. LOADDEV is used to identify the location of a program to be loaded as a result of an FCP–attached SCSI disk (Linux and other guest operating systems) IPL.

## Miscellaneous

- Information about using the CP Access Control Interface has been moved to the *z/VM: Service Guide*.
- A new parameter (FCTC) was added to the SPECIAL directory control statement. This supports Virtual FICON CTCA.





---

## Part 1. Overview

<b>Chapter 1. Planning and Administration Overview</b> . . . . .	3
Migration Planning . . . . .	3
Major z/VM Files. . . . .	3
System Planning and Administration . . . . .	4
User Planning and Administration . . . . .	5
Real and Virtual Storage Planning and Administration . . . . .	7
Auxiliary Storage (Expanded Storage and DASD) Planning and Administration	7
CP-Owned DASDs . . . . .	7
Dedicated DASDs . . . . .	8
DASDs Used for Minidisks . . . . .	8
DASD Space Used for Shared File Pools . . . . .	9
Expanded Storage . . . . .	9
Virtual Disks in Storage . . . . .	9
Terminals . . . . .	10
Unit Record Devices . . . . .	11
Dedicated Unit Record Devices . . . . .	11
Spooled Unit Record Devices . . . . .	11
Performance Planning and Administration . . . . .	12
I/O Reconfiguration in z/VM . . . . .	12
OpenExtensions Planning . . . . .	13



---

## Chapter 1. Planning and Administration Overview

This chapter 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 in this manual 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 chapter 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 chapter 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 787. 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 the *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 23, Chapter 5, “Defining I/O Devices,” on page 45, and Chapter 6, “The System Configuration File,” on page 51.

## Overview

- The logo configuration file (LOGO CONFIG) defines where CP can find:
  - Logos for local, logical, and VTAM<sup>®</sup>-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 23 and Chapter 7, “The Logo Configuration File,” on page 279.

- 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. (You do not need to name your directory USER DIRECT.)

For more information, see Chapter 17, “Creating and Updating a User Directory,” on page 447.

---

## System Planning and Administration

The following list briefly describes system 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 23 and Chapter 6, “The System Configuration File,” on page 51.
- **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 237 and “TIMEZONE\_DEFINITION Statement” on page 239.
- **Create a system identifier 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 identifier for each processor on which you run z/VM. The system identifier appears on printed output separator pages and in the status area of the display screen. For more information, see “SYSTEM\_IDENTIFIER Statement” on page 226 and “SYSTEM\_IDENTIFIER\_DEFAULT Statement” on page 228.
- **Set up service virtual machines** for accounting, error recording, symptom record recording, communication controller support for the emulator program, service spool support, Print Services Facility<sup>™</sup>/VM (PSF<sup>™</sup>/VM), and data storage management. Details on setting up these virtual machines can be found in Chapter 8, “Setting Up Service Virtual Machines,” on page 295.

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 329.
- **Plan for cross system extensions (CSE)** if users are to participate in a multisystem environment. CSE extends CP link protocols across multiple systems in a CSE complex. It controls read or read/write access for minidisks on CKD or ECKD<sup>(™)</sup> DASD, extends MESSAGE and QUERY commands, and manages

spool files for users across multiple z/VM systems. For additional information about CSE, refer to Chapter 10, “Setting Up Cross System Extensions (CSE),” on page 333.

- **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 12, “Security and Integrity in z/VM,” on page 379.
- **Improve system availability by defining and saving multiple copies of the CP module.** For more information, see Chapter 4, “Using the Stand-Alone Program Loader,” on page 37.
- **Create the stand-alone dump utility program and place it on tape or DASD.** For additional information on the stand-alone dump utility provided by z/VM, refer to Chapter 13, “Using the Stand-Alone Dump Utility,” on page 403.

---

## 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.** The user class restructure feature lets you 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 16, “Redefining Command Privilege Classes,” on page 433.
- **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 an entry in the z/VM user directory. You can create your own directory or update the sample that z/VM provides. See Chapter 17, “Creating and Updating a User Directory,” on page 447 which contains detailed descriptions of the control statements you can code in the directory.
- **Define virtual machine modes and processor configurations.** Coding the MACHINE statement in a user’s directory entry allows you to specify:
  1. The virtual machine mode (ESA, XA, or XC), which indicates the architecture the virtual machine simulates (ESA/390 or ESA/XC).

**Note:** XA mode is supported for compatibility and is functionally equivalent to ESA mode. Some CMS applications may 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 on the MACHINE directory control statement, see “MACHINE Directory Control Statement (General)” on page 510.

As an alternative to including a MACHINE statement in each user’s directory 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 directory profile. See “Creating Directory Profiles” on page 455.

If the MACHINE statement is not included in the user’s directory 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’s directory entry or in a profile. If no CPU statements are included in the directory 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 directory entry and are not included in a directory profile that contains a MACHINE statement. For more information, see the “GLOBALOPTS Directory Control Statement (Control)” on page 486.

- **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 17, “Creating and Updating a User Directory,” on page 447.
- **Make sure that the user IDs specified in the system configuration file have user directory entries.** The sample system configuration file that is shipped on the z/VM System DDR tapes or CD-ROM defines the following user IDs (on the SYSTEM\_USERIDS statement), and the sample directories contain entries 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 copies of the sample directories IBM provides, see the *z/VM Installation and Service Sample Files* informal document packaged with the z/VM System DDR tapes or CD-ROM. For information on the XAUTOLOG directory control statement, see “XAUTOLOG Directory Control Statement (General)” on page 578.
- **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 System Configuration File Statement” on page 248.

---

## 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.** To define your real storage configuration, code the STORAGE system configuration file statement. For more information, see Chapter 18, “Real Storage Planning and Administration,” on page 589 and “STORAGE Statement” on page 222.
- **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 statement in the virtual machine’s directory entry. The USER statement can also set a maximum storage amount that each particular virtual machine user can define. Code the *mstor* operand to specify the maximum virtual machine storage size that a user can define. (Virtual machine users can enter the DEFINE STORAGE command to redefine their virtual machine storage up to the maximum specified on the USER statement.) For more information, see “USER Directory Control Statement (Control)” on page 572.

**Note:** Allowing large numbers of virtual machines to have large storage sizes (primary address spaces and/or data spaces) may affect real storage availability. This is discussed in the descriptions of the USER and XCONFIG Directory Control Statements. See “USER Directory Control Statement (Control)” on page 572 and “XCONFIG ADDRSPACE Operand” on page 583.

- **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 detailed information, refer to the *z/VM: Saved Segments Planning and Administration*.

---

## Auxiliary Storage (Expanded Storage and DASD) 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, Expanded Storage, and virtual disk 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* and *z/VM: I/O Configuration*. For more information on coding the RDEVICE statement, see “RDEVICE Statement (DASD)” on page 195.
- **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*.

## Overview

- **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 136 and “SYSTEM\_RESIDENCE Statement” on page 230.
- **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 23.
- **Allocate temporary disk space to virtual machines.** Use the MDISK directory control 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 Control Statement (Device)” on page 513.

## 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* and *z/VM: I/O Configuration*. For more information on coding the RDEVICE statement, see “RDEVICE Statement (DASD)” on page 195.
- **Code a DEDICATE statement in the user directory entry 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 622.

**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* and *z/VM: I/O Configuration*. For more information on coding the RDEVICE statement, see “RDEVICE Statement (DASD)” on page 195.
- **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 255 or “SYSUVOL (Optional)” on page 703.

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 control statement to define a minidisk for a virtual machine. Use the LINK directory control 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 Control Statement (Device)" on page 513 and "LINK Directory Control Statement (Device)" on page 500.

**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 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 619.

## 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*.

## Expanded Storage

Expanded Storage is hardware available only on certain processors. It provides another level of general-purpose storage. Expanded Storage may be dedicated to one or more virtual machines and shared with CP. It can also be used for paging and minidisk caching. Do the following for Expanded Storage:

- **Plan a map of Expanded Storage** and determine which virtual machines should have access to this resource (see "Mapping Expanded Storage" on page 593).
- **Use the XSTORE directory statement to attach Expanded Storage to a virtual machine.** (You can also do this by using the ATTACH XSTORE command.) For information on the XSTORE directory statement, see "XSTORE Directory Control Statement (General)" on page 585. For information on the ATTACH XSTORE command, refer to *z/VM: CP Commands and Utilities Reference*.
- **Use the RETAIN XSTORE command** to set aside Expanded Storage for exclusive use by CP (see *z/VM: CP Commands and Utilities Reference*). Note that CP determines which portion of Expanded Storage it will use for paging and minidisk caching. Use the RETAIN XSTORE command to override this.
- **Code the NOMDC operand of the MINIOPT directory statement** if you do *not* want to use a portion of Expanded Storage as a minidisk cache (see "MINIOPT Directory Control Statement (Device)" on page 524 and Chapter 19, "Expanded Storage Planning and Administration," on page 591). Otherwise, you do not need to code the MINIOPT control statement.

## Virtual Disks in Storage

A virtual disk in storage is the temporary simulation of an FBA minidisk in an address space in system 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.

## Overview

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 136.

- Using the MDISK directory control statement. This creates a shareable virtual disk in storage. Use the LINK directory control 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 Control Statement (Device)” on page 513 and “LINK Directory Control Statement (Device)” on page 500.

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 gigabytes (based on the fact that each gigabyte of virtual disk defined requires 2050 pages of real storage below 2 gigabytes)
- 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 136.

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 the *z/VM: Running Guest Operating Systems* book.

## 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* and *z/VM: I/O Configuration*. Otherwise, see either “RDEVICE Statement (Graphic Display Devices)” on page 197 or Appendix C, “Defining I/O Devices Using HCPRIO,” on page 705.
- **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* and *z/VM: I/O*

*Configuration.* Otherwise, see either “EMERGENCY\_MESSAGE\_CONSOLES Statement” on page 124 or “OPERATOR\_CONSOLES Statement” on page 180.

- **To define a virtual machine operator console for a virtual machine, code the CONSOLE control statement in the directory entry for a user.** For more information, see “CONSOLE Directory Control Statement (Device)” on page 463.
- **Define terminals for virtual machines.** To define a terminal for a virtual machine, code the DEDICATE or SPECIAL control statements, or both, in the directory entry for a user. For more information, see “DEDICATE Directory Control Statement (Device)” on page 477 and “SPECIAL Directory Control Statement (Device)” on page 557.

---

## 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* and *z/VM: I/O Configuration*. Otherwise, see:
  - “RDEVICE Statement (Card Punches)” on page 190
  - “RDEVICE Statement (Card Readers)” on page 192
  - “RDEVICE Statement (Impact Printers)” on page 201
  - “RDEVICE Statement (3800 Printers)” on page 212
- **Code a DEDICATE statement in the virtual machine's user directory entry.** 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 Control Statement (Device)” on page 477.

### 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* and *z/VM: I/O Configuration*. Otherwise, see “RDEVICE Statement” on page 188.
- **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 150, “USERFORM Statement” on page 247, or “SYSFORM (Optional)” on page 675.
- **Specify classification titles for specific classes of spooled output on the PRINTER\_TITLE statement.** For more information, see either “PRINTER\_TITLE Statement” on page 182 or “SYSPCLAS (Optional)” on page 691.
- **Define forms control buffers (FCBs) and universal character sets (UCSs) for 3203, 3211, 3262, 4245, and 4248 printers.** For more information, see Chapter 14, “Creating and Modifying Image Libraries for Printers,” on page 411.
- **Define image libraries for 3800 and impact printers.** For more information, see Chapter 14, “Creating and Modifying Image Libraries for Printers,” on page 411.

- **Define spooled unit record devices for virtual machines.** For more information, see “SPOOL Directory Control Statement (Device)” on page 562.

---

## 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's 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 (if the machine is in LPAR mode)
- Query the channel paths
- Query the status of channel paths to devices.

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

---

## 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 control statements) and setting up the OpenExtensions Byte File System. For an overview of how to set up OpenExtensions, see the *z/VM: OpenExtensions User's Guide*.



---

## Part 2. System Planning and Administration

<b>Chapter 2. Configuring Your System</b> . . . . .	23
Specifying System Configuration Information . . . . .	23
Using Configuration Files . . . . .	23
What You Can Specify in the System Configuration File . . . . .	24
Contents of the System Configuration File . . . . .	25
The Bare Minimum . . . . .	25
Other Important Statements . . . . .	25
Real Devices . . . . .	25
User Volume List . . . . .	26
Cross System Operations . . . . .	26
Other System Attributes . . . . .	26
Semantic and Syntactic Statements . . . . .	27
Spool File Processing . . . . .	27
CP File System . . . . .	27
Logo Processing . . . . .	28
What You Can Specify in the Logo Configuration File . . . . .	28
Contents of the Logo Configuration File . . . . .	28
Choosing Logo Picture Files . . . . .	28
Input Area . . . . .	29
Online Messages . . . . .	29
Status Area . . . . .	29
<b>Chapter 3. Understanding the CP File System</b> . . . . .	31
Initial State of the CP File System . . . . .	31
Changing the List of Disks Accessed by CP . . . . .	32
Performance Considerations . . . . .	33
Displaying the Contents of CP-Accessed Minidisks . . . . .	33
Changing Information on CP-Accessed Minidisks . . . . .	34
<b>Chapter 4. Using the Stand-Alone Program Loader</b> . . . . .	37
Creating the Stand-Alone Program Loader . . . . .	37
Overriding the Console that CP Will Use . . . . .	39
Overriding Stand-Alone Program Loader Defaults . . . . .	39
To IPL from a SCSI Disk . . . . .	39
The Stand-Alone Program Loader Screen . . . . .	40
Passing IPL Parameters . . . . .	42
<b>Chapter 5. Defining I/O Devices</b> . . . . .	45
Device Support . . . . .	45
Unsupported Devices . . . . .	46
Device Sensing . . . . .	46
DASD . . . . .	46
Tapes . . . . .	47
Printers . . . . .	47
Unit Record Devices . . . . .	48
Displays . . . . .	49
Communication Controllers . . . . .	49
ESCON Devices . . . . .	49
Other Devices . . . . .	50
<b>Chapter 6. The System Configuration File</b> . . . . .	51
The Parm Disk . . . . .	51
Summary of System Configuration File Statements . . . . .	51

General Rules for Coding a System Configuration File . . . . .	55
Format . . . . .	55
Comments . . . . .	55
Continuations . . . . .	55
Case . . . . .	56
Record Qualifiers . . . . .	56
The Order of Statements in the File . . . . .	57
Checking the Syntax of Statements in the File . . . . .	57
ALTERNATE_OPERATORS Statement . . . . .	58
ASSOCIATE EXIT Statement . . . . .	60
ASSOCIATE MESSAGES / MSGS Statement . . . . .	64
CHARACTER_DEFAULTS Statement . . . . .	67
CP_ACCESS Statement . . . . .	70
CP_ADDON_INITIALIZE_ROUTINES Statement . . . . .	72
CP_OWNED Statement . . . . .	73
CPXLOAD Statement . . . . .	76
DEFINE ALIAS Statement . . . . .	81
DEFINE COMMAND / CMD Statement . . . . .	84
DEFINE DIAGNOSE Statement . . . . .	90
DEFINE EXIT Statement . . . . .	94
DEFINE LAN Statement . . . . .	97
DEFINE VSWITCH Statement . . . . .	100
DEVICES Statement . . . . .	105
DISABLE COMMAND / CMD Statement . . . . .	110
DISABLE DIAGNOSE Statement . . . . .	112
DISABLE EXITS Statement . . . . .	114
DISTRIBUTE Statement . . . . .	116
DRAIN (Disk) Statement . . . . .	118
EDEVICE Statement . . . . .	121
EMERGENCY_MESSAGE_CONSOLES Statement . . . . .	124
ENABLE COMMAND / CMD Statement . . . . .	126
ENABLE DIAGNOSE Statement . . . . .	128
ENABLE EXITS Statement . . . . .	130
ENFORCE_BY_VOLID Statement . . . . .	132
EQUATE Statement . . . . .	133
EXTERNAL_SYNTAX Statement . . . . .	135
FEATURES Statement . . . . .	136
FORM_DEFAULT Statement . . . . .	150
HOT_IO_RATE Statement . . . . .	152
IMBED Statement . . . . .	155
INIT_MITIME Statement . . . . .	157
IODF Statement . . . . .	158
JOURNALING Statement . . . . .	160
LOGO_CONFIG Statement . . . . .	164
MODIFY COMMAND / CMD Statement . . . . .	165
MODIFY DIAGNOSE Statement . . . . .	169
MODIFY EXIT Statement . . . . .	172
MODIFY LAN Statement . . . . .	175
MODIFY PRIV_CLASSES Statement . . . . .	176
MODIFY VSWITCH Statement . . . . .	178
OPERATOR_CONSOLES Statement . . . . .	180
PRINTER_TITLE Statement . . . . .	182
PRIV_CLASSES Statement . . . . .	184
PRODUCT Statement . . . . .	186
RDEVICE Statement . . . . .	188
RDEVICE Statement (Advanced Function Printers) . . . . .	189



RDEVICE Statement (Card Punches)	190
RDEVICE Statement (Card Readers)	192
RDEVICE Statement (Communication Controllers)	193
RDEVICE Statement (DASD)	195
RDEVICE Statement (Graphic Display Devices)	197
RDEVICE Statement (Impact Printers)	201
RDEVICE Statement (Special Devices)	205
RDEVICE Statement (Tape Units)	207
RDEVICE Statement (Unsupported Devices)	209
RDEVICE Statement (3800 Printers)	212
SAY Statement	217
SET Statement	219
START (Disk) Statement	220
STORAGE Statement	222
SYSTEM_DATEFORMAT Statement	225
SYSTEM_IDENTIFIER Statement	226
SYSTEM_IDENTIFIER_DEFAULT Statement	228
SYSTEM_RESIDENCE Statement	230
SYSTEM_USERIDS Statement	232
THROTTLE Statement	236
TIMEZONE_BOUNDARY Statement	237
TIMEZONE_DEFINITION Statement	239
TOLERATE_CONFIG_ERRORS Statement	241
TRANSLATE_TABLE Statement	243
USERFORM Statement	247
USER_DEFAULTS System Configuration File Statement	248
USER_VOLUME_EXCLUDE Statement	251
USER_VOLUME_INCLUDE Statement	253
USER_VOLUME_LIST Statement	255
VMLAN Statement	257
XLINK_DEVICE_DEFAULTS Statement	259
XLINK_SYSTEM_EXCLUDE Statement	263
XLINK_SYSTEM_INCLUDE Statement	264
XLINK_VOLUME_EXCLUDE Statement	266
XLINK_VOLUME_INCLUDE Statement	268
XSPPOOL_SYSTEM Statement	271
XSPPOOL_TRACE Statement	273
XSPPOOL_XLIST_INPUT Statement	274
XSPPOOL_XLIST_OUTPUT Statement	276
<b>Chapter 7. The Logo Configuration File</b>	<b>279</b>
Using a Logo Configuration File	279
Summary of Logo Configuration File Statements	279
General Rules for Coding a Logo Configuration File	279
Format	279
Comments	279
Continuations	280
Case	280
Categories	281
Precedence	281
Selection	282
Creating Logo Screens	282
Special Considerations When Creating Logos	283
CHOOSE_LOGO Statement	285
INPUT_AREA Statement	291
ONLINE_MESSAGE Statement	292

STATUS Statement . . . . .	293
<b>Chapter 8. Setting Up Service Virtual Machines . . . . .</b>	<b>295</b>
Setting Up Virtual Machines for Accounting . . . . .	295
Setting Up Virtual Machines to Collect Accounting Records . . . . .	297
Specifying a New Accounting Virtual Machine . . . . .	297
Starting Manual Retrieval of Accounting Records . . . . .	297
Disassociating a User ID from the Retrieval of Accounting Records . . . . .	298
Accounting Record Formats. . . . .	299
Accounting Records for Virtual Machine Resource Usage (Record Type 1) . . . . .	299
Accounting Records for Dedicated Devices (Record Type 2). . . . .	300
Accounting Records for Temporary Disk Space (Record Type 3) . . . . .	301
Accounting Records for Journaling (Record Types 04, 05, 06, 08, and 0I) . . . . .	301
Accounting Records for SNA/CCS (Record Type 07) . . . . .	304
Accounting Records for Inter-System Facility for Communications (Record Type 09) . . . . .	304
Accounting Records for logging changes to a user's privilege (Record Type 0A) . . . . .	306
Accounting Records for virtual disk in storage space (Record Type B) . . . . .	307
Accounting Records Network Data Transmissions (Record Type C) . . . . .	307
Accounting Records for CPU Capability (Record Type D) . . . . .	309
Adding Your Own Accounting Records and Source Code . . . . .	310
User-Initiated Accounting Records (Record Type C0) . . . . .	310
CP Accounting Exit (Module HCPACU) . . . . .	310
Setting Up Virtual Machines for Error Recording . . . . .	314
Setting Up Virtual Machines to Collect EREP Records . . . . .	315
Specifying a New EREP Virtual Machine . . . . .	316
Starting EREP Record Retrieval Manually . . . . .	316
Disassociating a User ID from the Retrieval of EREP Records . . . . .	317
Setting Up Virtual Machines for Symptom Record Recording . . . . .	318
Setting Up Virtual Machines to Collect Symptom Records. . . . .	319
Specifying a New Symptom Record Recording Virtual Machine. . . . .	319
Starting Manual Retrieval of Symptom Records . . . . .	320
Disassociating a User ID from the Retrieval of Symptom Records. . . . .	320
Setting Up a Virtual Machine for Communication Controller Support for Emulator Program (EP) . . . . .	321
PVM Example. . . . .	323
Setting Up Service Pool Virtual Machines. . . . .	325
Setting Up Print Services Facility/VM Virtual Machines . . . . .	326
Setting Up Virtual Machines for Data Storage Management . . . . .	327
<b>Chapter 9. Planning for SNA Console Communication Services (SNA/CCS) . . . . .</b>	<b>329</b>
Structure of the SNA Environment . . . . .	329
Establishing the SNA/CCS Terminal Environment. . . . .	330
Defining the VSM to z/VM . . . . .	330
Defining Logos Used by VSMs . . . . .	330
Defining SNA/CCS to VCNA . . . . .	330
Defining SNA/CCS to VSCS . . . . .	331
Enabling SNA Communication. . . . .	331
Starting the VTAM Service Machine. . . . .	332
Improving SNA/CCS Performance . . . . .	332
VSM Termination. . . . .	332
<b>Chapter 10. Setting Up Cross System Extensions (CSE) . . . . .</b>	<b>333</b>

Overview . . . . .	333
CSE Capabilities . . . . .	333
Users Supported . . . . .	334
Setting Up a CSE Complex . . . . .	334
Planning for CSE . . . . .	334
Requirements . . . . .	334
Restrictions . . . . .	335
Supported Features . . . . .	335
Sharing DASD Volumes . . . . .	336
Avoiding Shared DASD I/O Bottlenecks . . . . .	337
Placing of the CSE Area . . . . .	340
Defining Volumes Containing Minidisks to be Shared . . . . .	340
Cross System Linking and Synchronization of Directories . . . . .	340
Spool File Directory Statements . . . . .	341
Preparing for CSE . . . . .	341
Cabling the Hardware . . . . .	341
Preparing for a Single Source Directory . . . . .	341
Nonshared Directory Entries . . . . .	342
Enabling CSE . . . . .	345
Installing the VM/Pass-Through Facility and Its Modifications . . . . .	345
Defining CP for Each System . . . . .	346
System Definition for CSE . . . . .	346
CP_OWNED Statements . . . . .	347
Example of a System Configuration File . . . . .	347
Initializing the Volumes for Cross System Link . . . . .	349
Generating the Cross System Link Facility . . . . .	349
Initializing the CSE Area on Shared Minidisk Volumes . . . . .	350
Preparing the Communication Virtual Machine (CVM) . . . . .	350
Verifying Your CSE Configuration . . . . .	353
Verifying Cross System Link . . . . .	353
Enabling Cross System Spool . . . . .	353
Starting CSE Communication . . . . .	353
Verifying Cross System Spool . . . . .	354
Activating Spool for Systems Not Active after Start CSECOM . . . . .	355
Phasing CSE into the Production Environment . . . . .	356
Networking Implications of CSE . . . . .	357
Administering CSE . . . . .	358
How Linking Works in z/VM . . . . .	358
How CSE Extends Linking . . . . .	359
Controlling Cross System Link . . . . .	359
CSE Area . . . . .	359
Location of the CSE Area . . . . .	361
Protecting the CSE Area . . . . .	361
Placing of the CSE Area . . . . .	361
VM I/O Performance Measurement for Shared DASD . . . . .	365
What Is Cross System Spool? . . . . .	366
How CSE Extends Spooling . . . . .	367
z/VM Spooling Commands Extended by CSE . . . . .	368
Copy Input (Reader) Spool Files . . . . .	369
Copy Output (Print and Punch) Spool Files . . . . .	370
Spool File Numbering . . . . .	370
Access to Spool Files . . . . .	370
VM/Pass-Through Facility Functions for Cross System Spool . . . . .	371
Spool Synchronization Request Tracing . . . . .	371
Multiple Concurrent Logon Sessions . . . . .	371
Preparing to Deal with Outages . . . . .	372

Returning to Non-CSE Operation . . . . .	372
<b>Chapter 11. Customizing the CP Message Function . . . . .</b>	<b>373</b>
HCPMSU Module . . . . .	373
Entry Point HCPMSUEX . . . . .	374
Parameter Values . . . . .	374
Changing and Adding to the CP Message Function . . . . .	377
<b>Chapter 12. Security and Integrity in z/VM . . . . .</b>	<b>379</b>
Security-Enhancing Products . . . . .	379
Security Considerations and Guidelines . . . . .	380
z/VM Security Facilities . . . . .	383
Using an External Security Manager for Auditing and Protecting . . . . .	384
Verifying Storage Access . . . . .	384
Clearing Temporary Disk Space . . . . .	384
Permitting Bypassing of Directory Password Authorization . . . . .	385
Journaling the LOGON, AUTOLOG, XAUTOLOG, and LINK Commands . . . . .	385
Automatic Deactivation of Restricted Passwords . . . . .	386
Using Link Access Control Options . . . . .	387
Suppressing Passwords Entered on the Command Line . . . . .	387
Using an Integrated Cryptographic Facility . . . . .	387
Maintaining System Integrity . . . . .	389
z/VM System Integrity Requirements . . . . .	390
z/VM Integrity and CP Function . . . . .	392
Data Structures That Can Enhance System Integrity . . . . .	395
z/VM Options That Can Enhance System Integrity . . . . .	397
Storage Handling . . . . .	397
Program Stack, Security, and Integrity . . . . .	398
Reporting z/VM Integrity Problems . . . . .	400
<b>Chapter 13. Using the Stand-Alone Dump Utility . . . . .</b>	<b>403</b>
Overview . . . . .	403
Creating the Stand-Alone Dump Utility . . . . .	403
Before You Begin . . . . .	404
Devices You Can Use to IPL a Stand-Alone Dump . . . . .	405
Devices to Which You Can Send Dump Output . . . . .	406
Example for Generating the Stand-Alone Dump Utility . . . . .	406
Taking a Stand-Alone Dump . . . . .	408
Processing the Stand-Alone Dump Data on Tape . . . . .	409
The HCPSADMP EXEC . . . . .	409
Usage Notes . . . . .	409
<b>Chapter 14. Creating and Modifying Image Libraries for Printers. . . . .</b>	<b>411</b>
Creating Text Decks . . . . .	411
Creating Text Decks for the 3800 . . . . .	411
Creating Text Decks for Impact Printers . . . . .	412
Universal Character Sets and FCBs Supplied by IBM . . . . .	412
Adding New Universal Character Set Buffer Images . . . . .	414
Forms Control Buffers Supplied by IBM . . . . .	418
Adding a New Forms Control Buffer . . . . .	418
Adding an Extended Forms Control Buffer for the 4248 Printer . . . . .	420
Using VMFHASM or VMFHLASM to Create a Text Deck . . . . .	422
Image Libraries . . . . .	422
Default Image Library Names . . . . .	422
Installing the Image Library That IBM Provides . . . . .	423
Installing Your Own Image Library . . . . .	423

Modifying Image Libraries . . . . .	424
Adding Files to an Image Library . . . . .	424
Deleting Members from an Image Library . . . . .	424
Replacing Members or Modules in an Image Library . . . . .	424
Creating an Image Library Map . . . . .	424
Displaying Information about Image Libraries . . . . .	424
Purging Image Libraries . . . . .	425
Keeping Backup Copies of Image Libraries . . . . .	425
Where to Find More Information about Printers . . . . .	425
<b>Chapter 15. CCW Translation . . . . .</b>	<b>427</b>
Using CCW Translation . . . . .	427
Coding the Device Class Macroinstruction . . . . .	427
Unsupported Devices Tables . . . . .	428
Device Macroinstruction . . . . .	429



---

## 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 class restructure (UCR) files and 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 5, “Defining I/O Devices,” on page 45 discusses considerations for defining I/O devices.
- Chapter 6, “The System Configuration File,” on page 51 describes the system configuration file in more detail.
- Chapter 7, “The Logo Configuration File,” on page 279 describes the logo configuration file in more detail.
- Chapter 16, “Redefining Command Privilege Classes,” on page 433 describes how to define the user class restructure (UCR) files.
- Chapter 17, “Creating and Updating a User Directory,” on page 447 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. See “Passing IPL Parameters” on page 42 for specifying a name for the system configuration file. You can specify a name for the logo configuration file inside the system configuration file. Place any logo picture files on this disk also.

## Configuring Your System

**Note:** Although z/VM continues to support the system control file (HCPSYS ASSEMBLE), system real I/O configuration file (HCPRIO ASSEMBLE), and system logo definition file (HCPBOX ASSEMBLE), IBM strongly recommends that you use configuration files to define your system. Using the ASSEMBLE files is more difficult and error-prone, requires knowledge of the Assembler H Version 2 Licensed Program, 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.

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. Use the QUERY CPDISK command to find out what parm disks are accessed, the CPRELEASE command to release the parm disk, the LINK and ACCESS commands for write access, and the CPACCESS command to return the disk to CP. See Chapter 3, “Understanding the CP File System,” on page 31 for additional information on these commands.

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.” 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 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.

**Note:** You cannot define the size of system storage in the system configuration file. For information on defining the size of system storage to a value less than what is actually available, see the STORE parameter under "Passing IPL Parameters" on page 42.

## 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 51.

### 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 identifiers 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.

### 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 5, "Defining I/O Devices," on page 45 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

## Configuring Your System

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 Operations

You can control cross system spool operations, cross system link operations, and cross system commands using the following statements:

- `XLINK_SYSTEM_EXCLUDE` and `XLINK_SYSTEM_INCLUDE` specify the systems to be excluded from or included in cross system link protection.
- `XLINK_VOLUME_EXCLUDE` and `XLINK_VOLUME_INCLUDE` specify the DASD volumes to be excluded from or included in cross system link protection.
- `XSPOOL_SYSTEM` specifies the systems that are to participate in cross system commands and spooling operations.
- `XSPOOL_TRACE` allocates storage for the cross system spool (`XSPOOL`) trace tables.
- `XSPOOL_XLIST_INPUT` and `XSPOOL_XLIST_OUTPUT` specify virtual machines whose input or output spool files do not participate in cross system spooling and commands. After the system is running, you can use the `XSPOOL XLISTADD` and `XSPOOL XLISTDEL` commands to work with these lists.

The statements controlling cross system link operations are separate from those controlling cross system commands and spool operations.

### 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.
- **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.
- **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:

- **EQUATE** creates names for groups of systems that all have something in common and should, therefore, be treated similarly. You can use system identifiers 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 279.

### 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      (NOAUT0log)) 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:

## Understanding the CP File System

```
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:

```
cpclistfile * logo a
```

## Understanding the CP File System

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
Ø8E6	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:

```
cp!istfile 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;
```

## Understanding the CP File System

```
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. Using the Stand-Alone Program Loader

The Stand-Alone Program Loader (SAPL) is a generic loader that loads stand-alone programs stored in CMS module form on a CMS minidisk. Using SAPL, you can use any stand-alone program that can be generated in the form of a CMS module.

Examples of such programs are:

- The CP nucleus module (CPLOAD)
- Stand-Alone Program Loader Creation Utility (SALIPL)
- DASD Dump Restore Utility (DDR)
- Device Support Facilities Program (ICKDSF)

This chapter tells you how to use the Stand-Alone Program Loader to load a stand-alone program that is generated in the form of a module.

During the IPL process, various information is read from DASD devices. Here are some terms that will be used to refer to these devices:

- The **IPL device** is the device which is specified on the load screen and from which SAPL is loaded.
- The **SYSRES device** is the device from which SAPL loads the CP module.
- The **parm disk device** is the device on which the CP module finds the system configuration file.
- Other DASD devices are used but are not of interest for this discussion. They include the DASD devices used for the CP directory, warmstart, checkpoint, paging, spooling and user minidisks.

---

### Creating the Stand-Alone Program Loader

Before a DASD device can be IPLed, SAPL must be written to the device using the Stand-Alone Loader Creation Utility (SALIPL). See the *z/VM: CP Commands and Utilities Reference* for information on how you can write SAPL to the IPL device by using SALIPL from the CMS command line. See the “Running Utility Programs” in *z/VM: System Operation* for information on how to write SAPL to the IPL device by using SALIPL on the real processor.

### Important Note!

As of z/VM V5R1, the number of blocks on FBA DASD (SCSI or not) that SALIPL uses for SAPL has increased. SALIPL writes to blocks 5-207 on CP-formatted DASD. In previous releases, SALIPL wrote to blocks 5-31. You must ensure that no other functions, such as CP directory, warmstart, checkpoint, paging, spooling, or minidisks, are using the area to which SALIPL writes.

On a CMS minidisk with a RECOMP area, SALIPL writes SAPL to blocks 5-207 of the RECOMP area. In previous releases, blocks 5-31 of the RECOMP area were used. In order to accommodate the larger size of SAPL, you may need to increase the size of the RECOMP area on the CMS minidisk.

The following data is in the blocks that SALIPL writes out:

Blocks 5–196	SAPL program
Blocks 197–199	Reserved for IBM Use
Blocks 200–207	SCSI boot records

When SALIPL writes SAPL to the IPL device, defaults for the following attributes that SAPL uses are established:

- The location of the CMS minidisk where modules can be found
- The name of the module that is to be loaded by SAPL
- The location in storage at which the module will be loaded, if it is relocatable
- The IPL parameters that are passed to the program being loaded
- Operator instructions that are displayed on the SAPL screen.

These defaults can be overridden as described below.

The default module is CP (default name CPLOAD), but it may be a stand-alone program in CMS module form. You can have many CP modules in your installation, allowing for different conditions or providing backup. You select the CP module to be used:

- When you run Stand-Alone Program Loader Creation Utility (SALIPL)
- During IPL by overriding the SAPL defaults on the SAPL screen
- During SHUTDOWN REIPL by using the MODULE operand.

Although the file name of a CP module is variable, the file type must be MODULE. CP modules typically reside on a minidisk that is identified as a parm disk in the volume allocation table, but you can select the DASD volume and offset/extent from which SAPL will read. Therefore, you can have CP modules on multiple minidisks on multiple DASDs.

**Note:** Although SAPL will accept a load origin when loading a CP module, it has no effect. CP will always relocate itself to location X'2000'.

## Overriding the Console that CP Will Use

If you only need to override the console that CP will use as the system operator console during CP initialization, specify a load parameter of CONSxxxx. For example:

- Specify CONSSYSG to use the integrated 3270 console on the hardware management console
- Specify CONSSYSC to use the operating system messages panel on the hardware management console
- Specify CONSxxxx to cause CP to use the 3270 at address xxxx.

## Overriding Stand-Alone Program Loader Defaults

If the defaults that were established by the SALIPL utility when it wrote SAPL to the IPL device are acceptable, all you need to do is to IPL from the IPL device. If you need to change any of the defaults, however, then you must supply a load parameter with the address of a console for SAPL to use. The console specified must accept 3270 CCWs and data streams and have at least 20 lines and 80 columns.

If you are running SAPL on a real processor, you specify the console address by entering the device number of the console into the load parameter field of the load window on your processor console. The SAPL screen will be displayed on the specified console.

If you are running SAPL in a virtual machine, you specify the console address using the LOADPARM option of the CP IPL command. For example:

```
IPL 18B LOADPARM EA0
```

In this example, 18B is the DASD volume to load from and EA0 is the device number of the console to be used.

To cause the SAPL screen to appear on the integrated 3270 console on the hardware management console, specify SYSG as a load parameter. The operating system messages panel on the hardware management console (SYSC) cannot be used by SAPL because it does not accept 3270 CCWs and data streams.

## To IPL from a SCSI Disk

If you are IPLing CP in a virtual machine from a minidisk on an emulated FBA device (SCSI), just specify the minidisk device address as the IPL device on the IPL command as shown in the previous example.

To IPL from a SCSI disk on the real processor, access the load window on the processor console in order to specify additional information. To IPL from a SCSI disk in a virtual machine by using an FCP device, use the CP SET LOADDEV command to specify additional information that will be used by the CP IPL command:

- Specify the load address of the FCP device you wish to use.
- Specify the world wide port name (PORT on the SET LOADDEV command)
- Specify the logical unit number (LUN)
- Specify 0 for the boot program selector (BOOTPROG)
- Specify C8 for the boot record logical block address (BR\_LBA) if the IPL device has been CP-formatted by ICKDSF or CPFMTXA. The SALIPL utility writes the

## Using SAPL

SCSI boot record to block 200 (X'C8') of a CP-formatted device. For a CMS-formatted device, SALIPL writes the SCSI boot record at block 200 (X'C8') into the RECOMP area of the minidisk.

- For the load parameter operand, optionally specify a console address on which you wish the SAPL screen to appear.
- For the OS specific load parameters (SCPDATA) operand, optionally specify a console address on which you wish the SAPL screen to appear.

**Note:** As described earlier, the console specified must accept 3270 data streams. Specify SYSG for the integrated 3270 console on the HMC. The console may also be specified in the load parameter field of the load panel on the system console. Note that if both the OS specific load parameter SCPDATA and the load parameter fields are specified, SAPL will use the load parameter value.

If your configuration uses normal DASD devices (such as 3390), then the IPL device, SYSRES device, and parm disk device can all be the same device or different devices. **If the IPL device is an FCP device, then the IPL device, SYSRES device, and parm disk device must be the same SCSI disk.** This is because the FCP device, WWPN, and LUN combination specified on the load screen is the only DASD device path known during the IPL process.

## The Stand-Alone Program Loader Screen

When the IPL completes and a valid console has been specified for the load parameter, you will see a menu screen similar to Figure 1. The defaults will be filled in for some of the fields — all fields except the comments section can be altered.

```
STAND ALONE PROGRAM LOADER: z/VM VERSION n RELEASE n.n
DEVICE NUMBER: 018B      MINIDISK OFFSET: 35      EXTENT: -
MODULE NAME:  CLOAD      LOAD ORIGIN: 2000

-----IPL PARAMETERS-----
cons=0ea0 fn=maine

-----COMMENTS-----
Secondary parm disks can be found on 18B at offset 185 and on 18A at offsets
35 and 185.

-----

9= FILELIST 10= LOAD 11= TOGGLE EXTENT/OFFSET
```

Figure 1. Stand-Alone Program Loader

The Stand-Alone Program Loader screen contains the following fields:

### DEVICE NUMBER

Specifies the device number of the DASD volume from which SAPL will read. Initially, this will be the same as the device you IPLed, but it can be changed to any DASD volume containing CMS minidisks.

If you IPLed from an FCP device (SCSI disk), the DEVICE NUMBER field will contain the FCP device address. The WWPN and LUN are not displayed on



the SAPL screen and cannot be changed. If you IPLed from an FCP device, you can change the DEVICE NUMBER value to a DASD volume (3390). If you IPLed from a DASD volume, however, you cannot change to an FCP device.

<b>MINIDISK OFFSET</b>	Specifies the offset from the beginning of the DASD volume to the start of the CMS minidisk from which SAPL is to read. This number is cylinders for count key data devices and blocks for fixed block architecture devices.
<b>EXTENT</b>	Specifies the extent number from 1 to 9 (first to ninth). This field may be used only with DASD volumes that have been formatted using the Device Support Facilities Program (ICKDSF) and have an allocation map containing PARM extents written on them. When a valid extent is entered, the offset corresponding to the extent will be displayed in the MINIDISK OFFSET field.  <b>Note:</b> Only one of the fields, EXTENT or MINIDISK OFFSET, can be altered at a time. Use PF11 to toggle which field can be altered.
<b>MODULE NAME</b>	Specifies the name of a module on the minidisk that will be loaded by SAPL.
<b>LOAD ORIGIN</b>	Specifies the location in memory (HEX address) into which SAPL will load a relocatable module. Specify a LOAD ORIGIN value of 2000 if IPLing a CP module. As of z/VM V5R1, CP will always run with an origin of X'2000'. If a value other than 2000 is specified, then after initially loading at the specified address, CP will relocate itself to location X'2000'.
<b>IPL PARAMETERS</b>	Specifies the data that will be passed to the loaded program by SAPL. The contents of this field depends entirely on the program loaded by SAPL. For parameters valid for CP, see "Passing IPL Parameters" on page 42. <b>If you are IPLing CP from an FCP device (SCSI disk), PDVOL=addr must be specified as an IPL parameter.</b> See the PDVOL= description below for more information.
<b>COMMENTS</b>	Contains comments such as operator instructions. This field may only be altered by running the SALIPL utility.

There is also a message line after the comments area. Any informational messages or error messages will appear on the message line.

There is a list of program function (PF) key numbers and their functions at the bottom of the screen. PF11 toggles which of the two fields, MINIDISK OFFSET or EXTENT, is active. PF10 causes SAPL to try to load a module using the current contents of the fields. PF9 displays a list (FILELIST) of files that are on the minidisk. The module you wish to use can be selected on that screen. Figure 2 on page 42 is an example of a Stand-Alone Program Loader file list screen.

```

STAND ALONE PROGRAM LOADER: z/VM VERSION n RELEASE n.n
FILENAME FILETYPE FORMAT LRECL   RECORDS   BLOCKS   DATE       TIME
DYN      MODULE   V      65535     59       925 2003/07/07 12:15:28
SALIPL   MODULE   V      24400     4        7 2003/06/22 15:30:10
SYSTEM   CONFIG   F        80     190      4 2003/06/17 14:47:19
HCPSAL   MODULE   V      24400     4        7 2003/06/15 14:32:29
END      MODULE   F      1024    1896    474 2003/06/01 13:40:19
DYNF     MODULE   V      65535     56      869 2003/05/22 12:36:18
DDR      MODULE   V      65535     5        30 2003/05/21 16:21:44
SALTEST  MODULE   F      7656     1        2 2003/05/21 11:11:22
LOGO     CONFIG   F        80     48       1 2003/05/18 16:31:48
LOGO     CONFOLD  F        80     48       1 2003/11/22 14:43:28
SYSTEM   CONFN    F        80    277      6 2003/08/08 15:40:51
GOTHIC   LOGO     F        79     16       1 2003/06/18 17:11:12
UMS      LOGO     V        75     16       1 2003/06/18 17:04:53
SPOOL    LOGO     F        46     16       1 2003/06/17 18:13:58
SYSTEM   NETID    F        80      8       1 2003/04/29 11:40:35
ONLINE   MESSFILE F        80      1       1 2003/04/29 11:26:11
SYSTEM   LOGMSG   F        80      1       1 2003/10/03  9:04:22
3=QUIT  4=SORT(TYPE) 5=SORT(
DATE) 6=SORT(NAME) 7=BACK 8=FORWARD 11=SELECT

```

Figure 2. Stand-Alone Program Loader File List

The files displayed will initially be sorted in descending order by date and time. You can sort by file name/file type, file type/file name, and date/time by using the PF keys identified at the bottom of the screen. If you press PF11 (SELECT) when the cursor is on a line that lists a file, you will be returned to the main menu and the file name of the selected file will be placed in the MODULE NAME file.

## Passing IPL Parameters

Using SAPL, you can pass up to 240 characters of parameters into CP at IPL time. The default contents of the 240-character parameter field can be selected when SALIPL is used to write SAPL to the IPL device. You can choose to override the defaults by using the full screen option of SAPL.

During CP initialization, CP scans the IPL parameter list from left to right. If a parameter is specified more than once, CP uses the last valid occurrence of the parameter.

The following list describes the parameters that CP supports:

**CONS=addr**

**CON=addr**

Specifies the location of the operator console that CP is to use. Specify a one-digit to four-digit device address for a 3270 console. Specify SYSC for the operating system messages panel on the hardware management console (HMC). Specify SYSG for the integrated 3270 console on the HMC.

All OPERATOR\_CONSOles statements in the system configuration file are ignored. The console address specified will be added to the list of consoles defined on the EMERGENCY\_MESSAGE\_CONSOles statement found in the system configuration file if it has not already been included in that list. For more information, see the description of the EMERGENCY\_MESSAGE\_CONSOles statement.

**FN=filename**

Specifies the file name of the system configuration file. If you do not specify this option, CP uses SYSTEM as the file name.

**FT=***filetype*

Specifies the file type of the system configuration file. If you do not specify this option, CP uses CONFIG as the file type.

**NOEXITS**

Intended to assist you in a situation where your dynamic additions to CP are somehow wrong and are preventing CP from completing system initialization. Use the NOEXITS parameter to tell CP not to process any of these requests from your system configuration file:

- CPXLOAD statement
- ENABLE EXIT statement
- EXTERNAL\_SYNTAX statement
- ASSOCIATE MESSAGES statement
- The ENABLE operand of the ASSOCIATE EXIT statement

Except for examining the statement for syntactic correctness, CP will ignore the statement or the operand. If it was one of your dynamic additions to CP (for example, a CP exit) that was causing CP not to initialize, using NOEXITS may help you to IPL CP so that you can correct the problem.

**NOHCD**

Specifies that CP should ignore any IODF statement in the system configuration file. The system initializes as if no IODF statement were present and HCD is not used for I/O configuration.

An example of when the NOHCD IPL parameter would be useful is if you encounter a disabled wait state because of problems with the IODF statement or problems with the file specified on the IODF statement. Use the NOHCD parameter to IPL the system without HCD and then change the IODF statement, replacing the IODF or otherwise investigate the wait state condition.

**PROMPT**

Specifies that you wish to have CP prompt you about start-up options at IPL time. If you specify this option, CP ignores the AUTO\_WARM\_IPL and AUTO\_IPL operands on the FEATURES statement in the system configuration file.

The PROMPT parameter is only valid for the SAPL screen. This parameter is not acknowledged during CP bounce processing (SHUTDOWN REIPL or restarting after an abend).

**PDNUM=***n*

Specifies the number of the parm disk that you wish CP to use. CP will read the allocation map on the IPL device or the DASD volume indicated on the PDVOL option and will skip to the *n* parm disk. If the allocation map does not contain at least *n* parm disks, CP loads a disabled wait.

**PDOFF=***offset*

Specifies the offset at which the parm disk resides in the IPL device or the DASD volume indicated on the PDVOL option. If the DASD volume does not have at least *offset* cylinders or blocks, or if there is no CMS-formatted minidisk at the specified offset, CP loads a disabled wait.

**PDVOL=***addr*

Specifies the real address of the volume on which a parm disk resides that you wish CP to access. If there is no device at address *addr* or if the device is not a DASD volume, CP loads a disabled wait.

If you are IPLing CP from an FCP device (SCSI disk), PDVOL=*addr* must be specified as an IPL parameter. Any device address that is not in the I/O

## Using SAPL

configuration of the system to be IPLed can be specified for *addr*. CP requires this parameter in order build the EDEVICE control block for the SYSRES/parm disk device during CP initialization. See “EDEVICE Statement” on page 121 for more information.

### **STORE=nnnu**

Indicates the amount of storage you wish CP to use. This amount can be less than the real storage that is installed. If you specify an amount that exceeds the actual real storage available on the CPU, CP uses the real storage amount.

*nnnn* is a 1-digit to 4-digit decimal number. *u* can be:

- M Megabytes
- G Gigabytes
- T Terabytes
- P Petabytes
- E Ekabytes

---

## Chapter 5. 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 macro statement 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 the *z/OS and z/VM: Hardware Configuration Manager User's Guide* 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 the *z/VM: General Information* book 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 the *z/VM: General Information* book. 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 either RDEVICE TYPE UNSUPPORTED statement in the system configuration file or the CLASS operand on the RDEVICE macroinstruction. 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 188.

---

## DASD

Table 1 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 1. Configuration Guide for DASD

Device	Statement Needed in Configuration File
3380 Models A04, AA4, B04, AD4, BD4, AE4, BE4, AJ4, BJ4, AK4, BK4	<ul style="list-style-type: none"><li>• When not shared, not required</li><li>• When shared, TYPE DASD YES</li></ul>
3380 Model CJ2	<ul style="list-style-type: none"><li>• When not shared, not required</li><li>• When shared, TYPE DASD YES</li></ul>
3390 Models A14, A18, B14, B18, B1C, A24, A28, B24, B28, B2C, A34, B34, A38, B38, B3C	<ul style="list-style-type: none"><li>• When not shared, not required</li><li>• When shared, TYPE DASD YES</li></ul>

Table 1. Configuration Guide for DASD (continued)

Device	Statement Needed in Configuration File
3390 Models A98, B9C	<ul style="list-style-type: none"> <li>• When not shared, not required</li> <li>• When shared, TYPE DASD YES</li> </ul>
9336 Model 20	Not required

See “RDEVICE Statement (DASD)” on page 195 for a complete description of the RDEVICE statement.

---

## Tapes

Table 2 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 2. 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*
<b>Notes:</b>	
*The 'TYPE TAPE' operand can be specified to provide query capability for tapes that are intentionally left offline at IPL.	
**See the <i>DFSMS/VM Installation and Customization Guide</i> for information on defining the 3495 Tape Library Dataserver for VM use.	

See “RDEVICE Statement (Tape Units)” on page 207 for a complete description of the RDEVICE statement.

---

## Printers

Table 3 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 3. 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

## Defining I/O Devices

Table 3. Configuration Guide for Printers (continued)

Device	Statement Needed in Configuration File
3800 Models 3, 6, 8	TYPE Advance Function Printing™ (AFP)™ or TYPE 3800
3812	<ul style="list-style-type: none"> <li>• When coax attached, not required</li> <li>• When BSC attached, TYPE BSC_ADAPTER</li> <li>• When SDLC (VTAM attached), not required</li> </ul>
3816	<ul style="list-style-type: none"> <li>• When coax attached, not required</li> <li>• When BSC attached, TYPE BSC_ADAPTER</li> <li>• When SDLC (VTAM attached), not required</li> </ul>
3820	<ul style="list-style-type: none"> <li>• When channel attached for use as a PSF printer, TYPE AFP</li> <li>• When VTAM attached, TYPE 3705</li> </ul>
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 193 for a complete description of the appropriate RDEVICE statement. For the other printers, see “RDEVICE Statement (Advanced Function Printers)” on page 189 and “RDEVICE Statement (Impact Printers)” on page 201 for a complete description.

---

## Unit Record Devices

Table 4 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 4. 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 192 and “RDEVICE Statement (Card Punches)” on page 190 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 197.

## Communication Controllers

Table 5 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 5. Configuration Guide for Communication Controllers

Devised	Statement Needed in Configuration File
3705	<ul style="list-style-type: none"> <li>• When running EP, TYPE 3705</li> <li>• When running NCP, not required</li> <li>• When running PEP for BSC lines, TYPE BSC_ADAPTER</li> <li>• When running PEP for ASCII lines, TYPE TELE2_ADAPTER</li> </ul>
3720	Not required when running NCP
3725	<ul style="list-style-type: none"> <li>• When running EP, TYPE 3705</li> <li>• When running NCP, not required</li> <li>• When running PEP for BSC lines, TYPE BSC_ADAPTER</li> <li>• When running PEP for ASCII lines, TYPE TELE2_ADAPTER</li> </ul>
3745	<ul style="list-style-type: none"> <li>• When running NCP, not required</li> <li>• When running PEP for ASCII lines, TYPE TELE2_ADAPTER</li> </ul>
9221 Integrated Token Ring	TYPE ICA_TOKENRING
9221 Integrated Ethernet	TYPE ICA_ETHERNET

See “RDEVICE Statement (Communication Controllers)” on page 193 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

## Defining I/O Devices

- 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 209 in the description of the RDEVICE statement in Chapter 6, “The System Configuration File.” For information on defining ESCON-attached devices, see the appropriate device tables in this chapter.

---

## Other Devices

Table 6 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 6. Configuration Guide for Other Devices*

Device	Statement Needed in Configuration File
3088	Not required
3172	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
8232	Not required
CTCA	TYPE CTCA
OSA	Not required

---

## 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. The parm disk must reside on the IPL volume. The system definition files on the parm disk include:

- The main system configuration file, usually called SYSTEM CONFIG, which 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, which contains information about the creation and configuration of logos, including the file names and file types of the different logo files. For details, see Chapter 7, “The Logo Configuration File,” on page 279.

**Note:** 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 in console mode and change the parm disk extent on the Stand-alone Program Loader screen to the alternate parm disk extent.

---

### Summary of System Configuration File Statements

Table 7 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 7. System Configuration File Statements (SYSTEM CONFIG)

Statement	Description	Page
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.	67
ASSOCIate EXit	Assigns one or more entry points or external symbols to an exit point during initialization.	60
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.	64
CHARACTER_DEFAults	Establishes system-wide defaults for the logical character delete, escape, line delete, line end, and tab symbols.	67

## System Configuration File

Table 7. System Configuration File Statements (SYSTEM CONFIG) (continued)

Statement	Description	Page
CP_ACCess	Defines the list of disks that CP accesses immediately when it brings the user directory on line.	70
CP_ADDON_INITIALIZE_ROUtines	Lists the installation-added entry points that will be called during system initialization.	72
CP_OWNeD	Defines and generates a list of up to 255 CP-owned volumes.	73
CPXLoad	Loads a file containing customer-written CP routines from the parm disk or a CP-accessed disk into CP's virtual storage during initialization.	76
DEFine ALIAS	Defines a new alias for an existing CP command on the system during initialization.	81
DEFine COMmand/CMD	Defines a new CP command or a new version of an existing CP command on the system during initialization.	84
DEFine DIAGnose	Defines a new DIAGNOSE code on the system during initialization.	90
DEFine EXit	Defines a new exit point in CP during initialization.	94
DEFINE LAN	Defines a VM LAN segment.	97
DEFINE VSWITCH	Creates a CP system-owned switch (a Virtual Switch) to which virtual machines can connect.	100
DEVICes	Tells CP what to do with various devices at IPL.	105
DISable COMmand/CMD	Prevents CP from processing requests for the specified CP command during and after initialization.	110
DISable DIAGnose	Prevents CP from processing requests for the specified locally-developed DIAGNOSE codes during and after initialization.	112
DISable EXits	Prevents CP from calling all entry points and external symbols associated with one or more exit points during and after initialization.	114
DISTRIBUTE	Specifies the distribution features for the local system.	116
DRAin	Stops new operations on specified real devices.	118
EDEVice	Defines an emulated device that represents a real device.	121
EMERGENCY_MESSAGE_CONSoles	Defines consoles for system emergency messages.	124
ENable COMmand/CMD	Permits CP to process requests for the specified CP command during and after initialization.	126
ENable DIAGnose	Permits CP to process requests for the specified locally-developed DIAGNOSE codes during and after initialization.	128
ENable EXits	Permits CP to call all entry points and external symbols associated with one or more exit points during and after initialization.	130
ENFORCE_BY_VOLId	Enforces attachment of DASD devices by their VOLIDs on the ATTACH command.	132
EQUate	Groups names of systems that all have something in common and should thus be treated similarly.	133

Table 7. System Configuration File Statements (SYSTEM CONFIG) (continued)

Statement	Description	Page
EXTERNAL_SYNTAX	Adds locally-developed system configuration file statements to the system without modifying the system configuration file processor, HCPZSC ASSEMBLE.	135
FEATURES	Establishes certain attributes of the system at system initialization.	136
FORM_DEFAULT	Generates default user form names.	150
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.	152
IMBED	Specifies files to imbed into the SYSTEM CONFIG file at IPL.	155
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.	157
IODF	Specifies the IODF that HCD will use in its control of the I/O configuration.	158
JOURNALING	Specifies whether CP should include the journaling facility in the system being generated; establishes attributes of the facility.	160
LOGO_CONFIG	Specifies the name and type of a logo configuration file.	164
MODIFY_COMMAND/CMD	Redefines an existing CP command on the system during initialization.	165
MODIFY_DIAGNOSE	Redefines an existing DIAGNOSE code on the system during initialization.	169
MODIFY_EXIT	Redefines or removes an existing dynamic CP exit point during initialization.	172
MODIFY_LAN	Modifies the properties of a VM LAN segment.	175
MODIFY_PRIV_CLASSES	Changes the privilege classes and establishes initial values.	176
MODIFY_VSWITCH	Modifies the properties of an existing Virtual Switch.	178
OPERATOR_CONSOLES	Defines a list of consoles from which CP can choose an operator console.	180
PRINTER_TITLE	Specifies the printed output classes that are to contain classification titles.	182
PRIV_CLASSES	Changes the privilege class that authorizes certain internal CP functions.	184
PRODUCT	Defines a product or feature to the system.	186
RDEVICE	Adds to the system's definition of a set of real devices.	188
SAY	Writes a line of text to the operator's console during initialization.	217
SET	Defines default system configuration values.	219
START	Restarts devices after they have been drained; changes the processing options in effect for devices.	220
STORAGE	Configures the use of real storage.	222
SYSTEM_DATEFORMAT	Sets the system-wide default date format for commands that provide multiple date formats.	225

## System Configuration File

Table 7. System Configuration File Statements (SYSTEM CONFIG) (continued)

Statement	Description	Page
SYSTEM_IDentifier	Creates a system identifier for the processor on which you run z/VM.	226
SYSTEM_IDENTIFIER_DEfault	Provides CP with a default system identifier.	228
SYSTEM_RESidence	Describes the layout of the system residence disk.	230
SYSTEM_USERids	Specifies user IDs that will perform special functions during and after IPL.	232
THROTtle	Limits the number of I/O operations that guest operating systems can initiate to a specific real device.	236
TIMEZONE_Boundary	Tells CP which time zone to choose at IPL.	237
TIMEZONE_DEfinition	Defines system time zones according to their distance from UTC.	239
TOLERATE_CONFIG_ERRors	Marks sections of the system configuration file in which CP is not to tolerate errors.	241
TRANSLATE_TABLE	Specifies replacements for standard translation tables.	243
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.	247
USER_DEFAULTS	Defines defaults to be used for <ul style="list-style-type: none"> <li>• global lines per page values for virtual printers and consoles defined on the system, and</li> <li>• querying other users' POSIX database information and having their POSIX security values changed.</li> </ul>	248
USER_VOLUME_EXclude	Defines volumes to be excluded from the user volume list.	251
USER_VOLUME_INclude	Defines user volumes by a generic volume identifier.	253
USER_VOLUME_List	Generates a list of user DASD volumes not to be used for paging, spooling, directory, or temporary disk space.	255
VMLAN	Controls global attributes for VM Guest LAN and Virtual Switch support.	257
XLINK_DEVICE_DEfaults	Changes the defaults for the CSE track location format for particular types of DASD.	259
XLINK_SYSTEM_EXclude	Specifies the systems that CP is to exclude from cross-system link.	263
XLINK_SYSTEM_INclude	Specifies the systems that CP is to include in cross-system link.	264
XLINK_VOLUME_EXclude	Specifies the DASD volumes that CP is exclude from cross-system link.	266
XLINK_VOLUME_INclude	Specifies the DASD volumes that CP is include in cross-system link.	268
XSPOOL_SYStem	Specifies the systems that are to participate in cross-system commands and spooling operations.	271
XSPOOL_TRace	Establishes the number of pages of CP storage allocated for the XSPOOL trace tables.	273
XSPOOL_XLIST_INput	Defines virtual machines whose input spool files will not participate in cross-system functions.	274

Table 7. System Configuration File Statements (SYSTEM CONFIG) (continued)

Statement	Description	Page
XSPPOOL_XLIST_OUTput	Defines virtual machines whose output spool files will not participate in cross-system functions.	276

## 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

## System Configuration File

```
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

```
B0BVM1: B0BVM2: 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.

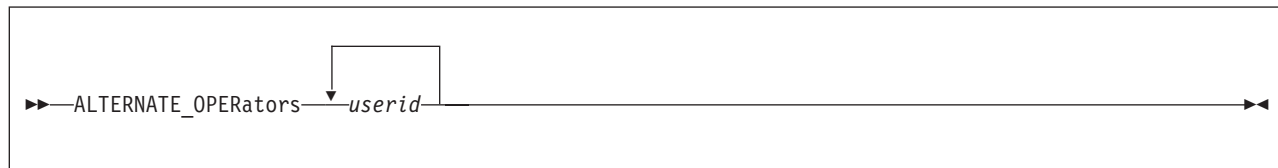
```
User_Volume_List  ESAUS1  ESAUS2  ESAUS3
:
User_Volume_List  ESAUS4  ESAUS5  ESAUS6
```

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, refer to the *z/VM: CP Commands and Utilities Reference* book.

### 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

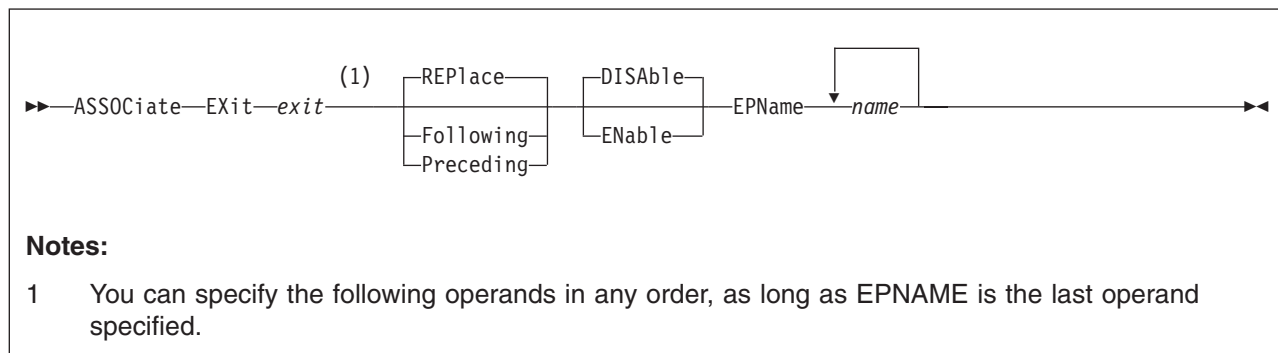
1. If alternate operator user IDs are specified and the primary system operator logs off, CP will select the default operator ID before the alternates if the default operator is logged on or disconnected.
2. If no alternate operator user IDs are specified or no alternate can be selected when the primary system operator logs off, there will be no operator until one of 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 on the default primary system operator ID, see the "SYSTEM\_USERIDS Statement" on page 232 or "SYSOPR (Required)" on page 687.
4. For more information on defining operator privilege class, see the "PRIV\_CLASSES Statement" on page 184 or "SYSFCN (Optional)" on page 673.

## 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



### 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 using the ASSOCIATE EXIT command. For more information, see the *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.) 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 (page 76) to load the customer-written CP routines for the exit point into CP's 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 about the CPXLOAD command, see the *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 the *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,

## System Configuration File

- Specify an ENABLE EXITS statement (page 130) further on in your system configuration file,
- Enter an ASSOCIATE EXIT command after initialization and specify the ENABLE operand, or
- Enter an ENABLE EXITS command after initialization.

For more information about the ASSOCIATE EXIT or ENABLE EXITS command, see the *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 the *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.

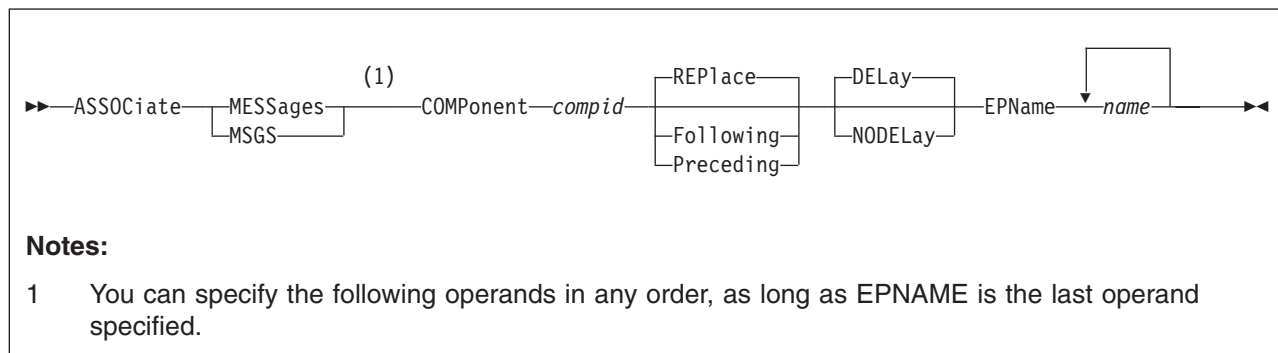
7. To display the real address of the CP exit block for a specific exit point after initialization, use the LOCATE XITBK command. For more information, see the *z/VM: CP Commands and Utilities Reference*. Again, if you have not associated one or more entry points or external symbols with the specified exit point, there is no CP exit block for CP to locate and display. Instead, CP issues error message HCP2752E.
8. To display the real address of the CP indirect call locator block for a specific exit point, use the LOCATE ICLBK command. For more information, see the *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 (page 172) or command. For more information about the MODIFY EXIT command, see the *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 the *z/VM: CP Commands and Utilities Reference*.
11. To remove the customer-written CP routines from CP's system execution space:
  - a. Use the DISASSOCIATE command to revoke all entry point and external symbol assignments made with the ASSOCIATE EXIT statement (or command)
  - b. Use the CPXUNLOAD command to unload the customer-written CP routines.

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 the *z/VM: CP Exit Customization* book.

### 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 EName 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 EName hcpsrc04

## ASSOCIATE MESSAGES / MSGS Statement



### Purpose

Test 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 the *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 (VM/ESA) is HCP, which is, by default, the last message repository in the search list. For more information, see Usage Note 1.

#### **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 CP's 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 about the CMS GENMSG command, see the *z/VM: CMS Commands and Utilities Reference*.

## System Configuration File

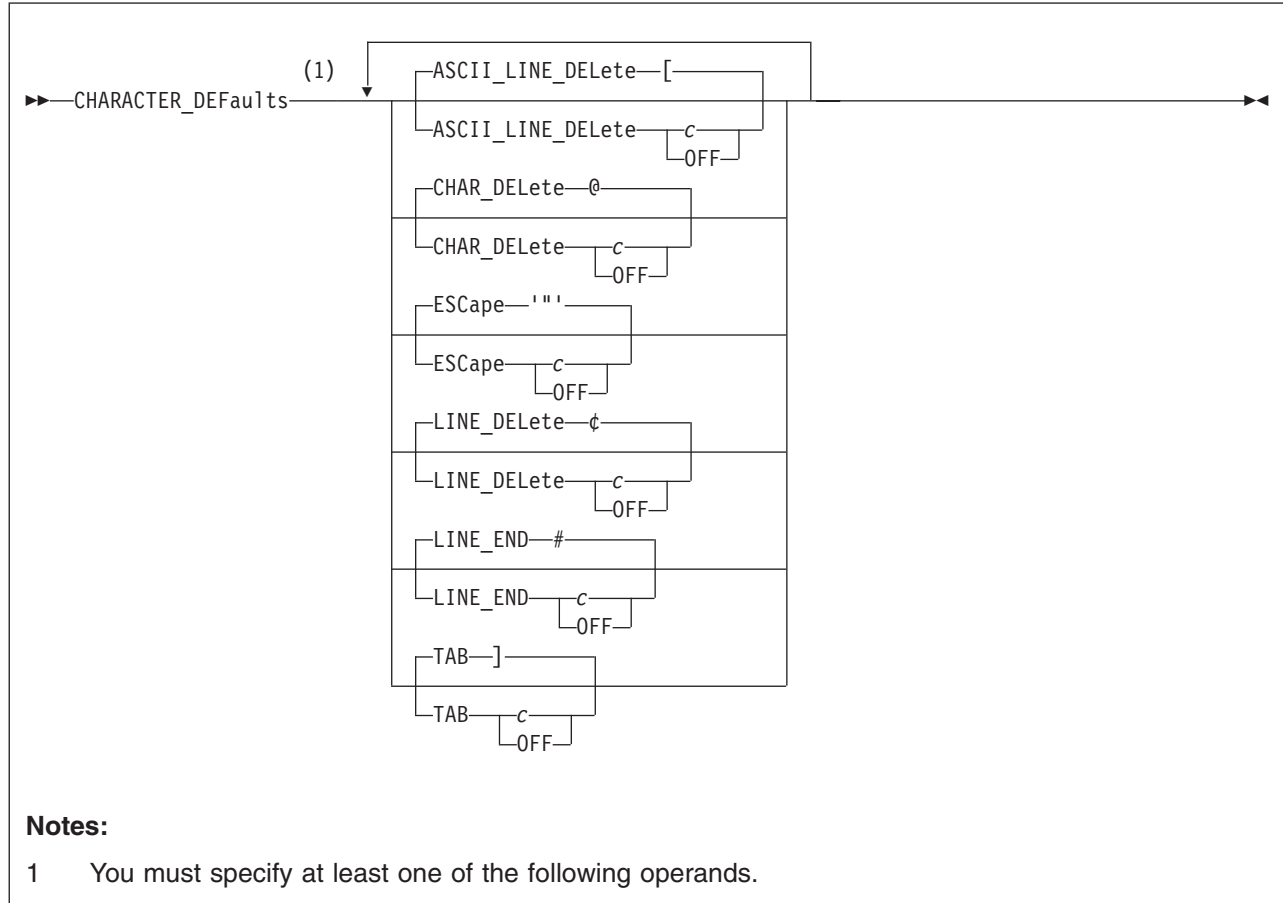
- b. Load the message repository file using the CPXLOAD statement (page 76) or command. For more information about the CPXLOAD command, see the *z/VM: CP Commands and Utilities Reference*.
3. If CP cannot find the entry point you specified for EPNAME when the message is being displayed, CP displays message 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 the *z/VM: CP Commands and Utilities Reference*.
5. To remove the message repository file from CP's 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 about the DISASSOCIATE or CPXUNLOAD command, see the *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 about the HCPCONSL macroinstruction and about using your own message repositories, see the *z/VM: CP Exit Customization* book.

## 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
```

## CHARACTER\_DEFAULTS Statement



### 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 (`"`).

## System Configuration File

### **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 control statement in the system directory (page 572)
  - By having the user issue the CP TERMINAL command after logging on. For more information, see the *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  
  
b X'6A'  
    SYSTAB TABCHAR
```

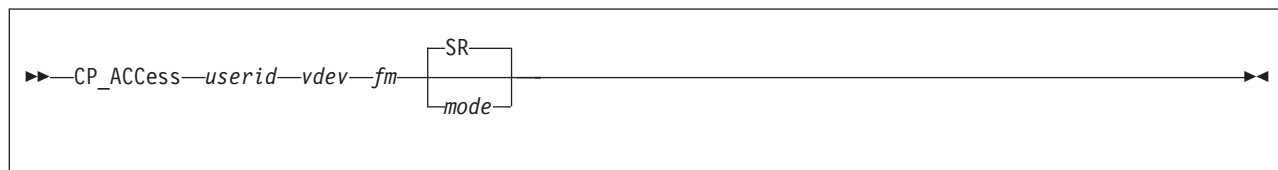
```
[ X'AD'  
  SYSALDEL LINDEL
```

## 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 user's entry in the system 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 CPDISK command. For more information about the QUERY CPDISK command, 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 the *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 control statement. For more information about the OPTION statement, see "OPTION Directory Control Statement (General)" on page 533.

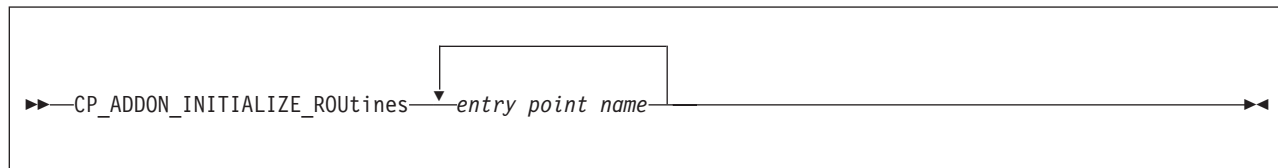
## Examples

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.

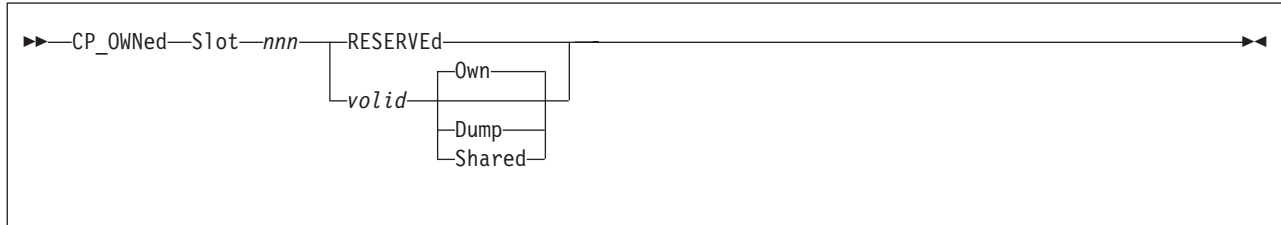
### Examples

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



### 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.

If you are not using cross system spool (XSPOOL) file sharing, you can use any volumes listed on a CP\_OWNED statement and containing spool space to read and write spool files. (CP writes spool files only once, when they are created.)

If you are using cross system spool (XSPOOL) file sharing, physical sharing of the spooling DASD provides shared access to spool files for all systems in the complex. Although physically this is both read and write sharing, logically the sharing is read-only. Therefore, you must identify some volumes as available for writing the spool files created on this system, and others as available only for reading spool files created on other systems. You can identify these volumes using the OWN and SHARED options of the CP\_OWNED statement.

**Note:** We strongly recommend that you do not share the system residence volumes and paging volumes among systems in a cross system spooling complex.

### 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.

#### RESERVED

tells CP to reserve the slot for future use.

#### *valid*

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

#### Own

(the default) tells CP that the system you are generating owns the specified volume and that only this system can create or destroy spool files on this volume. The other systems in the cross system spooling complex, which define

## System Configuration File

the specified volume with the SHARED option, have shared access to this system's spool files because they can read the data on this volume.

### Dump

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

### Shared

tells CP that another system in the cross system spooling complex owns the specified volume. This system cannot create or destroy spool files on the identified volume. It can, however, read spool files previously written by the owning system.

## 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 on the CPFMTXA utility, see the *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 about the CP ATTACH command, see the *z/VM: CP Commands and Utilities Reference*.
3. If you want to add new volumes to a system that has already been IPLed, add new volume identifications either in empty slots in the CP-owned volume list or at the end of the list, using the CP DEFINE CPOWNED command. For more information about the CP DEFINE CPOWNED command, see the *z/VM: CP Commands and Utilities Reference*.

By adding new volumes in empty slots or at the end of the CP-owned list, you do not affect the warm start data on the volumes already listed in the CP-owned list. This means you can warm start the system without disturbing the relative entry number that locates system spool buffers.

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 valid 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 are to be preserved over such a change should be dumped to tape using the CP 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 about the CP SPXTAPE command, see the *z/VM: CP Commands and Utilities Reference*.
  5. For a single VM/ESA system, that is, one that is not using cross system spooling, you should specify the OWN option for all CP volumes. Do not specify the SHARED option.
  6. If you are using cross system spooling:
    - a. You must include all volumes that have SPOL extents in a CP\_OWNED statement for each system, with the same slot number in each system's CP\_OWNED statement. We suggest that you group all volumes with SPOL extents by system and place them first in the CP-owned volume list.
    - b. If you specify the DUMP option for a specific volume, CP treats that volume the same way as it would if you had specified the OWN option; only this system may allocate spool space for dumps on this volume.  
You must specify the SHARED option for the same volume in the CP-owned volume lists of all the other systems in the CSE complex.
    - c. In the CP-owned volume list, you cannot change the order of the volumes that contain SPOL space unless you perform a clean start on all systems in the complex. If you do change the order, you may have to use the CP SPXTAPE command to avoid losing spool files. However, if you add new SPOL space volumes to the end of the CP-owned volume lists of the associated systems without disturbing the slot numbers of existing volumes, a clean start is not required.
  7. At system initialization (IPL), if more than one DASD volume has the same volume serial number (*valid*) and that *valid* is in the CP or user volume list, the volume with the lowest device number is attached to the system. This rule does not apply to duplicates of the system residence (IPLed) volume.
  8. The list of CP-owned volumes generated by CP\_OWNED statements completely supersedes the list of CP-owned volumes generated by the SYSCPVOL macroinstruction in HCPSYS ASSEMBLE.

## Examples

1. To define a CP-owned volume list that contains:
  - A system residence volume, which your system owns
  - A volume for spool and temporary disk space, which your system owns
  - A volume for spool space, which another system owns
  - A volume for dump space, which your system owns
  - Three empty slots for volumes you will be getting shortly

use the following CP\_OWNED statements:

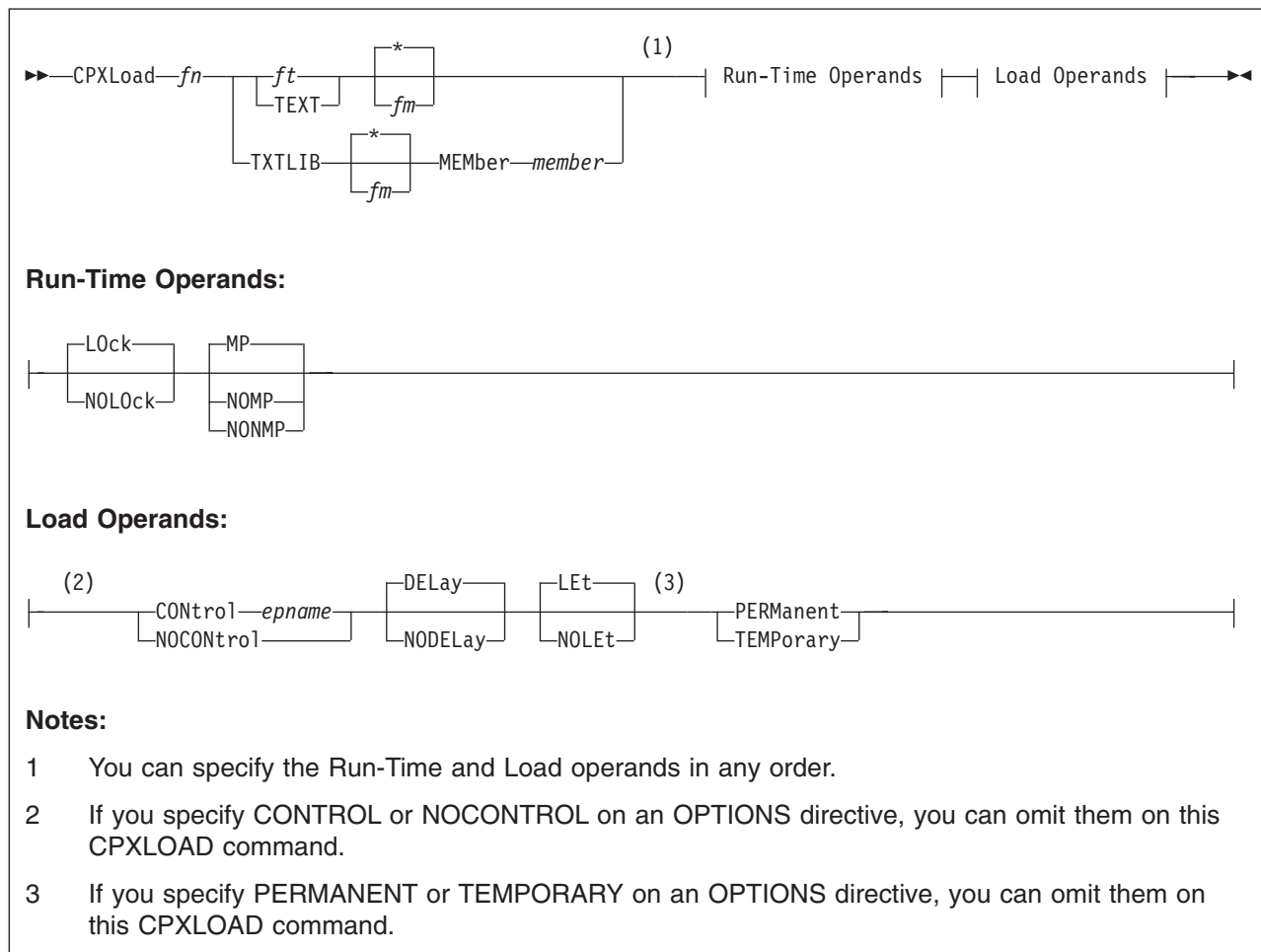
```

CP_Owned Slot 001 esares Own /* System Residence volume */
CP_Owned Slot 002 essys1 Own /* Spool and T-disk space */
CP_Owned Slot 003 essys2 Shared /* Spool space for VM2 */
CP_Owned Slot 004 sysdmp Dump /* Dump space */

CP_Owned Slot 005 Reserved /* Leave some slots open in */
CP_Owned Slot 006 Reserved /* case we need to add some */
CP_Owned Slot 007 Reserved /* extra spool space later. */

```

## CPXLOAD Statement



## Purpose

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

You can also load customer-written CP routines into CP's system execution space after initialization using the CPXLOAD command. For more information, see the *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.

## 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 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

## System Configuration File

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 its 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 non-commented 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 about the CP\_ACCESS statement, see page 70.
4. You can specify run-time 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 about the OPTIONS directive and several other directives, see the *z/VM: CP Exit Customization* book.
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 (page 60) or command. For more information about the ASSOCIATE EXIT command, see the *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 (page 64) or command. For more information about the ASSOCIATE MESSAGES or MSGS command, see the *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 the *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 the *z/VM: CP Commands and Utilities Reference*.
9. To remove CPXLOADed files from CP's system execution space:
  - a. Use the DISASSOCIATE command to revoke all entry point and external symbol assignments made with the ASSOCIATE EXIT, MESSAGES, or MSGS statements or commands
  - b. Use the CPXUNLOAD command to unload the customer-written CP routines.For more information about the commands listed above, see the *z/VM: CP Commands and Utilities Reference*.
10. For more information about loading customer-written CP routines into CP's system execution space, about run-time and load operands, and about CPXLOAD directives, see the *z/VM: CP Exit Customization* book.
11. 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.

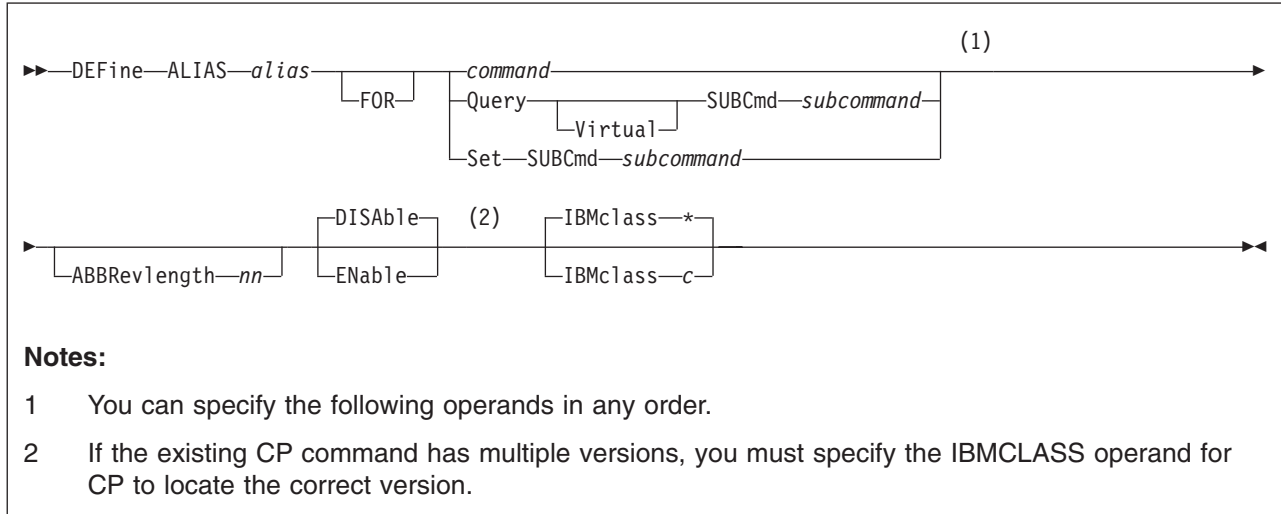
The load ID that CP assigns is a 1-digit to 10-digit decimal number between 0 and 2,147,483,647. The first time you use the CPXLOAD statement (or command), CP assigns that set of customer-written CP routines a load ID of 0. CP increases the load ID by 1 for each subsequent CPXLOAD request. If all the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



## DEFINE ALIAS Statement



## 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 using the DEFINE ALIAS command. For more information, see the *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.

## System Configuration File

### 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.) 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 system storage.
- D** this is a command for system-wide control of spool files.
- E** this is a command to examine system 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

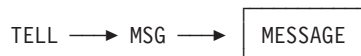
1. You can create many aliases for one command, but you cannot create an alias for an alias. For example, the CP MSG command is actually an alias for the CP MESSAGE command:



You can create another alias for the MESSAGE command:



But you cannot create an alias for MSG:



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 defining it, use the ENABLE COMMAND or CMD statement (page 126) or command. For more information about the ENABLE COMMAND or CMD command, see the *z/VM: CP Commands and Utilities Reference* book.
6. To deactivate an alias after defining it, use the DISABLE COMMAND or CMD statement (page 110) or command. For more information about the DISABLE COMMAND or CMD command, see the *z/VM: CP Commands and Utilities Reference* book.
7. Once 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 more information about creating an alias, see the *z/VM: CP Exit Customization* book.

## Examples

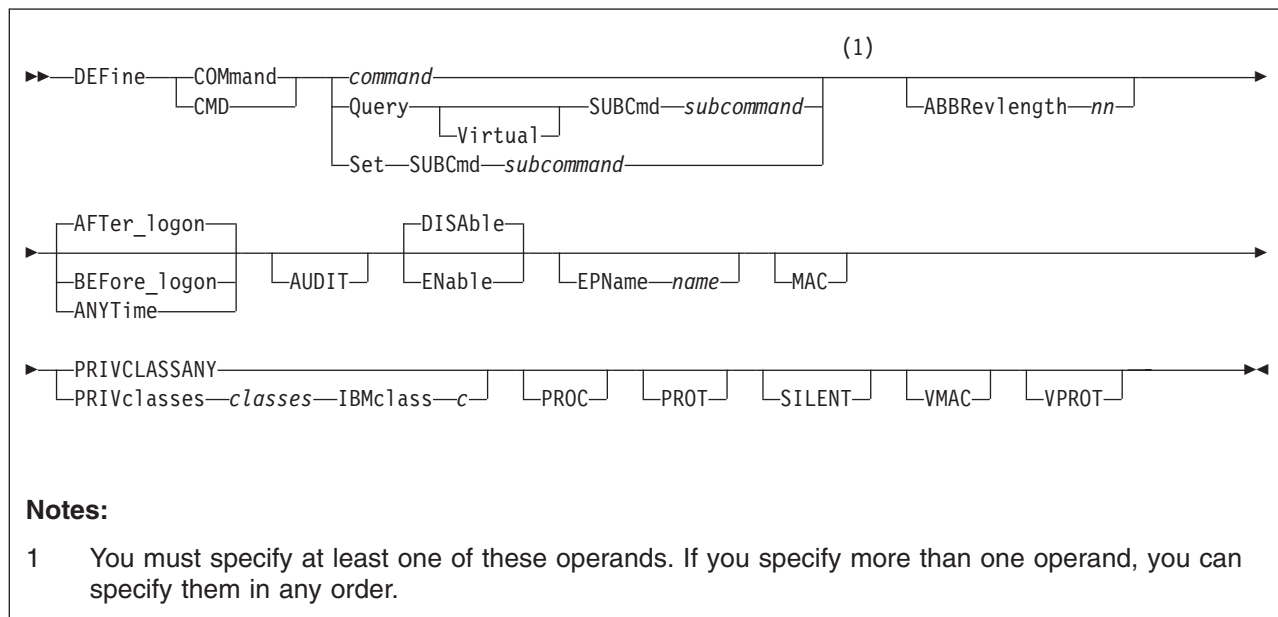
1. To have CP define GOODNIGHT as an alias for the CP SHUTDOWN command and make it available immediately after initialization, use the following:
 

```
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



### 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 using the DEFINE COMMAND or CMD commands. For more information, see the *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.

### **ABBRlength** *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.) 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.

## System Configuration File

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).

#### Notes:

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 system storage.
- D** this is a command for system-wide control of spool files.
- E** this is a command to examine system 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 the *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 1 command table entry block. If the command has more than 1 privilege class, CP has 1 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 1 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 and the following table lists any other operands you are required to supply when defining a new command or command version.

When Adding a New:	You Must Specify:
Command	EPNAME PRIVCLASSANY - or - EPNAME PRIVCLASSES IBMCLASS
Command Version	PRIVCLASSANY - or - PRIVCLASSES IBMCLASS

3. To load the command processing code into CP's system execution space, use the CPXLOAD statement (page 76) or command. For more information about the CPXLOAD command, see the *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 after defining it, use the ENABLE COMMAND or CMD statement (page 126) or command. For more information about the ENABLE COMMAND or CMD command, see the *z/VM: CP Commands and Utilities Reference*.
5. To display the real 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 the *z/VM: CP Commands and Utilities Reference*.

## System Configuration File

6. To change the definition of an existing CP command, use the MODIFY COMMAND or CMD statement (page 165) or command. For more information about the MODIFY COMMAND or CMD command, see the *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 after defining it, use the DISABLE COMMAND or CMD statement (page 110) or command. For more information about the DISABLE COMMAND or CMD command, see the *z/VM: CP Commands and Utilities Reference*.
9. To remove the command processing code from CP's system execution space, use the CPXUNLOAD command. For more information, see the *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 about user-defined commands, see the *z/VM: CP Exit Customization* book.

## 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 hcpxmgs, /* 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. */
```



**Important Note:** 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 hcpcmbe, /* command to use instead */
                          IBMClass g, /* of CP BEGIN. */
                          PrivClasses g

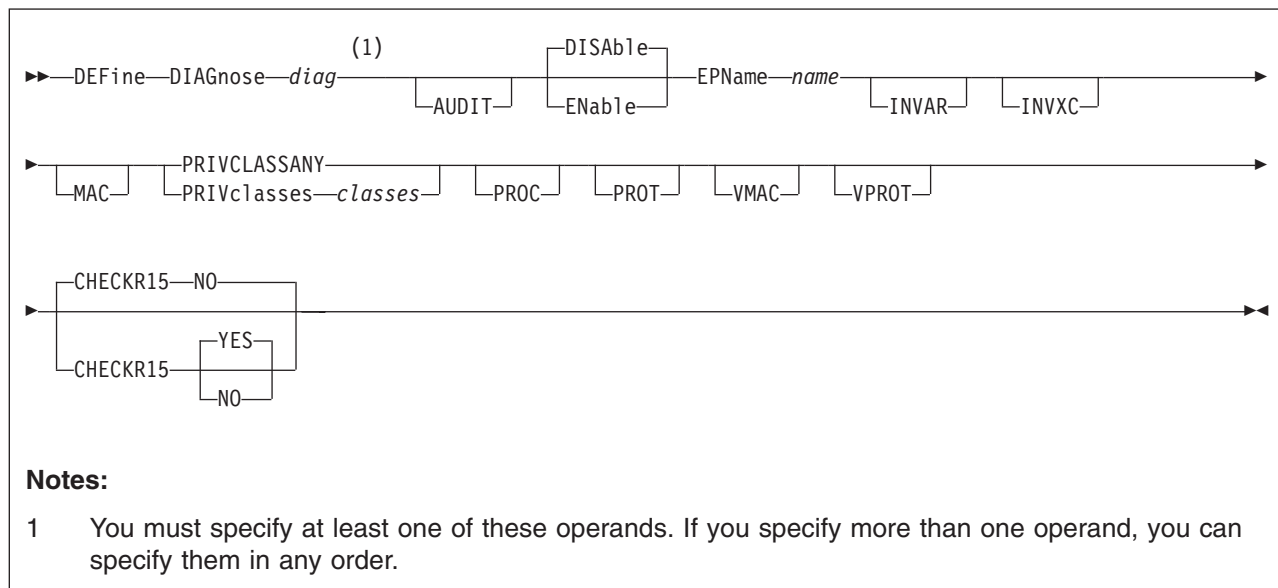
Define Cmd resume EPName hcpcmbe, /* Create "CP RESUME" command */
                  IBMClass g, /* to use instead of CP */
                  PrivClasses g /* BEGIN. */
```

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



## 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 the *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.

### **DISAb1e**

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.) 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 this DIAGNOSE code if the virtual machine issuing the DIAGNOSE code is in host access register mode.

**INVXC**

tells CP not to process this DIAGNOSE code if the virtual machine issuing the DIAGNOSE code is in Enterprise System Architecture/Extended Configuration (ESA/XC) 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.

## System Configuration File

### 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 CP's system execution space, use the CPXLOAD statement (page 76) or command. For more information about the CPXLOAD command, see the *z/VM: CP Commands and Utilities Reference*.
3. To activate a new DIAGNOSE code after defining it, use the ENABLE DIAGNOSE statement (page 128) or command. For more information about the ENABLE DIAGNOSE command, see the *z/VM: CP Commands and Utilities Reference*.
4. To change the definition of an existing DIAGNOSE code, use the MODIFY DIAGNOSE statement (page 169) or command. For more information about the MODIFY DIAGNOSE command, see the *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 the *z/VM: CP Commands and Utilities Reference*.
6. To display the real address of the CP DIAGNOSE code table block for a DIAGNOSE code after initialization, use the LOCATE DGNBK command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
7. To deactivate a DIAGNOSE code after defining it, use the DISABLE DIAGNOSE statement (page 112) or command. For more information about the DISABLE DIAGNOSE command, see the *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 CP's system execution space, use the CPXUNLOAD command. For more information, see the *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 the *z/VM: CP Exit Customization* book.

## Examples

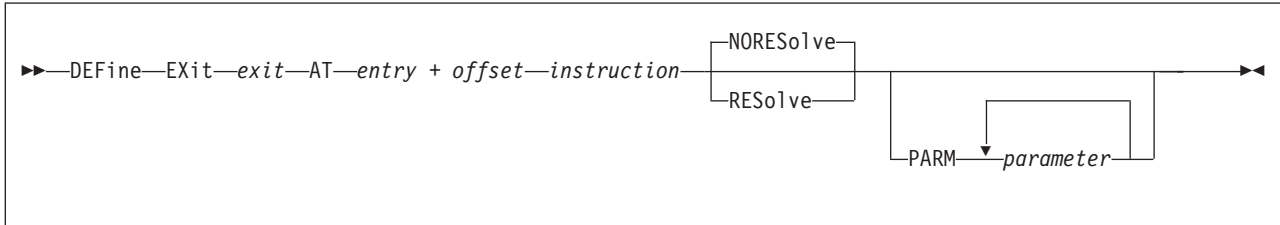
- To have CP define a general purpose DIAGNOSE code (X'10C') that will be processed by entry point HCPSRC00, use the following:
 

```
Define Diagnose 10c EPname hcpsrc00,
                  PrivClasses g
```
- 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
```
- 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 using the DEFINE EXIT command. For more information, see the *z/VM: CP Commands and Utilities Reference* book.

### 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> < or. -<reg>
<reg> ::= G<digit> | R<digit>
<digit> ::= 0...15 | 0...9, A...F
<disp> ::= 0...7FFF
    
```

## Usage Notes

1. To load the exit processing code into CP's system execution space, use the CPXLOAD statement (page 76) or command. For information about the CPXLOAD command, see the *z/VM: CP Commands and Utilities Reference*.
2. To assign entry points or external symbols to an exit point, use the ASSOCIATE EXIT statement (page 60) or command. For information about the ASSOCIATE EXIT command, see the *z/VM: CP Commands and Utilities Reference*.
3. To activate a new exit after defining it, use the ENABLE EXITS statement (page 130) or command. For information about the ENABLE EXITS command, see the *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 (page 172) or command. For information about the MODIFY EXIT command, see the *z/VM: CP Commands and Utilities Reference*.
5. To display information about an exit (status, entry point name, and parameters), use the QUERY EXITS command. For more information, see the *z/VM: CP Commands and Utilities Reference* book.
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 the *z/VM: CP Exit Customization* book. 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 exits, see the *z/VM: CP Exit Customization* book.

## DEFINE EXIT

### Examples

1. To define an exit that will be entered every time a MESSAGE command is issued, use the following:

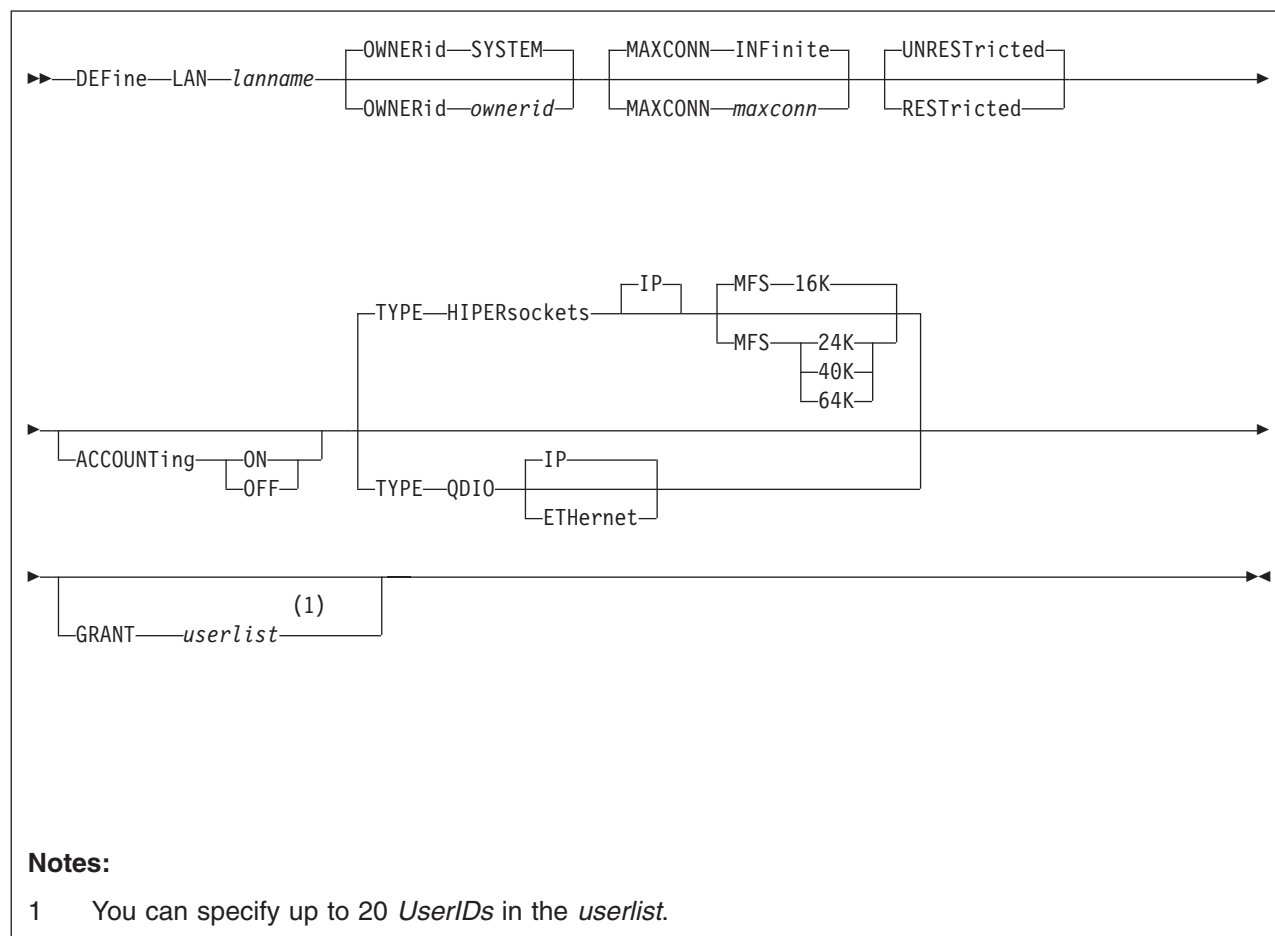
```
Define Exit f422 At hcpmsgms + 4 41700000 /* Exit from MESSAGE cmd */  
  
Enable Exits f422 /* */
```

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



## Purpose

Use the DEFINE LAN statement to create a VM LAN segment in the host CP system. 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). Each VM LAN segment is qualified by *ownerid* and *lanname*. See the *z/VM: CP Commands and Utilities Reference* for more information on these commands.

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 175.

## 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

## DEFINE LAN

### **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.

### **RESTRictedIUNRESTRicted**

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. 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 VM 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 16K. The MFS operand is not valid for QDIO but the effective MFS is 8992 for a QDIO adapter.

### **TYPE QDIO**

defines a VM LAN for use by simulated QDIO adapters. A QDIO LAN can only accept connections from a simulated QDIO adapter.

**IPIEthernet**

indicates whether the transport for the LAN is ETHERNET or IP. An ETHERNET LAN operates at the Layer 2 level of the OSI model.

**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.

## Examples

1. To create a user LAN named QDIO that will allow up to 16 connections, specify the following:
 

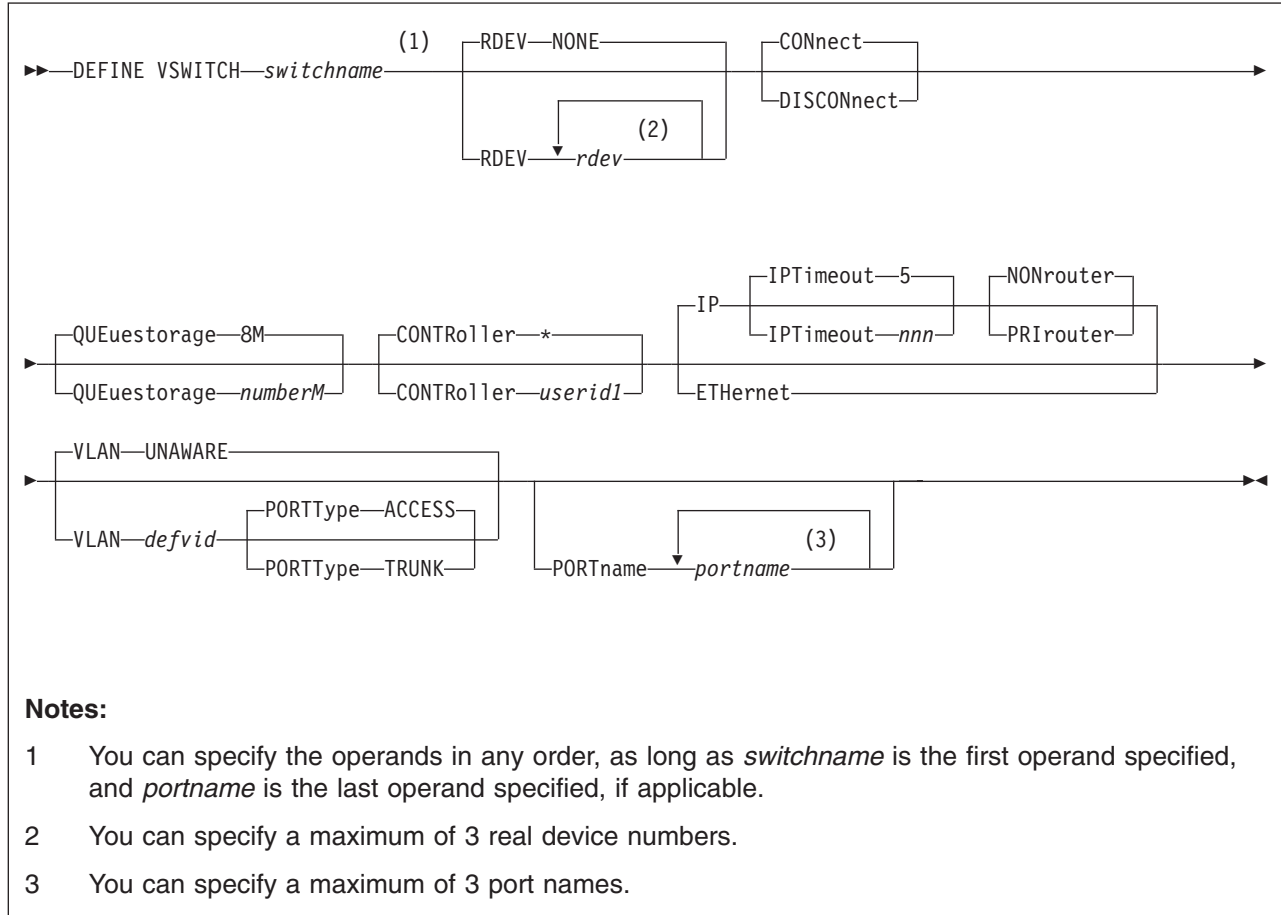
```
define lan type qdio maxconn 16
```
2. To create a system LAN named INEWS that allows up to 100 connections, specify the following:
 

```
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

### DEFINE VSWITCH Statement



### Purpose

Use the `DEFINE VSWITCH` statement 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 a simulated QDIO Network Interface Card (NIC) and connect it to this switch with the `NICDEF` directory statement. See “`NICDEF` Directory Control Statement (Device)” on page 529 for more information about the statement.

The `MODIFY VSWITCH` statement can be used to define the initial access list of the virtual switch. See “`MODIFY VSWITCH` Statement” on page 178 for more information.

### 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.

#### **RDEV** *rdev*

is a real device address to be used to connect the Virtual Switch to a QDIO OSA Express device. Specify each real device number as a 1- to 4-digit

hexadecimal number. You can specify a maximum of three real device numbers. If you specify more than one device number, each must be separated from the others by at least one blank.

Each real device address represents a trio of devices. For example, specifying **RDEV 111 222 333** means that the first devices, 111-113, are used to provide the connection to the real hardware LAN segment. If there is a problem with the connection, devices 222-224 are used next to provide the connection, and if those devices fail to connect, devices 333-335 are used. This feature provides dynamic recovery for OSA Express device failures.

**RDEV NONE** means that the Virtual Switch should not be connected to the real LAN segment.

### **CONnect**

indicates that the device identified by the RDEV keyword must be activated, and traffic must flow through the device to the real LAN segment.

### **DISCONnect**

indicates that the device identified by the RDEV keyword must not be activated, and no traffic must flow through the device to the real LAN segment.

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.

### **QUEestorage *numberM***

indicates the upper limit of the amount of fixed storage CP and Queued Direct I/O Hardware Facility will use for buffers.

*number* defines the maximum number of megabytes of storage that can be consumed for QDIO queues. Fixed storage is allocated as needed based on network traffic, until the maximum of *numberM* are allocated.

*number* is a number from 1 to 8. 8M is the default value.

### **CONTRoller \* | *userid1***

identifies the z/VM user ID that controls the OSA Express device connected at the device address identified by *rdev*. **CONTROLLER \*** means CP selects from any of the eligible TCP/IP stacks. See Usage Note 3 on page 102 for more information about the function of a controller.

If you specify multiple real devices on the RDEV keyword, specify **CONTROLLER \***, or allow it to default. The controller functions are then spread across multiple z/VM TCP/IP stacks, providing more flexibility in case of a failure.

### **IPIETHERnet**

indicates whether the transport for the Virtual Switch is ETHERNET or IP. An ETHERNET Virtual Switch operates at the Layer 2 level of the OSI model while an IP Virtual Switch operates at Layer 3.

### **IPTimeout *nnn***

indicates the length of time in minutes that a remote IP address table entry remains in the IP address table for the Virtual Switch.

*nnn* is a number from 1 to 120. 5 minutes is the default value.

### **NONrouter**

indicates that the OSA Express device identified by the RDEV keyword will not act as a router to the Virtual Switch. If a datagram is received at this device for an unknown IP address, the datagram will be discarded. This is the default.

## DEFINE VSWITCH

### **PRIRouter**

indicates that the OSA Express device identified by the RDEV keyword will act as a primary router to the Virtual Switch. If a datagram is received at this device for an unknown IP address, the datagram will be passed to the Virtual Switch.

### **VLAN UNAWARE**

indicates the Virtual Switch does not support IEEE standard 802.1Q. 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). This is the default.

### **VLAN *defvid***

Defines the Virtual Switch as a VLAN-aware switch supporting IEEE standard 802.1Q. The *defvid* defines the default VLAN ID to be associated with untagged frames received and transmitted by the Virtual Switch. It is a number from 1 to 4094. A VLAN-aware virtual switch provides VLAN controls at the switch level (with SET VSWITCH GRANT VLAN and PORTTYPE commands) that may not be overridden by a guest host.

### **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 *defvid*, untagged traffic sent or received by the guest is associated with the default VLAN ID (*defvid*) of the Virtual Switch..

### **PORTname *portname***

is a 1- to 8-character name that identifies the OSA Express adapter. You can specify a maximum of three port names. Multiple port names are used when different port names are needed for the multiple *rdevs* specified on the RDEV operand. See Usage Note 9 on page 103 for more information.

## 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's connection to a real hardware LAN segment through one of the *rdev* values specified on the RDEV operand 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:
  - If CONTROLLER *userid1* is specified on the DEFINE VSWITCH command or System Configuration statement, only *userid1* is selected.
  - If CONTROLLER \* is specified or allowed to default, CP selects from any eligible TCP/IP stack.

A TCP/IP stack becomes eligible when:

- An IUCV \*VSWITCH statement is included in its 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 stack has completed initialization.
- The stack has virtual device addresses available for CP to attach the control device.

The virtual address 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 address (*vdev*) that matches the *rdev* address specified on the DEFINE VSWITCH or SET VSWITCH command. See *z/VM: TCP/IP Planning and Customization* for more information about the VSWITCH CONTROLLER statement.

**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.

4. The IP transport type is IPv4 only. In order to support IPv6 through the Virtual Switch RDEV, the ETHERNET transport is required.
5. MODIFY VSWITCH cannot be used to change the type of transport. The Virtual Switch will need to be redefined.
6. Use the QUERY CONTROLLER command output to find the TCP/IP stack that is the Virtual Switch controller. Use the QUERY VSWITCH command to display information about the Virtual Switch.
7. CP manages the devices used to control a Virtual Switch's connection to a real LAN segment through an OSA Express device. CP attaches the devices to the z/VM TCP/IP virtual machine. CP also defines a device of type VSWITCH-OSD to the TCP/IP stack, concatenating *switchname* with *vdev* and "DEV" to form the device name and *switchname* with *vdev* and "LINK" to form the link name. These names appear in the TCP/IP query and trace information.

DEVICE and LINK statements must not be included in the TCP/IP configuration file for these devices.

**Note:** Adapter number 0 (link number 0) will be used when attaching the OSA Express device to the controller.

8. Multiple real devices and portnames can be specified on the RDEV and PORTNAME operands. This feature allows failover to an alternate real device in the event of a failure with the current OSA-Express device. 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. If your OSA-Express device requires a portname, specify one portname for each real device number.
9. When the real device identified by one of the *rdevs* is started, TCP/IP assigns the port name as the hardware adapter name. If an adapter name was already assigned by a previous connection, then the port name must be the same as assigned by any other connection in order to share the adapter. This includes sharing the OSA Express adapter with this logical partition or all other partitions.

## DEFINE VSWITCH

The PORTNAME operand is optional. However, some levels of the OSA Express adapters require that the PORTNAME operand is specified. When such an adapter is in use and the PORTNAME operand is omitted, an error message is displayed during switch initialization.

If the device is already started, you must stop it by issuing the SET VSWITCH *switchname* DISCONNECT command before changing the port name.

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.

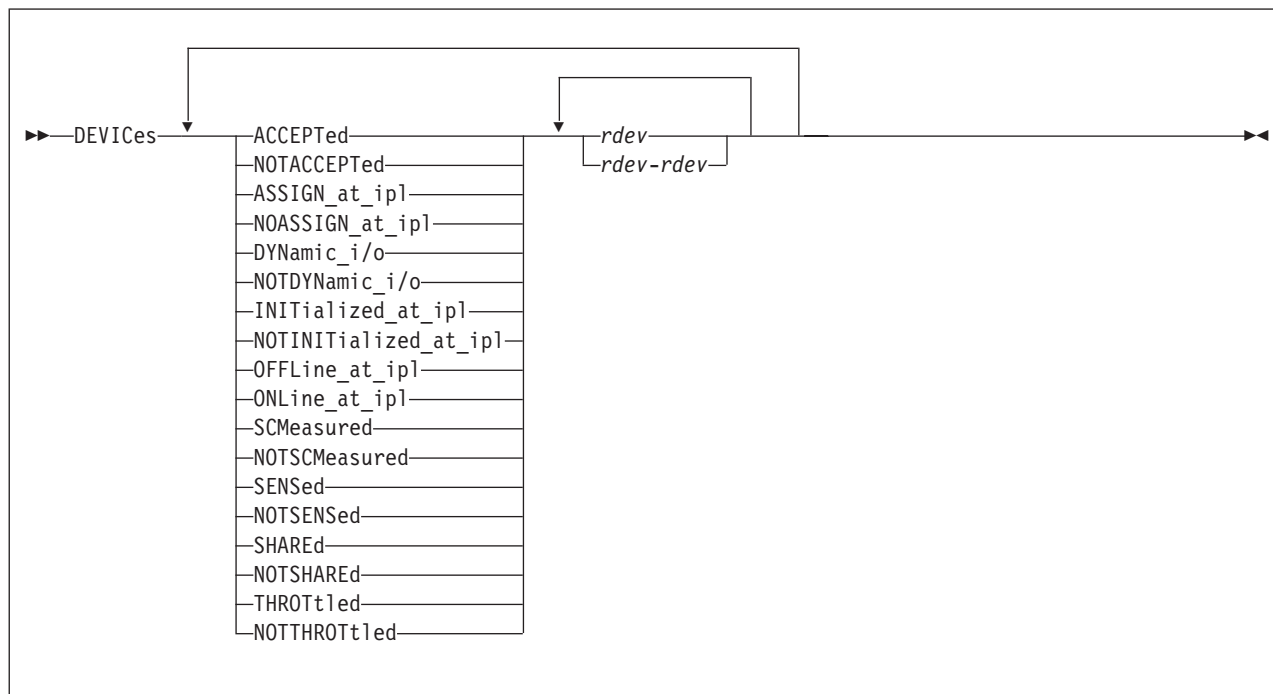
## Examples

1. To define a switch named BIGANG that connects to a real LAN through device fd00, specify the following:

```
define vswitch bigang rdev fd00
```



## DEVICES Statement



### Purpose

Use the DEVICES statement to tell CP how to handle specific devices during initialization. You can tell CP whether to:

- Accept (and build real device blocks for) specified devices
- Allow dynamic changes for specified devices
- Initialize the specified devices during IPL
- Measure the subchannels for specified devices
- Assign the tape drive to the system when the device is being brought online.
- Use the information returned from a sense ID request to help define the device
- Share DASD between independent operating systems
- Limit (throttle) 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**

tells CP to accept 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.

## DEVICES

### **NOTACCEPTed**

tells CP not to 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 using RDEVICE statements in the system configuration file or using RDEVICE macroinstructions in HCPRIO.

### **ASSIGN\_at IPL**

tells CP to assign the tape drive to the system when the device is being brought online.

### **NOASSIGN\_at IPL**

tells CP not to assign the tape drive when the device is being brought online.

### **DYNAMIC\_I/O**

tells CP to allow 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 will not let you make dynamic I/O changes on any individual device or devices.

### **NOTDYNAMIC\_I/O**

tells CP not to allow dynamic I/O changes on the specified device or devices.

### **INITIALIZED\_at IPL**

tells CP to initialize the specified device or devices during IPL.

### **NOTINITIALIZED\_at IPL**

tells CP not to initialize the specified device or devices during IPL.

**Note:** The last two operands are functionally equivalent to the ONLINE\_AT\_IPL and OFFLINE\_AT\_IPL operands respectively.

### **OFFLINE\_at IPL**

tells CP not to initialize the specified device or devices at IPL.

**Note:** Due to the fact that Open Systems Adapter (OSA) devices may not be ready to come online until after system IPL has completed, this statement will not always work for OSA devices.

### **ONLINE\_at IPL**

tells CP to initialize the specified device or devices at IPL.

### **SCMeasured**

tells CP to collect subchannel measurement data for the subchannels of the specified device or devices.

### **NOTSCMeasured**

tells CP not to collect subchannel measurement data for the subchannels of the specified device or devices.

### **SENSEd**

tells CP to use the information returned from a sense ID request to determine the device class or type.

### **NOTSENSEd**

tells CP not to use the information returned from a sense ID request to determine the device class or type.

**SHAREd**

tells CP that the specified DASD device or devices will be shared between independent operating systems.

**NOTSHAREd**

tells CP that the specified DASD device or devices will not be shared with any other independent operating system.

**THROTtled**

tells CP to limit (throttle) the rate of I/O coming from the specified device or devices.

**NOTTHROTtled**

tells CP not to 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 will be controlled by HCD. In this case, the following DEVICES statements are ignored if they are specified:

- DEVICES ACCEPTED
- DEVICES NOTACCEPTED
- DEVICES DYNAMIN\_I/O
- DEVICES NOTDYNAMIC\_I/O
- DEVICES OFFLINE\_AT\_IPL
- DEVICES ONLINE\_AT\_IPL
- DEVICES SENSED
- DEVICES NOTSENSED
- DEVICES SHARED
- DEVICES NOTSHARED

Refer to “IODF Statement” on page 158 for more information.

2. If you do not specify the DEVICES statement in the system configuration file, the defaults that CP uses depend on what is specified on the SYSOPTS macroinstruction in the HCPSYS ASSEMBLE file.

- If the SYSOPTS setting is SENSE=YES (the default), CP will:
  - Accept all devices,
  - Allow dynamic changes on all devices (if you specified FEATURES ENABLE DYNAMIC\_I/O),
  - Initialize all devices,
  - Measure all devices,
  - Sense the characteristics of all devices,
  - Not share any DASD devices,
  - Not throttle any devices, and
  - Not assign any tape drives to CP at IPL.

For any devices that you have not explicitly defined with the RDEVICE statement in the system configuration file or with the RDEVICE macroinstruction in HCPRIO, CP will sense the device, if possible, and dynamically build an RDEV for it.

- If the SYSOPTS setting is SENSE=NO, CP will accept, initialize, measure, and bring online only those devices that you explicitly define with RDEVICE statements or macroinstructions. CP will not throttle or allow dynamic

## DEVICES

changes on any of these devices. CP will share DASD devices only if you specify the SHARED operand on the RDEVICE statement or macroinstruction for that DASD. CP ignores all other devices.

Also, if you specify the wrong device type on the RDEVICE statement or macroinstruction, CP will not bring the device online because of conflicting device information. That is, if you define a 3420 tape drive and the device is really a 3480 tape drive, CP will not bring the device online. If you specify the wrong device class, CP will not bring the device online. That is, if you define a tape drive and the device is really a DASD, CP will not bring the device online.

For more information about the RDEVICE statement, see page 188. For more information about the RDEVICE macroinstruction, see page 705.

3. To allow or prevent dynamic changes after IPL, use the SET DYNAMIC\_I/O command. For more information, see the *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 the *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 or macroinstruction. For more information about the SET RDEVICE command, see the *z/VM: CP Commands and Utilities Reference*. For more information about the RDEVICE statement or macroinstruction, see page 195 or page 745, respectively.
6. To define new devices after IPL, use the SET RDEVICE command. For more information, see the *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.

## Examples

1. To have CP:
  - Not accept a group of terminals, except for one,
  - Allow dynamic I/O changes on all devices, except for 1905 through 1943.
  - Initialize all devices, except 8E0 through 8EF,
  - Measure all devices, except DASD 2F04,

- Leave a group of disks offline or not initialized,
- Not sense a group of 3705 devices,
- Share a string of DASDs at 1100 through 114F,
- Throttle the I/O rate to a shared database at 460 through 48F, and
- Assign tape drive 181 to CP,

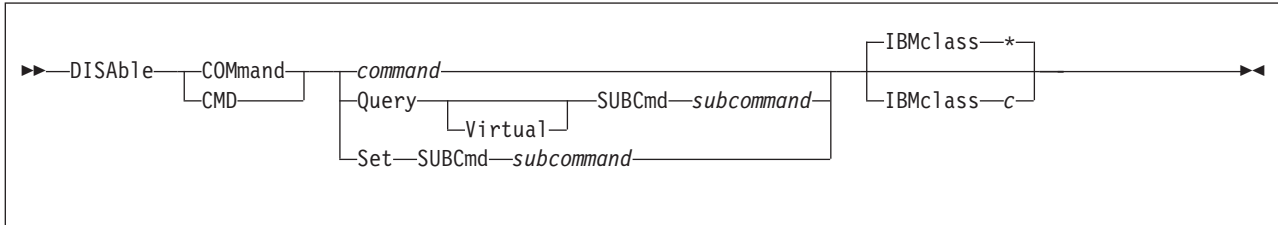
use the following DEVICES statement:

```

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 3705 */
        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_ip1 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 using the DISABLE COMMAND or CMD commands. For more information, see the *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 system storage.
- D** this is a command for system-wide control of spool files.
- E** this is a command to examine system 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 CP's system execution space, use the CPXUNLOAD command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
2. To load the command processing code into CP's system execution space, use the CPXLOAD statement (page 76) or command. For more information about the CPXLOAD command, see the *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 (page 84) or command. For more information about the DEFINE COMMAND or CMD command, see the *z/VM: CP Commands and Utilities Reference*.
4. To activate a CP command:
  - While defining it, use the ENABLE operand of the DEFINE COMMAND or CMD statement (page 84) or command.
  - After defining it, use the ENABLE COMMAND or CMD statement (page 126) or command.

For more information about the DEFINE or ENABLE COMMAND or CMD commands, see the *z/VM: CP Commands and Utilities Reference*.
5. To change the definition of an existing CP command, use the MODIFY COMMAND or CMD statement (page 165) or command. For more information about the MODIFY COMMAND or CMD command, see the *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 real 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 the *z/VM: CP Commands and Utilities Reference*.
8. For more information about enabling and disabling commands, see the *z/VM: CP Exit Customization* book.

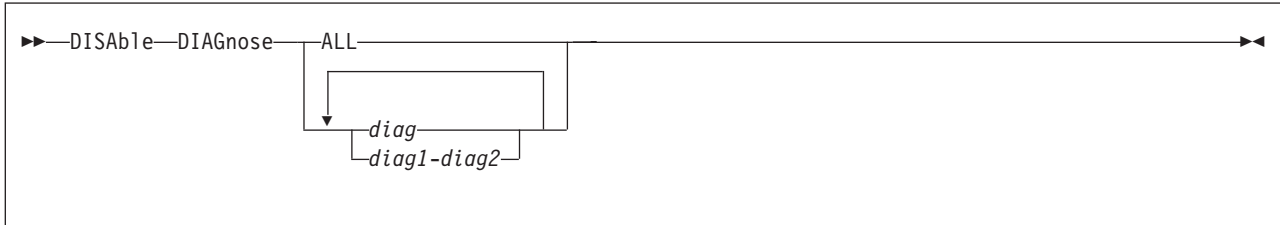
## Examples

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 the *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 CP's system execution space, use the `CPXLOAD` statement (page 76) or command. For more information about the `CPXLOAD` command, see the *z/VM: CP Commands and Utilities Reference*.
3. To define a new `DIAGNOSE` code, use the `DEFINE DIAGNOSE` statement (page 90) or command. For more information about the `DEFINE DIAGNOSE` command, see the *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 (page 128) or command. For more information about the ENABLE DIAGNOSE command, see the *z/VM: CP Commands and Utilities Reference*.
5. To change the definition of an existing DIAGNOSE code, use the MODIFY DIAGNOSE statement (page 169) or command. For more information about the MODIFY DIAGNOSE command, see the *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 the *z/VM: CP Commands and Utilities Reference*.
7. To display the real address of the CP DIAGNOSE code table block for a DIAGNOSE code after initialization, use the LOCATE DGNBK command. For more information, see the *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 CP's system execution space, use the CPXUNLOAD command. For more information, see the *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 the *z/VM: CP Exit Customization* book.

## Examples

1. To have CP disable DIAGNOSE code X'100', use the following:  

```
Disable Diagnose 100
```

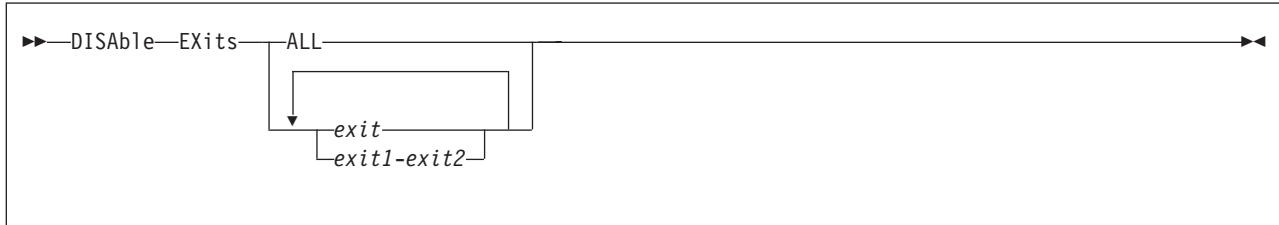
After initialization, any virtual machine that issues DIAGNOSE code X'100' receives a program specification exception in that virtual machine.
2. To have CP disable all DIAGNOSE codes on your system, use the following:  

```
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-iec if4-1fc
```

## 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 using the DISABLE EXITS command. For more information, see the *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 CP's system execution space, use the CPXLOAD statement (page 76) or command. For more information about the CPXLOAD command, see the *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 (page 60) or command. For more information about the ASSOCIATE EXIT command, see the *z/VM: CP Commands and Utilities Reference*.
3. To activate an exit point after defining it, use the ENABLE EXITS statement (page 130) or command. For more information about the ENABLE EXITS command, see the *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 the *z/VM: CP Commands and Utilities Reference*.

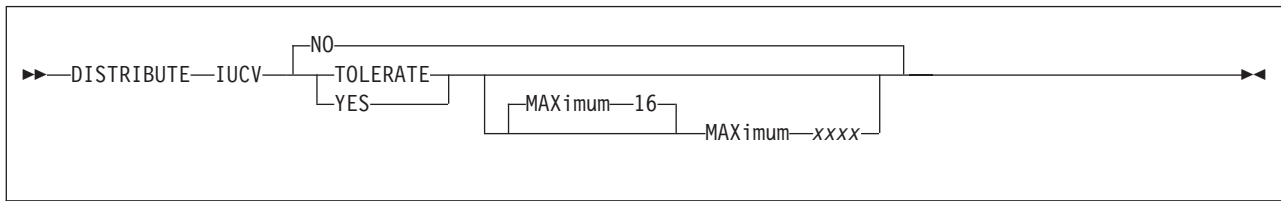
5. To display the real address of the CP exit block for a specific exit point, use the LOCATE XITBK command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
6. To display the real address of the CP indirect call locator block for a specific exit point, use the LOCATE ICLBK command. For more information, see the *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 (page 172) or command. For more information about the MODIFY EXIT command, see the *z/VM: CP Commands and Utilities Reference*.
8. To remove the exit point code from CP's system execution space, use the CPXUNLOAD command. For more information, see the *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 the *z/VM: CP Exit Customization* book.

## 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

### Format



### Purpose

Use the DISTRIBUTE statement to specify distribution features for the local system.

### Internal Representation

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.

### Parameters

#### 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.

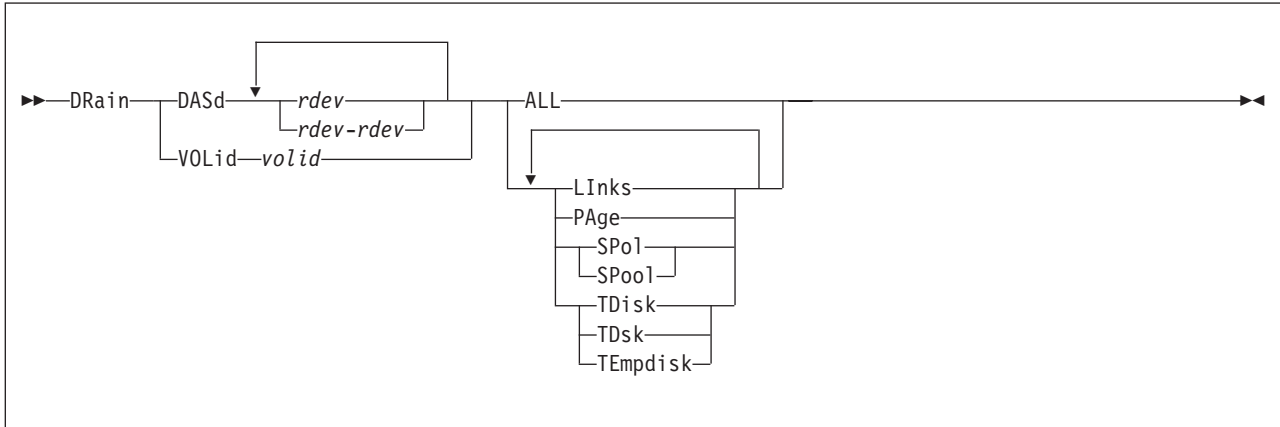
**Note:** It is recommended that all nodes in a CS collection have the same value specified for MAXIMUM because the maximum value is enforced only by the originating node.

### Usage

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. 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 16M 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 (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** *valid*

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 about the DRAIN (Disk), START (Disk), or DETACH commands, see the *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 about the CP ATTACH and DETACH commands, see the *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 about the START (Disk) or QUERY DASD DRAINING commands, see the *z/VM: CP Commands and Utilities Reference*. For more information about the START (Disk) statement, see page 220.

## Examples

Use the DRAIN and START statements shown in Figure 3 on page 120 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 (Disk)

```
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 3. 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:

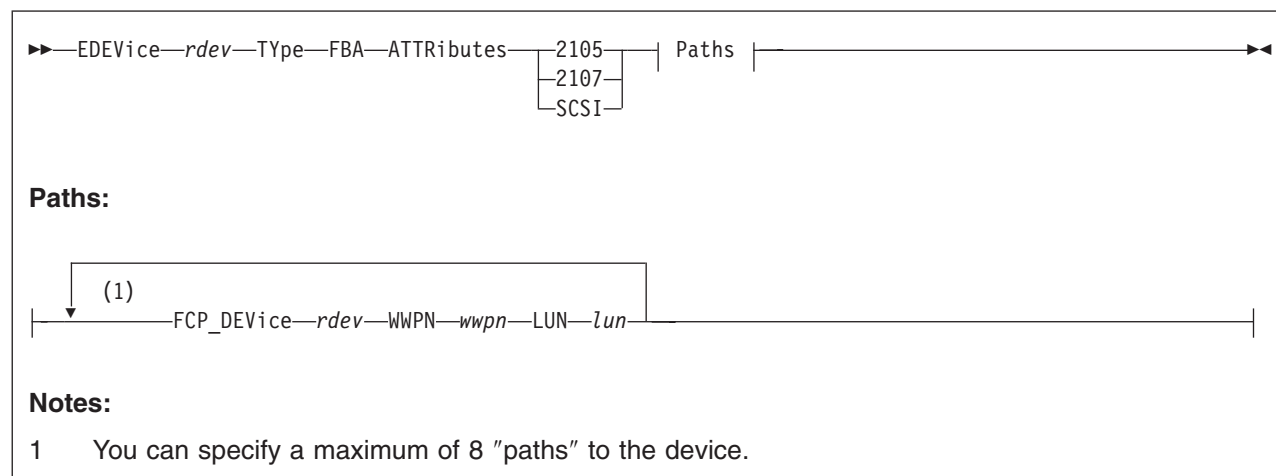
```
Vol-ID Rdev  Draining: PAge  LInks  SPool  TDisk
      ⋮
SYSPG1 0777          No    Yes   Yes   Yes
      ⋮
```

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



## Purpose

Use the EDEVICE statement to define an emulated device that represents a real SCSI device.

## 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 ESS 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

**rdev**  
specifies the device number that is associated with the emulated device. The *rdev* must be a 1-digit to 4-digit hexadecimal number between X'0000' and X'FFFF'.

**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.

## EDEVICE

### 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.

### 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.

### 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-digit to 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 should 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 may 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.**

## 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 path is a duplicate of an already existing path for the emulated device.
  - 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.

## Examples

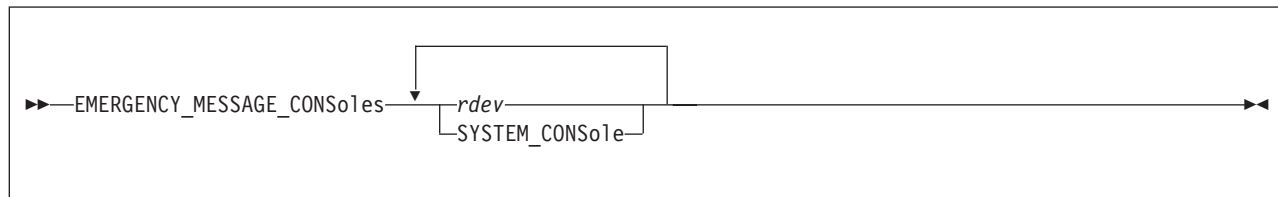
1. To define an emulated device 8181 for a 2105 SCSI device with one path, enter the following:
 

```
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
```

---

## 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. (See *z/VM: General Information* for a complete list of supported 3270-type displays.)

#### **SYSTEM\_CONSOLE**

adds the system console to the list of consoles that are sent emergency messages.

### 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. Refer to "IODF Statement" on page 158 for more information.
2. 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 or the `RIOGEN` macroinstruction in `HCPRIO`, 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 about the OPERATOR\_CONSOLES statement, see page 180. For more information about the RIOGEN macroinstruction, see page 766.

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.

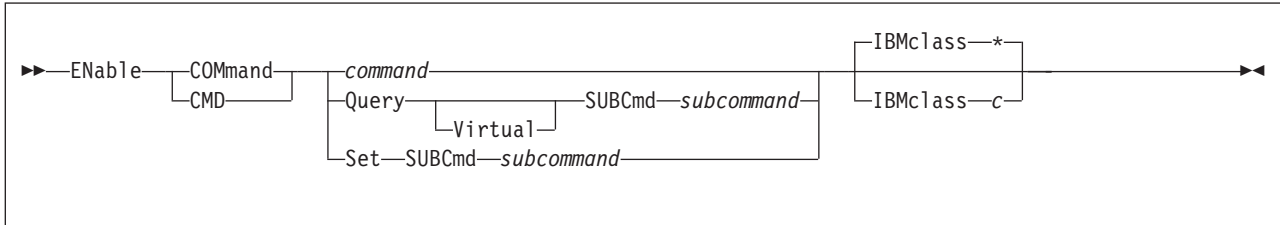
There will be a total of 101 emergency message consoles if the maximum number of 100 is specified on the EMERGENCY\_MESSAGE\_CONSOLES statement and the additional console address is added to the list.

### 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 the *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 system storage.
- D** this is a command for system-wide control of spool files.
- E** this is a command to examine system 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 CP's system execution space, use the CPXLOAD statement (page 76) or command. For more information about the CPXLOAD command, see the *z/VM: CP Commands and Utilities Reference*.
2. To activate a CP command while defining it, use the ENABLE operand of the DEFINE COMMAND or CMD statement (page 84) or command. For more information about the DEFINE COMMAND or CMD command, see the *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 (page 84) or command.
  - After defining it, use the DISABLE COMMAND or CMD statement (page 110) or command.

For more information about the DEFINE or DISABLE COMMAND or CMD commands, see the *z/VM: CP Commands and Utilities Reference*.
4. To change the definition of an existing CP command, use the MODIFY COMMAND or CMD statement (page 165) or command. For more information about the MODIFY COMMAND or CMD command, see the *z/VM: CP Commands and Utilities Reference*.
5. To display the real 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 the *z/VM: CP Commands and Utilities Reference*.
6. To remove the command processing code from CP's system execution space, use the CPXUNLOAD command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
7. For more information about enabling and disabling commands, see the *z/VM: CP Exit Customization* book.

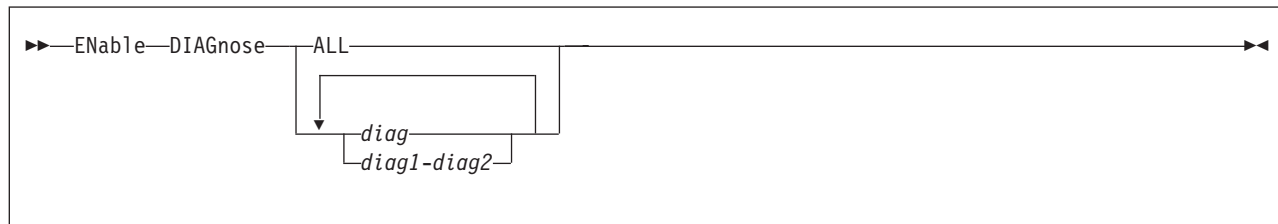
## Examples

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 the *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 (page 90) or command. For more information about the DEFINE DIAGNOSE command, see the *z/VM: CP Commands and Utilities Reference*.
2. To load the DIAGNOSE processing code into CP's system execution space, use the CPXLOAD statement (page 76) or command. For more information about the CPXLOAD command, see the *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.



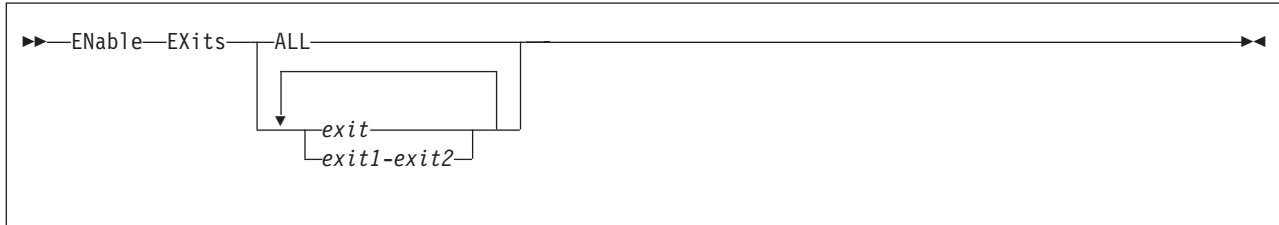
4. To change the definition of an existing DIAGNOSE code, use the MODIFY DIAGNOSE statement (page 169) or command. For more information about the MODIFY DIAGNOSE command, see the *z/VM: CP Commands and Utilities Reference*.
5. To display information about a DIAGNOSE code (status, entry point name, and privilege class) after initialization, use the QUERY DIAGNOSE command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
6. To display the real address of the CP DIAGNOSE code table block for a DIAGNOSE code, use the LOCATE DGNBK command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
7. To deactivate a DIAGNOSE code:
  - While defining it, use the DISABLE operand of the DEFINE DIAGNOSE statement (page 90) or command.
  - After defining it, use the DISABLE DIAGNOSE statement (page 112) or command.

For more information about the DEFINE or DISABLE DIAGNOSE commands, see the *z/VM: CP Commands and Utilities Reference*.
8. To remove the DIAGNOSE processing code from CP's system execution space, use the CPXUNLOAD command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
9. 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.
10. For more information about user-defined DIAGNOSE codes, see the *z/VM: CP Exit Customization* book.

## 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 permit CP to call the entry points and external symbols associated with an exit point after initialization using the ENABLE EXITS command. For more information, see the *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 CP's system execution space, use the CPXLOAD statement (page 76) or command. For more information about the CPXLOAD command, see the *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 (page 60) or command. (For more information about the ASSOCIATE EXIT command, see the *z/VM: CP Commands and Utilities Reference*.) You can also use the ASSOCIATE EXIT statement to change the entry points and external symbols 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.

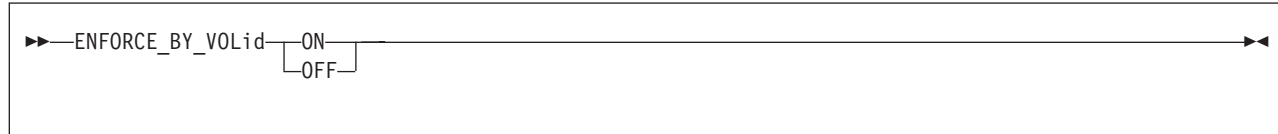
4. To display whether there are any unknown entry points or external symbols associated with an exit point after initialization, use the QUERY UNRESOLVED command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
5. To display status and usage statistics information about a specific exit point after initialization, use the QUERY EXITS command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
6. To display the real address of the CP exit block for a specific exit point, use the LOCATE XITBK command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
7. To display the real address of the CP indirect call locator block for a specific exit point, use the LOCATE ICLBK command. For more information, see the *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 (page 172) or command. For more information about the MODIFY EXIT command, see the *z/VM: CP Commands and Utilities Reference*.
9. To stop CP from calling all the entry points and external symbols associated with one or more exit points after defining those exit points, use the DISABLE EXITS statement (page 114) or command. For more information about the DISABLE EXITS command, see the *z/VM: CP Commands and Utilities Reference*.
10. To remove the exit point code from CP's system execution space, use the CPXUNLOAD command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
11. For more information about user-defined exit points, see the *z/VM: CP Exit Customization* book.

## 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

---

## 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.

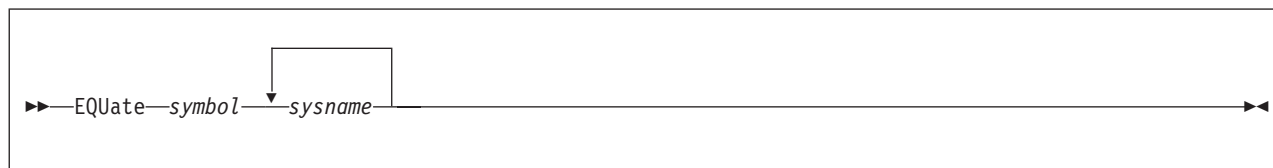
#### OFF

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

## 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-character 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-character 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. 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: ...
```

## EQUATE

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.

2. You can also use the *systemid* on the SYSTEM\_IDENTIFIER statement (page 226) as a record qualifier. Just be sure to define the *systemid* in a SYSTEM\_IDENTIFIER statement before you try to use it as a nickname.

## 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 cleveland,
               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:

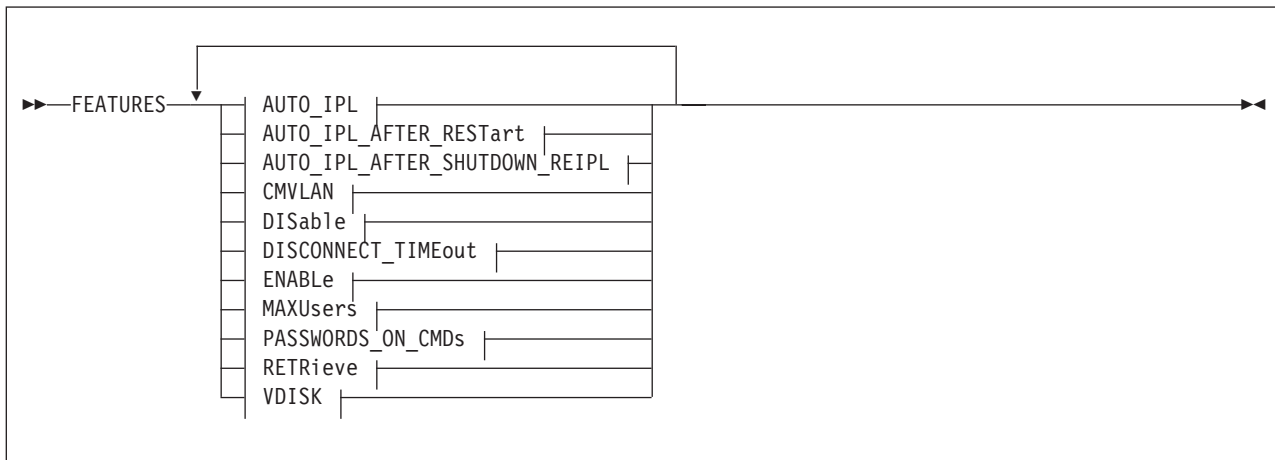
```
CPXLoad tttsyn text a          /* Load the module          */
      Temporary,             /* ... in case we must unload it later */
      NoControl,            /* ... to unload without more checking */
      Nodelay,              /* ... load immediately from parmdisk */
      Lock                   /* ... 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 */
```

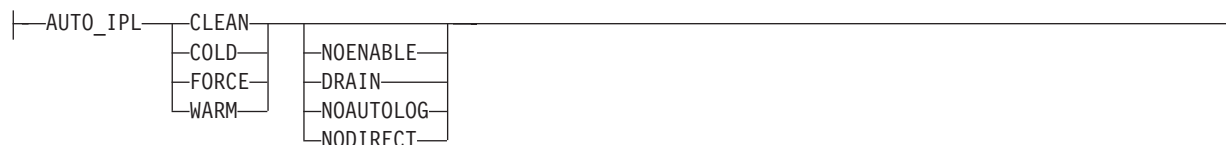
```
External_Syntax EPname tttsyntax
```

In this example, the module TTTSYN contains your syntax tree, which starts at label TTTSYNTAX. This example shows how you would get CP to load the module and make CP consider it additional syntax during initialization.

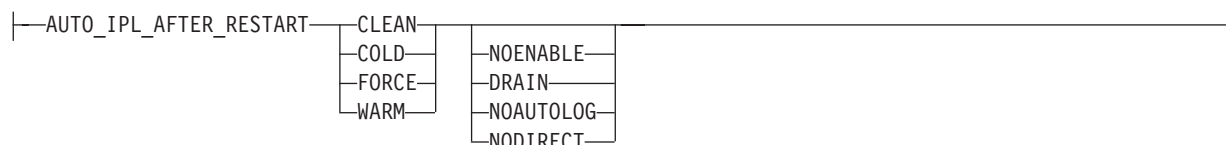
## FEATURES Statement



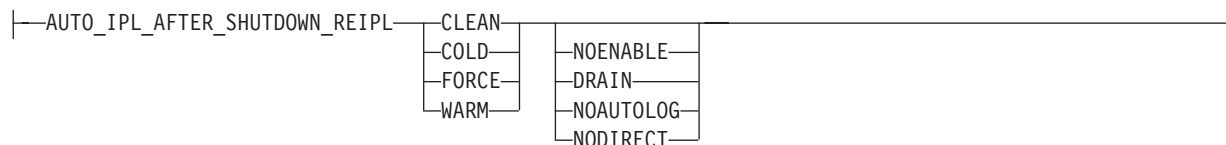
### AUTO\_IPL:



### AUTO\_IPL\_AFTER\_RESTART:

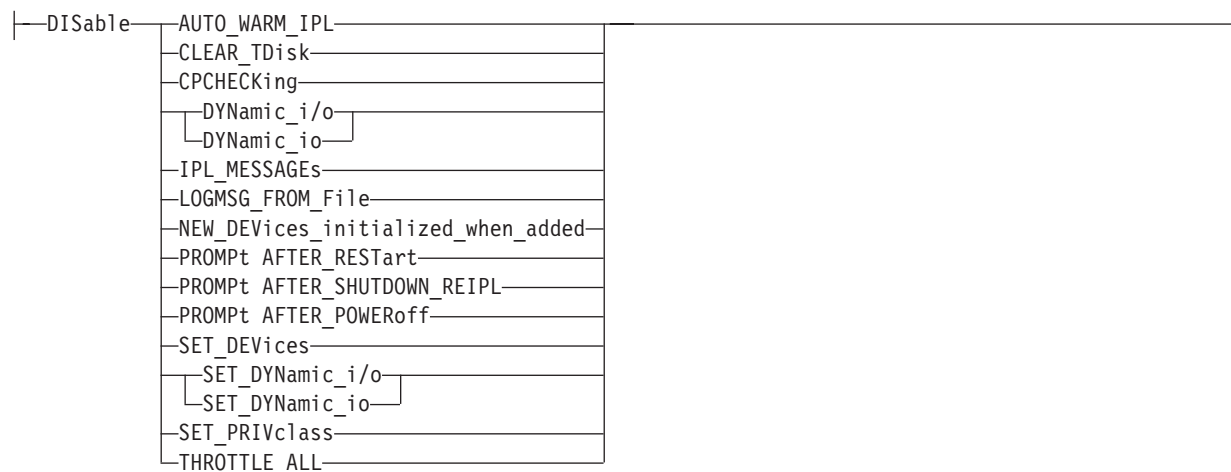


### AUTO\_IPL\_AFTER\_SHUTDOWN\_REIPL:





**DISable:**

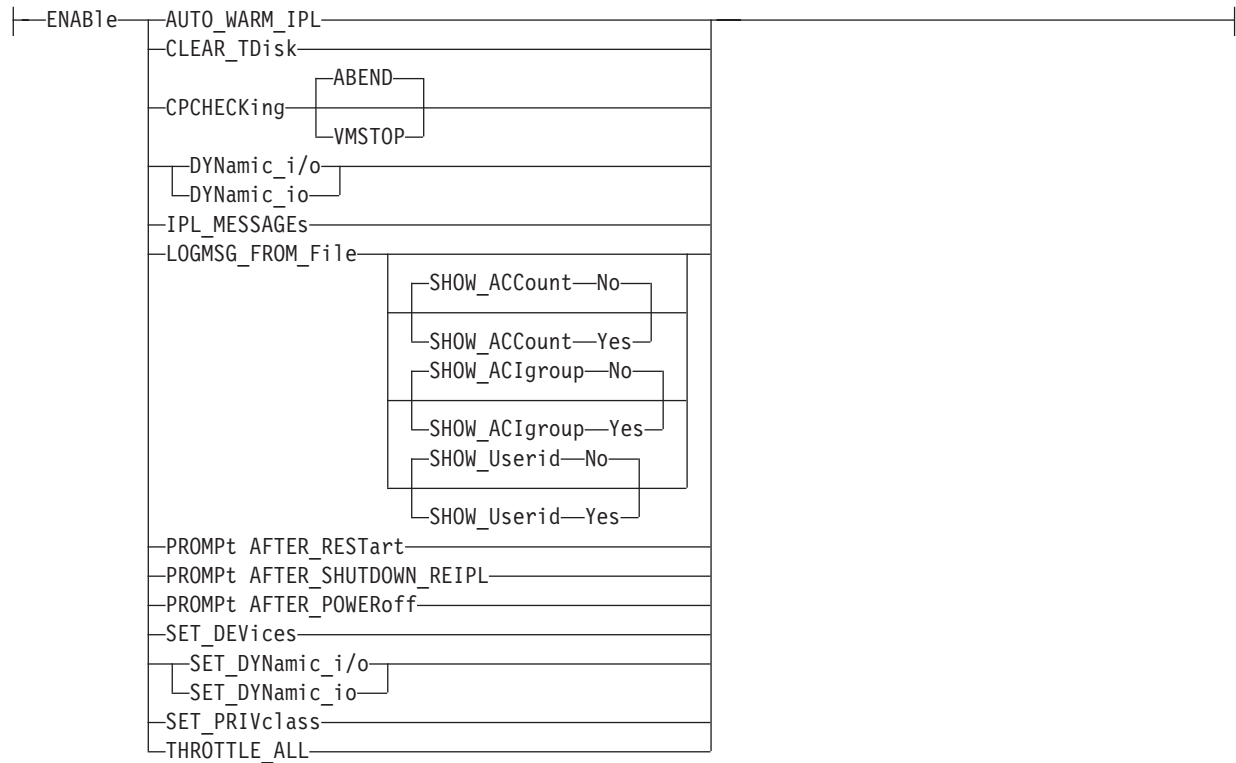


**DISCONNECT\_TIMEout:**

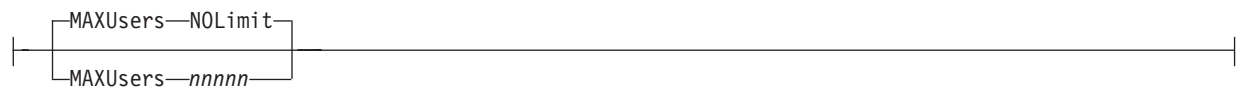


## FEATURES

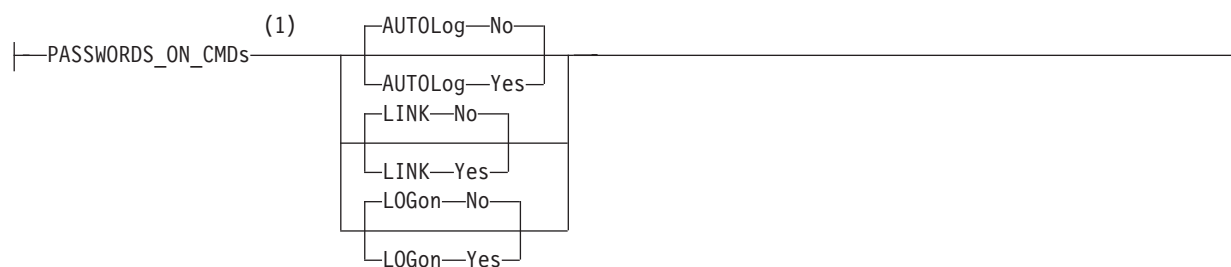
### ENABLE:



### MAXUsers:



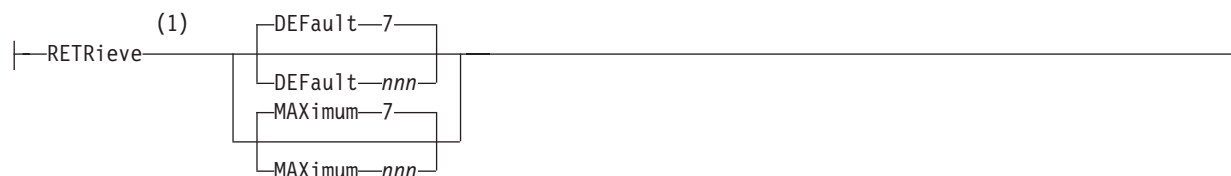
**PASSWORDS\_ON\_CMDs:**



**Notes:**

- 1 You must specify at least one of the operands.

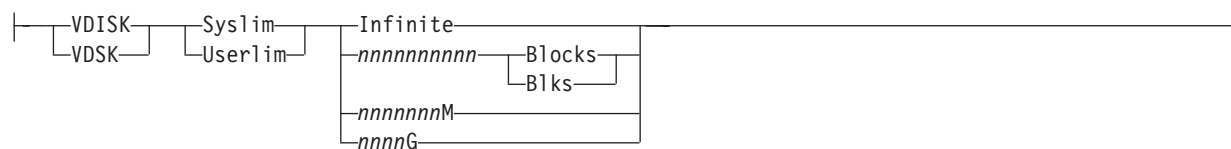
**RETRieve:**



**Notes:**

- 1 You must specify at least one of the operands.

**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.

## FEATURES

### 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. Note that initially, each option is disabled until enabled using the ENABLE operand.

#### **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.

**CPCHECKing**

tells CP that internal CP checking is not to be executed.

**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 on the CP SET LOGMSG command, see the *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 about the CP VARY (Real Device) command, see the *z/VM: CP Commands and Utilities Reference*.

**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.

**PROMPt AFTER\_POWERoff**

tells CP not to force a prompt to the operator when CP detects that this IPL is during a power on after a SHUTDOWN POWEROFF.

**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 about the CP SET DEVICES command, see the *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 about the CP SET DYNAMIC\_I/O command, see the *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

## FEATURES

change the privilege classes of users logged on to the system. For more information about the CP SET PRIVCLASS command, see the *z/VM: CP Commands and Utilities Reference*.

### **THROTTLE\_ALL**

tells CP to allow throttling of ALL devices on the system except CP OWNED DASD.

### **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. Note that initially, each option is disabled until it is enabled.

### **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.

### **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.

### **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 the *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).

**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.

**PROMPt AFTER\_POWERoff**

tells CP to force a prompt to the operator when CP detects that this IPL is during a power on after a SHUTDOWN POWEROFF, so that REIPL can be stopped or the type of start desired (WARM, FORCE, etc.) can be specified.

**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 about the CP SET DEVICES command, see the *z/VM: CP Commands and Utilities Reference*.

## FEATURES

### **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 about the CP SET DYNAMIC\_I/O command, see the *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 about the CP SET PRIVCLASS command, see the *z/VM: CP Commands and Utilities Reference*.

### **THROTTLE\_ALL**

tells CP to allow throttling of ALL devices on the system including CP OWNED DASD.

### **MAXusers NOLimit**

(the default) tells CP that an unlimited number of users can log on.

### **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 to 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 about the CP AUTOLOG, XAUTOLOG, LINK, and LOGON commands, see the *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 about the CP SET PF $nn$  RETRIEVE command, see the *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 system storage available for allocation as virtual disks in storage. If an installation default is not defined, CP uses a built-in default.

You can supercede 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 about the CP SET and QUERY VDISK commands, see the *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 system 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.

## FEATURES

### **Infinite**

indicates that there is no limit. All available system storage may be allocated to virtual disks in storage.

### *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.

#### *nnnnnnnn***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. 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

Refer to “IODF Statement” on page 158 for more information.

2. 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 directory entries. For more information about the CP SET PRIVCLASS command, see the *z/VM: CP Commands and Utilities Reference*.
3. 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 about the CP QUERY and SET RETRIEVE commands, see the *z/VM: CP Commands and Utilities Reference*.

4. 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.
5. 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 about the Stand-Alone Program Loader (SAPL) and the IPL parameters, see page 37.
6. 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 about the `SET NEW_DEVICES` command, see the *z/VM: CP Commands and Utilities Reference*.
7. 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

8. 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 about the `SET DEVICES` command, see the *z/VM: CP Commands and Utilities Reference*.
9. 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.
10. 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).

## Examples

1. To have CP:
  - Enable all log message support
  - Allow end users and the operator to issue the `SET PRIVCLASS` command

## FEATURES

- 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 */

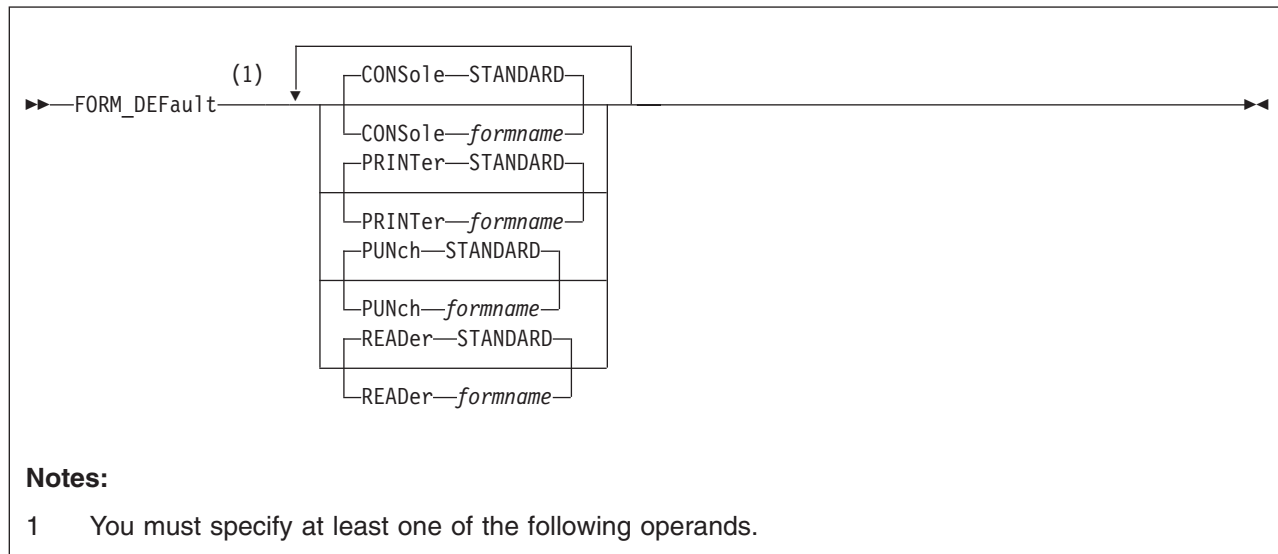
    Passwords_On_Cmds,

        AutoLog No,        /* Prompt user for password on AUTOLOG command */
        Link Yes,          /* Do not prompt for password on LINK command */
```

## FEATURES

```
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



### 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.

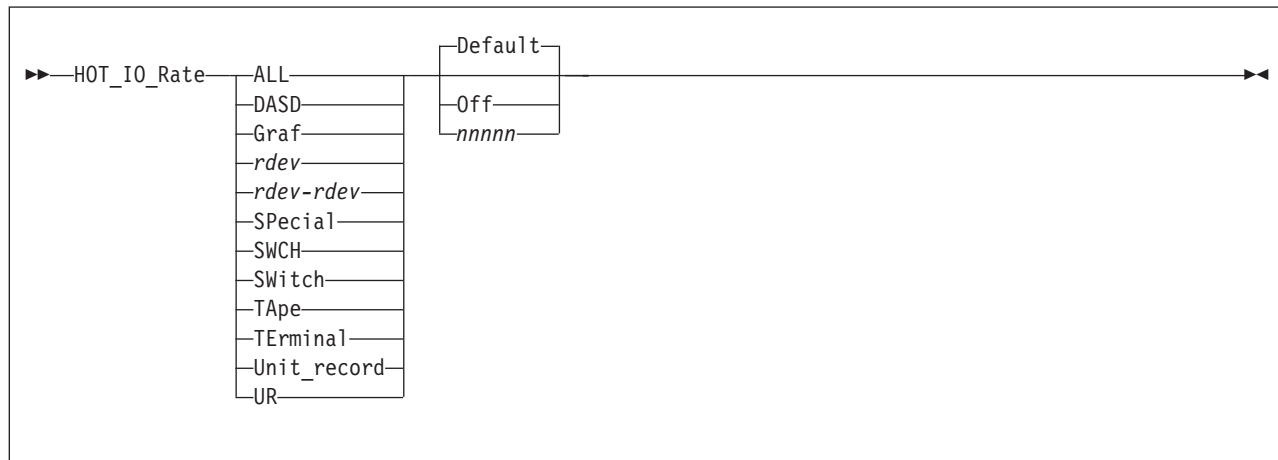
### Examples

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

```
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.

**TErминаl**

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. Refer to "IODF Statement" on page 158 for more information.
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.

## HOT\_IO\_RATE

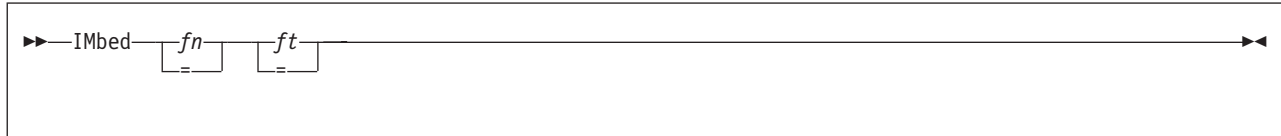
4. If you need to change the hot I/O rate after IPL, use the CP SET HOTIO command. To check the current hot I/O rates, use the CP QUERY HOTIO command. For more information about the CP SET HOTIO and QUERY HOTIO commands, see the *z/VM: CP Commands and Utilities Reference*.

## 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

- 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.
- An imbed will not succeed if the `-SYSTEM-` variable is included and no system identifier exists to replace it. This situation arises when you do not specify a `SYSID` macroinstruction in `HCPSYS` and you do not specify `SYSTEM_IDENTIFIER` and `SYSTEM_IDENTIFIER_DEFAULT` statements in the system configuration file before the IMBED statement.
- 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

- 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 */
```

## IMBED

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 (page 133) 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.

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

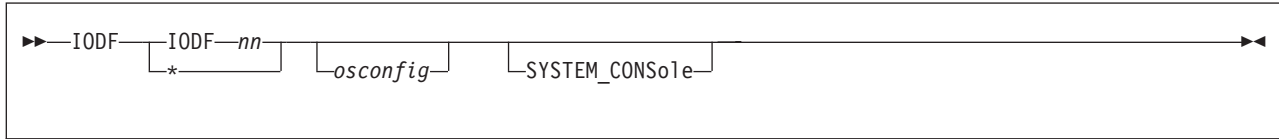
1. Lowering the initialization MITIME value below the default value of 15 seconds may cause certain devices to not come online during IPL due to timeout errors. These devices may be able to be brought online with the VARY ON command after the IPL has completed.
2. When using the INIT\_MITIME statement 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*nn* 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*nn*

specifies the filename of the production IODF to be used. The filename must be specified in the form IODF*nn* and the filetype that will be used with that filename is PRODIODF. This file must be located on the system parameter disk (SYSPARM) 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*nn* and the filetype that will be used with that filename is PRODIODF. This file must be located on the system parameter disk (SYSPARM) at system IPL time.

#### *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 system console can serve as the 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

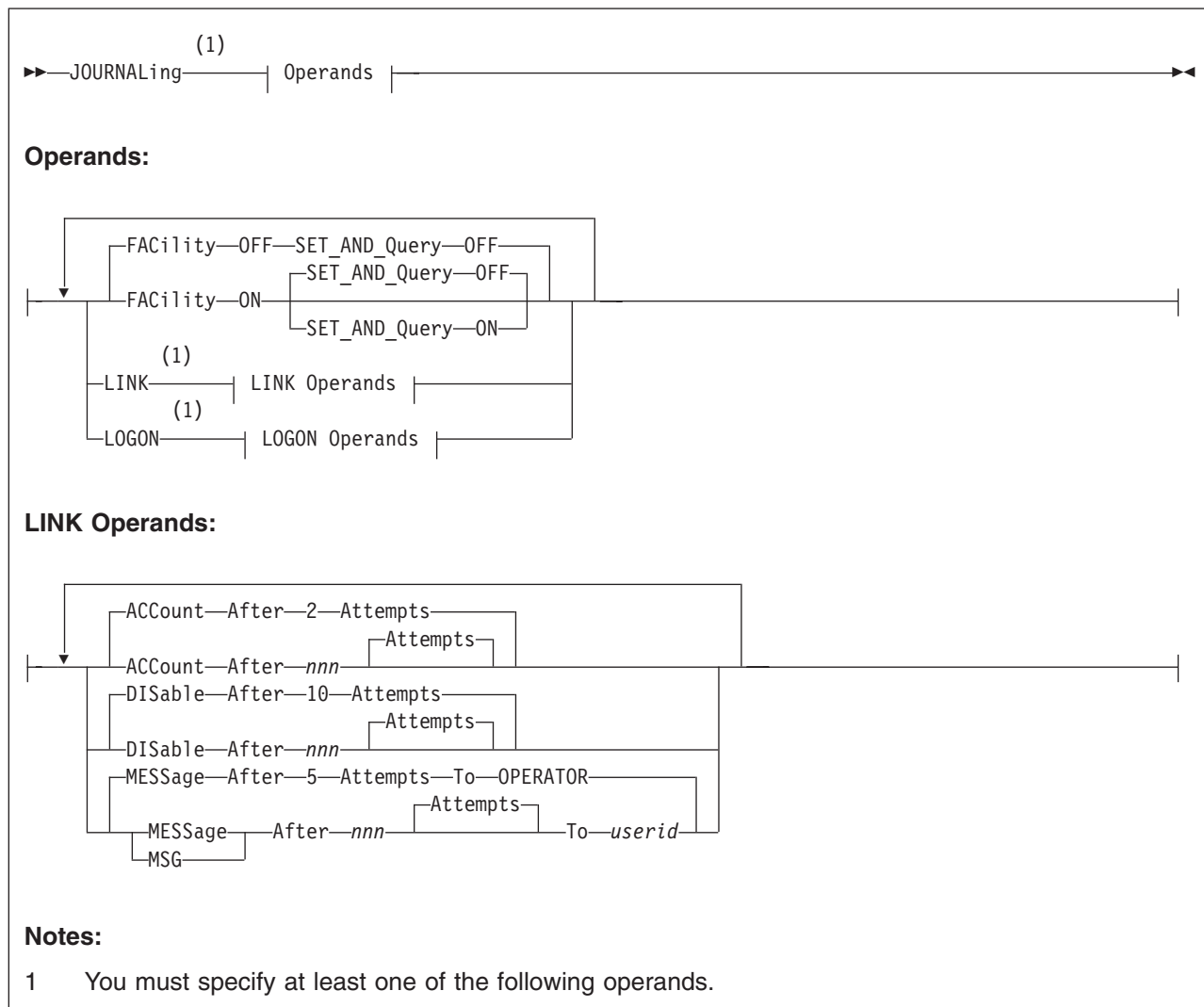
1. To have HCD control only the hardware I/O configuration, as specified in the IODF01 PRODIODF production IODF file, use the following IODF statement in the system configuration file:
 

```
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:
 

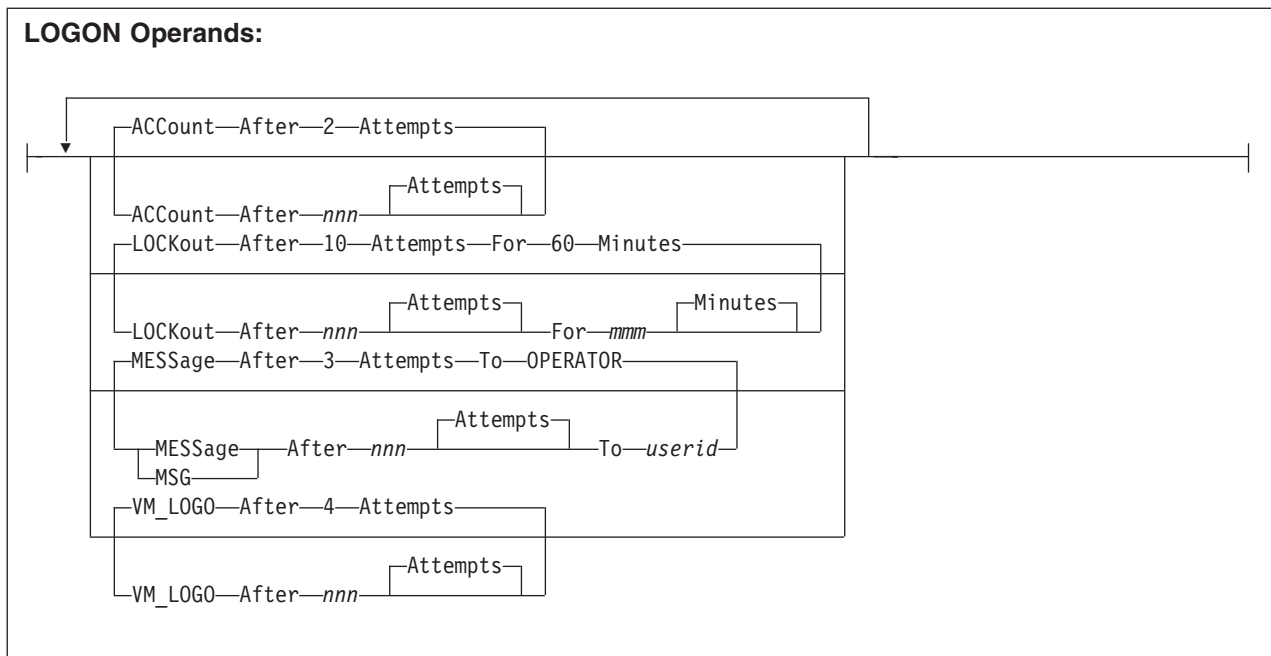
```
IODF IODF02 CONFIG04
```
3. To have HCD control both the hardware and software I/O configuration, as specified in the IODF02 PRODIODF file (which contains the CONFIG04 operating system configuration), and to have the system console as a possible operator console, use the following IODF statement in the system configuration file:
 

```
IODF IODF02 CONFIG04 SYSTEM_CONSOLE
```

## JOURNALING Statement







## 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.

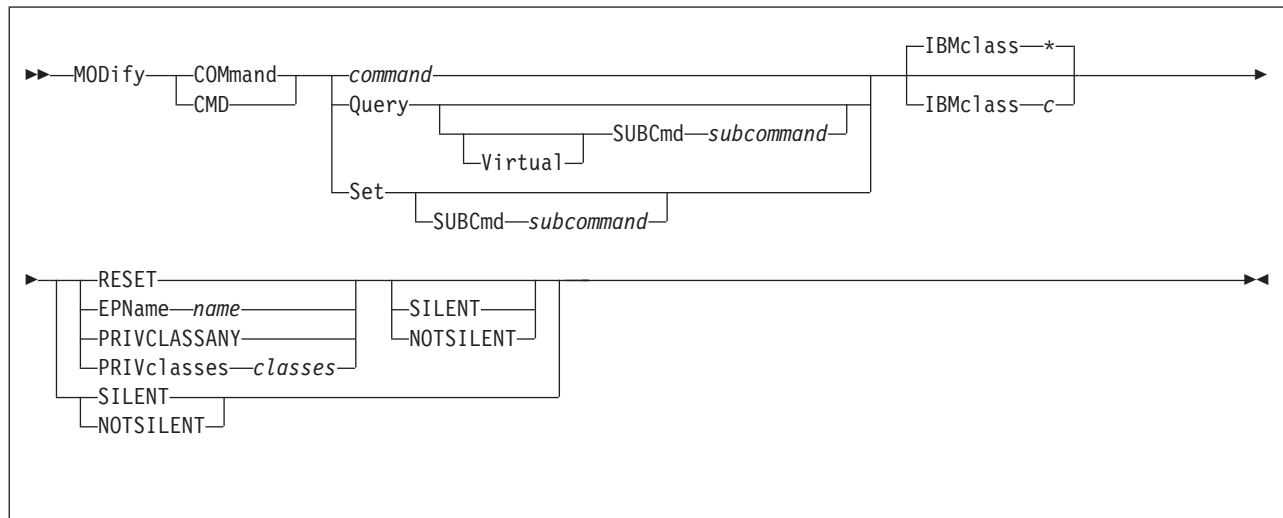
2. CP will not read a logo configuration file if the -SYSTEM- variable is included and no system identifier exists to replace it. This situation arises when you do not specify a SYSID macroinstruction in HCPSYS and you do not specify SYSTEM\_IDENTIFIER and SYSTEM\_IDENTIFIER\_DEFAULT statements in the 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.

**Note:** The OVERRIDE utility can provide some of the same functions. However, you should use either the OVERRIDE method or the MODIFY statements. If any MODIFY statement is detected at IPL time, the UCR (User Class Restructure) OVERRIDE changes will be bypassed. IBM recommends using the new MODIFY statements.

You can also redefine an existing CP command after initialization using the MODIFY COMMAND or MODIFY CMD commands. For more information, see the *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

## MODIFY COMMAND / CMD

variable *subcommand* is a 1-character to 12-character alphanumeric string. For more information about specifying generic subcommand names, see Usage Note 1.

### 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.

### **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 system storage.
- D** this is a command for system-wide control of spool files.
- E** this is a command to examine system 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 the *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 and those subcommands share common characters, you can use a generic subcommand name rather than specifying multiple MODIFY COMMAND or CMD statements (or commands). A generic subcommand name is a 1-character 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 (page 84) or command. For more information about the DEFINE COMMAND or CMD commands, see the *z/VM: CP Commands and Utilities Reference*.
3. To define a new alias for an existing CP command, use the DEFINE ALIAS statement (page 81) or command. For more information about the DEFINE ALIAS command, see the *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 CP's system execution space, use the CPXLOAD statement (page 76) or command. For more information about the CPXLOAD command, see the *z/VM: CP Commands and Utilities Reference*.
6. To activate a CP command:
  - While defining it, use the ENABLE operand of the DEFINE COMMAND or CMD statement (page 84) or command.
  - After defining it, use the ENABLE COMMAND or CMD statement (page 126) or command.

For more information about the DEFINE or ENABLE COMMAND or CMD commands, see the *z/VM: CP Commands and Utilities Reference*.
7. To display the real 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 the *z/VM: CP Commands and Utilities Reference*.
8. To deactivate a CP command:
  - While defining it, use the DISABLE operand of the DEFINE COMMAND or CMD statement (page 84) or command.
  - After defining it, use the DISABLE COMMAND or CMD statement (page 110) or command.

## MODIFY COMMAND / CMD

For more information about the DEFINE or DISABLE COMMAND or CMD commands, see the *z/VM: CP Commands and Utilities Reference*.

9. To remove the command processing code from CP's system execution space, use the CPXUNLOAD command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
10. For more information about user-defined commands, see the *z/VM: CP Exit Customization* book.

## 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:

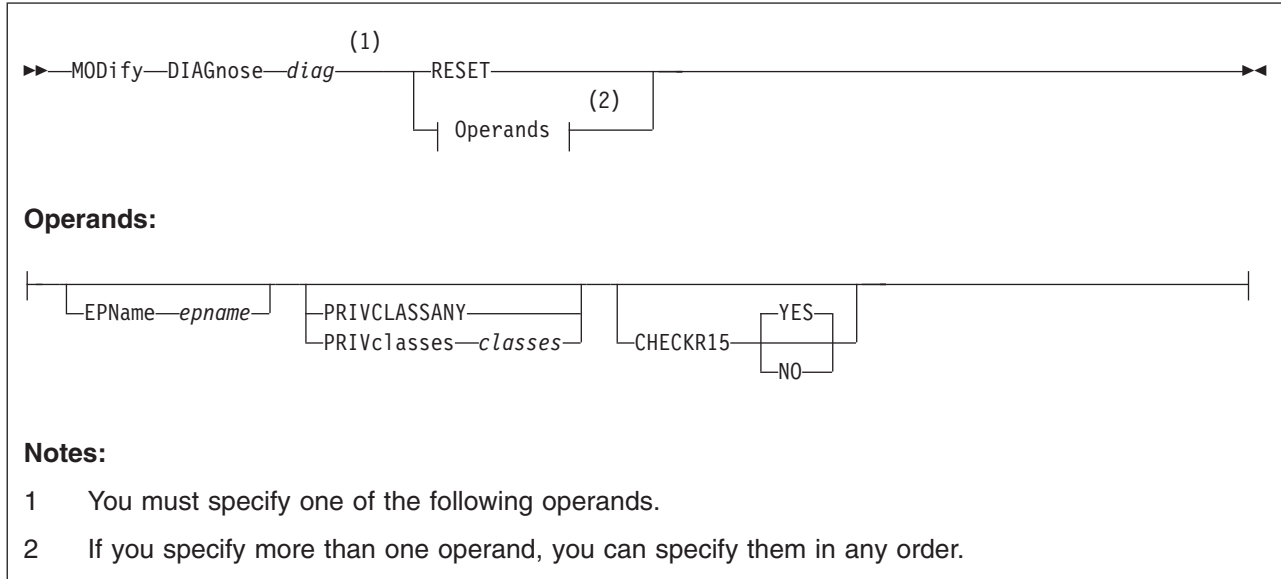
```
Modify Command Set SubCmd privclass, /* Make sure only a few can */
      IBMclass 0,                    /* issue SET PRIVCLASS      */
      PrivClasses gz
```

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



### Purpose

Use the MODIFY DIAGNOSE statement to redefine an existing DIAGNOSE code on the system during initialization.

**Note:** The OVERRIDE utility can provide some of the same functions. However, you should use either the OVERRIDE method or the MODIFY statements. If any MODIFY statement is detected at IPL time, the UCR (User Class Restructure) OVERRIDE changes will be bypassed. IBM recommends using the new MODIFY statements.

You can also redefine an existing DIAGNOSE code after initialization using the MODIFY DIAGNOSE command. For more information, see the *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.

## MODIFY DIAGNOSE

### **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 (page 90) or command. For more information about the DEFINE DIAGNOSE command, see the *z/VM: CP Commands and Utilities Reference*.
2. To load the DIAGNOSE processing code into CP's system execution space, use the CPXLOAD statement (page 76) or command. For more information about the CPXLOAD command, see the *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 new DIAGNOSE code:
  - While defining it, use the ENABLE operand of the DEFINE DIAGNOSE statement (page 90) or command.
  - After defining it, use the ENABLE DIAGNOSE statement (page 128) or command.

For more information about the DEFINE or ENABLE DIAGNOSE commands, see the *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 the *z/VM: CP Commands and Utilities Reference*.
6. To display the real address of the CP DIAGNOSE code table block for a DIAGNOSE code, use the LOCATE DGNBK command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
7. To deactivate a DIAGNOSE code:

- While defining it, use the DISABLE operand of the DEFINE DIAGNOSE statement (page 90) or command.
- After defining it, use the DISABLE DIAGNOSE statement (page 112) or command.

For more information about the DEFINE or DISABLE DIAGNOSE commands, see the *z/VM: CP Commands and Utilities Reference*.

- To remove the DIAGNOSE processing code from CP's system execution space, use the CPXUNLOAD command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
- 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.
- 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.

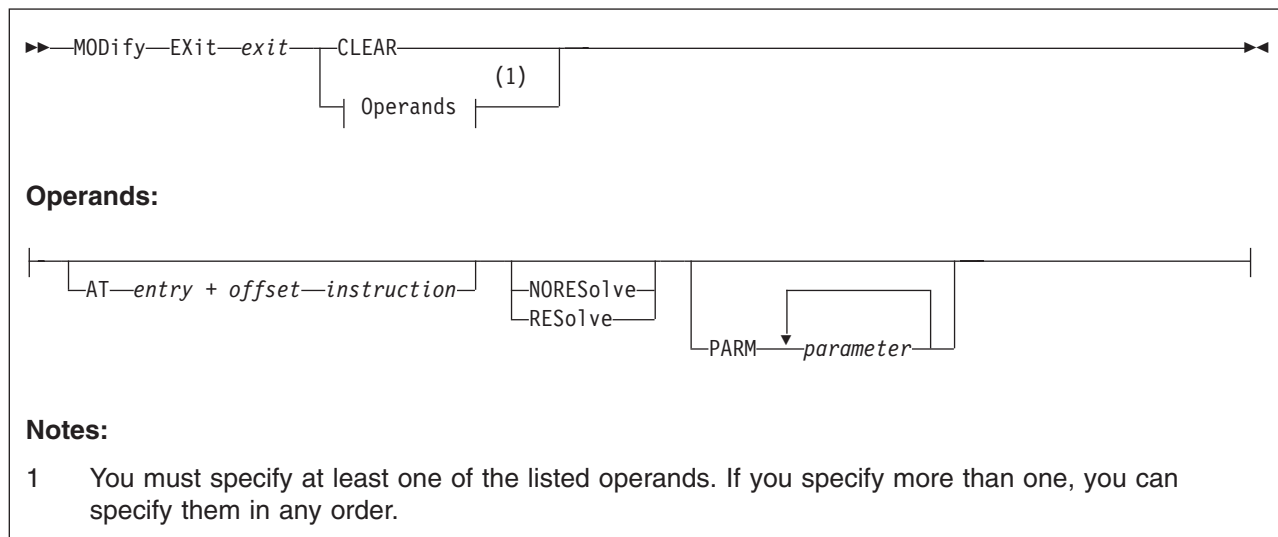
If a return code is invalid (negative, not a multiple of 4 or too big ( RC > 24 )), then a soft abend will occur.
- For more information about user-defined DIAGNOSE codes, see the *z/VM: CP Exit Customization book*.

## Examples

- 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



### 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 using the MODIFY EXIT command. For more information, see the *z/VM: CP Commands and Utilities Reference* book.

### 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><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 define a new dynamic exit point, use the DEFINE EXIT statement (page 94) or command. For information about the DEFINE EXIT command, see the *z/VM: CP Commands and Utilities Reference*.
2. To load the exit processing code into CP's system execution space, use the CPXLOAD statement (page 76) or command. For information about the CPXLOAD command, see the *z/VM: CP Commands and Utilities Reference*.
3. To activate a new exit point, use the ENABLE EXITS statement (page 130) or command. For information about the ENABLE EXITS command, see the *z/VM: CP Commands and Utilities Reference*.
4. To display information about an exit point (status, entry point name, and parameters), use the QUERY EXITS command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
5. To deactivate an exit point, use the DISABLE EXITS statement (page 114) or command. For information about the DISABLE EXITS command, see the *z/VM: CP Commands and Utilities Reference*.
6. To remove the exit processing code from CP's system execution space, use the CPXUNLOAD command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
7. Exit numbers are allocated as follows:
  - X'0000' to X'7FFF' are reserved for IBM use.

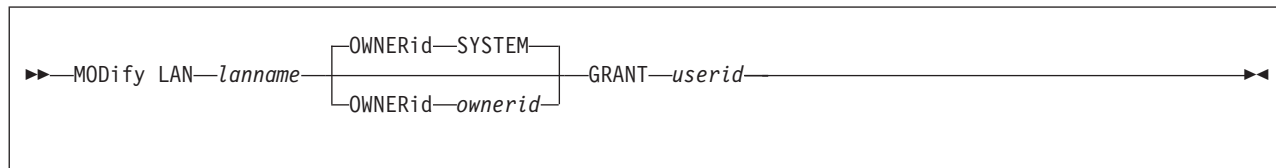
## MODIFY EXIT

- X'8000' to X'FFFF' 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 the *z/VM: CP Exit Customization* book. 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 exits, see the *z/VM: CP Exit Customization* book.

## 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 VM Guest LAN segment during initialization.

### Operands

#### **lanname**

is the name of the VM LAN segment 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 user *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.

### Usage Notes

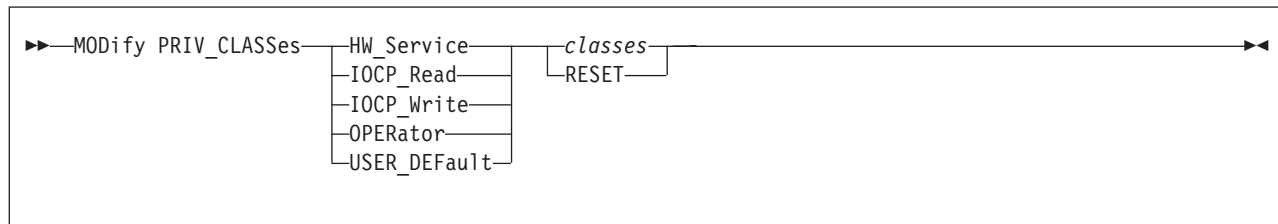
1. If the VM LAN is defined with the UNRESTRICTED attribute, any user can connect a virtual adapter to that LAN. The CP access list has no meaning for an UNRESTRICTED LAN. This will be overridden if an External Security Manager (ESM) is in control of the Virtual Switch.
2. When a RESTRICTED LAN is defined, 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 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.

**Note:** The OVERRIDE utility can provide some of the same functions. However, you should use either the OVERRIDE method or the MODIFY statements. If any MODIFY statement is detected at IPL time, the UCR (User Class Restructure) OVERRIDE changes will be bypassed. IBM recommends using the new MODIFY statements.

### 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 about the IOCP utility, see the *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 directory entries.

#### RESET

tells CP to reset the privilege classes to their original default settings.

#### classes

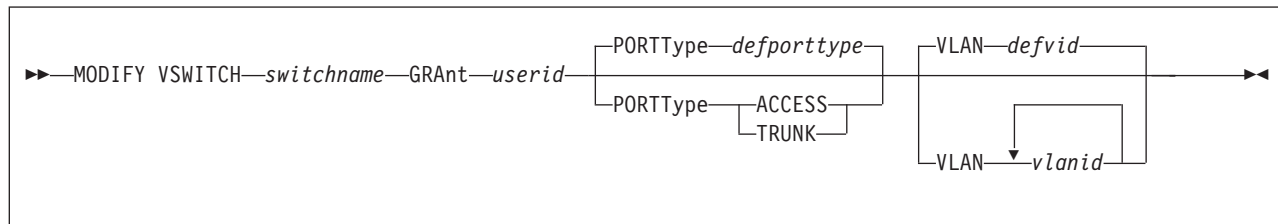
tells CP which class or classes are to be set. The variable *classes* is a 1- to 8-character alphanumeric string from A through Z and from 0 to 6.



## 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 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 about the CVTOVRID utility, see the *z/VM: CP Commands and Utilities Reference*.

## MODIFY VSWITCH Statement



### Purpose

Use the MODIFY VSWITCH statement to modify the attributes of an existing Virtual Switch.

### 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 *userid* to be added to the access list for this Virtual Switch. This allows *userid* to connect a QDIO adapter to *switchname*. If an External Security Manager (ESM) is in control of the Virtual Switch, it may override the CP access list.

#### **PORTType ACCESS**

defines the type of connections that are established by the *userid* 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 by the *userid* 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 *defvid*, untagged traffic sent or received by the guest is associated with the default VLAN ID of the virtual switch.

#### **VLAN** *vlanid*

identifies the VLAN ID to be used to restrict traffic across the adapter *userid* connected to *switchname*. When VLAN is used with GRANT, *userid* is authorized to connect to the Virtual Switch and *vlanid* is used as a filter. Specifying GRANT with one or more VLAN IDs adds to the set of VLAN IDs associated with *userid*. A limit of four *vlanids* can be associated with *userid* in the CP access list.

*vlanid* is a number from 1 to 4094.

If **VLAN** *vlanid* is not specified, the default VLAN for the *userid* is the default VLAN ID as specified on the DEFINE VSWITCH command or statement.

If an External Security Manager (ESM) is in control of the Virtual Switch, it may override the CP access list.

## Usage Notes

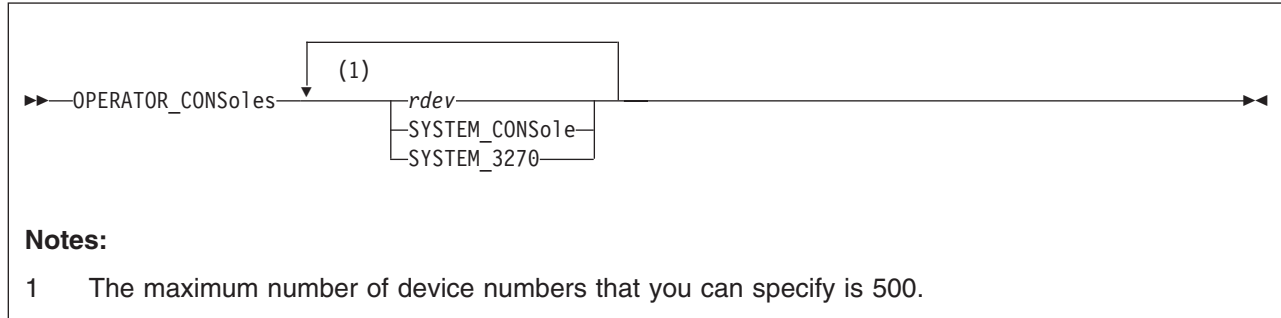
1. VLAN configuration information can be found in Planning for Guest LANs and Virtual Switches in *z/VM: Connectivity*.
2. If the Virtual Switch has been defined as VLAN UNAWARE, the PORTTYPE option and the VLAN option on GRANT are not allowed.
3. If the Virtual Switch has been defined as VLAN aware and no VLAN membership is given with the GRANT (or provided by an external security manger), the guest is a member of the Virtual Switch's default VLAN ID (*defvid*).
4. If the Virtual Switch has been defined as VLAN aware and no PORTTYPE is given with the GRANT, the guest is assigned the porttype of the Virtual Switch's default porttype (*defporttype*).

## Examples

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
```

## OPERATOR\_CONSOLES Statement



### 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 Hardware Management Console can serve as a system operator console.

#### **SYSTEM\_3270**

specifies that the integrated 3270 console on the Hardware Management Console 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. Refer to "IODF Statement" on page 158 for more information.
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 (page 124).

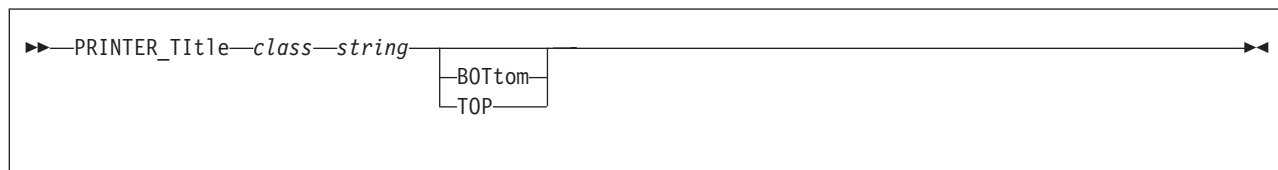
3. 2250 and 3250 displays are not valid operator consoles.
4. During a software re-IPL caused by either a system incident or the CP SHUTDOWN REIPL command, CP determines whether the operator virtual machine is connected to the primary system console. If the operator virtual machine is disconnected, logged 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 or a SYSOPER macroinstruction. 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 window overrides any consoles specified with the OPERATOR\_CONSOLES statement. See Chapter 4, "Using the Stand-Alone Program Loader," on page 37.

### 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          */
```

## 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 quote 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

- 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

- 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

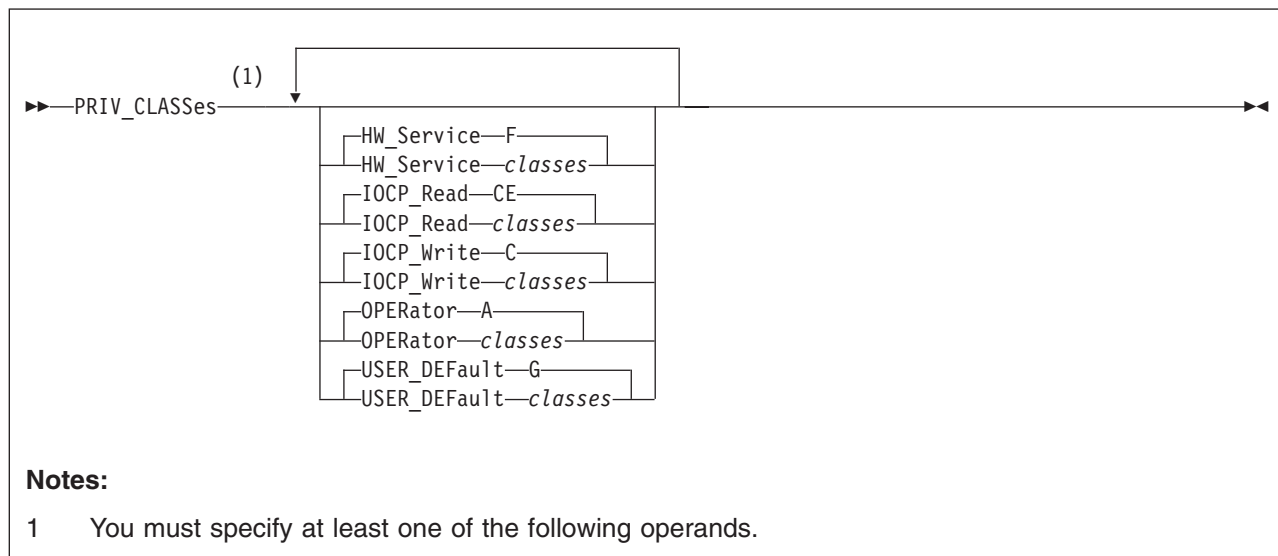
Printer_Title C 'COMPANY XYZ CONFIDENTIAL' Top

Printer_Title Q 'It''s Not Confidential'

Printer_Title X "Contents aren't Confidential"

```

## PRIV\_CLASSES Statement



## 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. (For more information about the IOCP utility, see the *z/VM: CP Commands and Utilities Reference* book.) 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.

### IOCP\_Write *classes*

tells CP which class or classes are authorized to use the write function of the IOCP utility. (For more information about the IOCP utility, see the *z/VM: CP*



*Commands and Utilities Reference* book.) 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.

#### **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 directory entries. 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

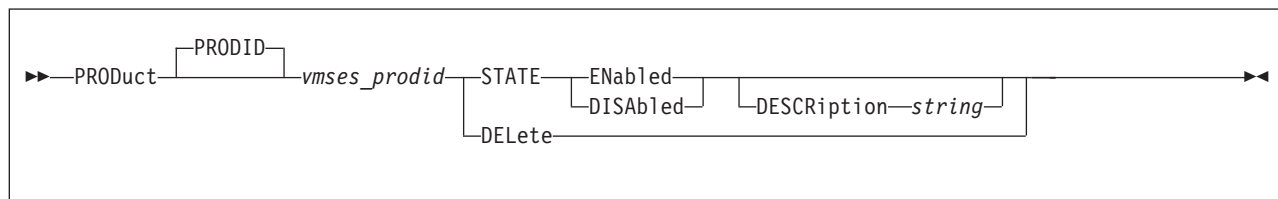
### 1. To authorize:

- 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 directory entries 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** *vmstes\_prodid*

is the identifier used by VMSES/E to install and service the given product. *vmstes\_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 *vmstes\_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 quote 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, *vmstes\_prodid*.

## Usage Notes

1. Each product is identified by a unique *vm ses\_prodid*. If a subsequent PRODUCT statement or SET PRODUCT command specifying an existing *vm ses\_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 (5684096K) to the system with a comment that describes what the specified product ID identifies, use the following PRODUCT statement:

```
PRODUCT 5684096K STATE ENABLED DESCRIPTION 'RSCS Networking Version 3
Release 2 Modification 0'
```

2. To define and enable TCP/IP VM (5735FALQ) to the system with a comment that describes what the specified product ID identifies, use the following PRODUCT statement:

```
PRODUCT 5735FALQ STATE ENABLED DESCRIPTION 'TCP/IP LEVEL 310 - TCP/IP
FEATURE (BASE)'
```

3. 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.'
```

4. 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.

5. 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
5684096K Enabled RSCS Networking Version 3 Release 2 Modification 0
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 (page 189)
- Card punches (page 190)
- Card readers (page 192)
- Communication controllers and line adapters (page 193)
- DASD (page 195)
- Graphic display devices (page 197)
- Impact printers (page 201)
- Special devices (page 205)
- Tape units (page 207)
- Unsupported devices (page 209)
- 3800 printers (page 212).

### 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.

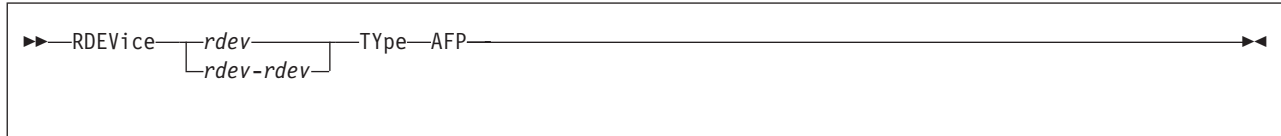
### Usage Notes

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. Refer to "IODF Statement" on page 158 for more information.
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 HCPRIO. 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 <sup>1</sup>	3820 <sup>2</sup>	3825	3827	3828
3835	3900			

1. Supported only for models 3, 6, and 8.
2. Supported only when the 3820 printer is channel-attached.

**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'.

#### 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*.

### 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:

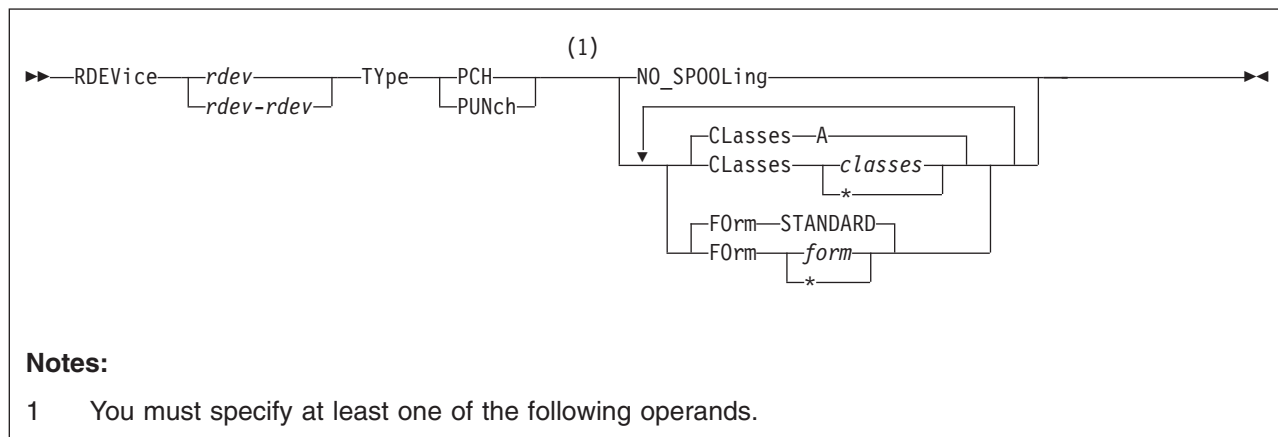
```
Rdevice 0111-0113 Type AFP
```

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 (Card Punches)

### RDEVICE Statement (Card Punches)



### Purpose

Use this RDEVICE statement to define one or more card punches to CP. CP supports the following card punch:

3525<sup>1</sup>

1. 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'.

**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 the *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.

### Examples

1. To define a card punch at real device number 0120, use the following RDEVICE statement:

```
Rdevice 0120 Type Punch
```

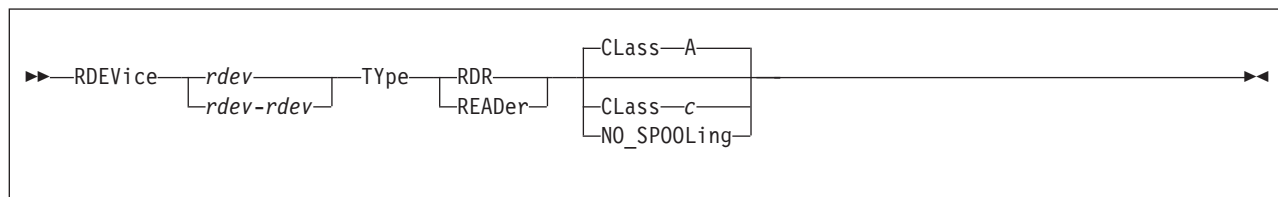
2. To define three card punches at device numbers 0121, 0122, and 0123, and to indicate that they can process any file regardless of form number, use one of the following RDEVICE statements:

```
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'.

**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 the *z/VM: CP Commands and Utilities Reference*.

**NO\_SPOOLing**

tells CP not to use the card reader for spooling.

### Examples

- To define a card reader at device number 0130, use the following RDEVICE statement:
 

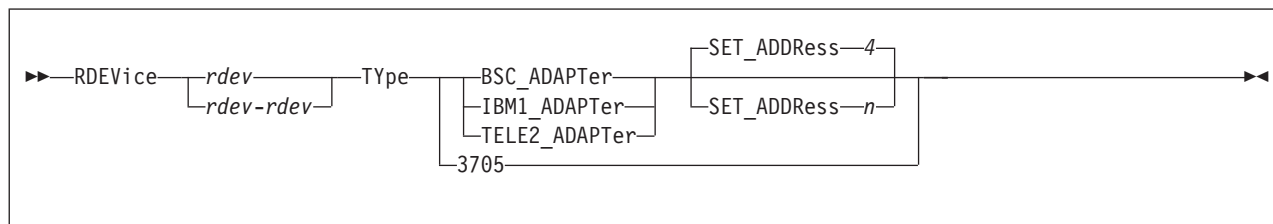
```
Rdevice 0130 Type Reader
```
- To define three card readers at device numbers 0131, 0132, and 0133, and to indicate that CP should not use them for spooling, use one of the following RDEVICE statements:
 

```
Rdevice 0131-0133 Type Reader No_Spooling
```
- 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 following communication controllers and line adapters:

3705                      3720                      3725                      3745

**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'.

#### TYpe

tells CP that the real devices you are defining are:

##### **BSC\_ADAPTer**

tells CP that you are defining one or more IBM Binary Synchronous Terminal Control Type II devices for a 3705, 3725, or 3745. (The BSC adapter is supported as dedicated-only.)

##### **IBM1\_ADAPTer**

tells CP that you are defining one or more IBM Terminal Adapter Type I attaches a 2741 to a 3705, 3725, or 3745.

##### **TELE2\_ADAPTer**

tells CP that you are defining one or more CPT-TWX (Models 33 or 35) Terminal, 3101, 3151, 3161, 3162, or 3163 attaches to a Line Interface Base Type I in a 3705, 3725, or 3745.

##### **SET\_ADDRes *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.

## RDEVICE (Communication Controllers)

### 3705

defines the base address used to load the 3705 communications controller.

## Usage Notes

1. You must define the base (load) address of a 3705 communications controller using the RDEVICE TYPE 3705 statement, using the CP SET RDEVICE TYPE 3705 command, or by genning the 3705 base address in HCPRIO. Although this is not a requirement for the 3720, 3725, or 3745 base addresses (because they can be sensed dynamically), you may also define the base addresses for these devices as TYPE 3705 in this same manner. For more information about the CP SET RDEVICE command, see the *z/VM: CP Commands and Utilities Reference*.
2. 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.
3. 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

## Examples

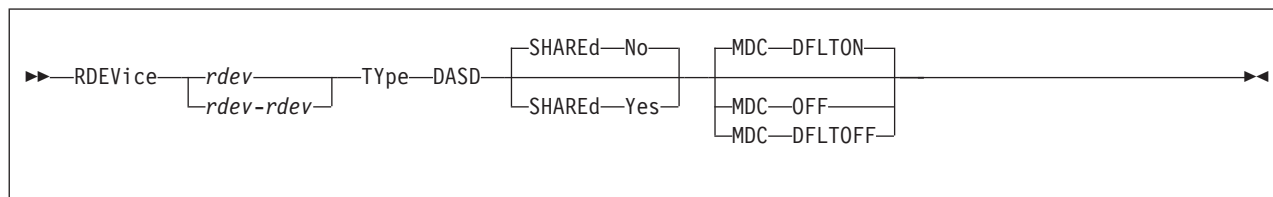
1. To define an IBM Terminal Adapter Type 1 at device number 01F0, use the following RDEVICE statement:

```
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 an IBM 3705 communication controller at device address 004, and 6 3161, 3162, and 3163 lines at device addresses 030 through 035, use the following RDEVICE statements:

```
Rdevice 004      Type 3705          /* 3705 load address */
Rdevice 010-02F Type BSC_Adapter   /* Bisync lines for PVM */
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:

3380                      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, new function (like the system configuration file) does not support the 3850. When devices are no longer supported, you may find that the existing code has not been removed from the product. If you have a 3850, you may still be able to use it by coding an RDEVICE macroinstruction in HCPRIO ASSEMBLE. For more information, see “DASD Types” on page 746.

### Operands

*rdev*

*rdev-rdev*

is the real device number (or numbers) of the DASD 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'.

#### 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.

#### 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. For more information on sharing DASDs, see Chapter 21, “DASD Sharing,” on page 619.

## RDEVICE (DASD)

### 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 to not 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.

## Examples

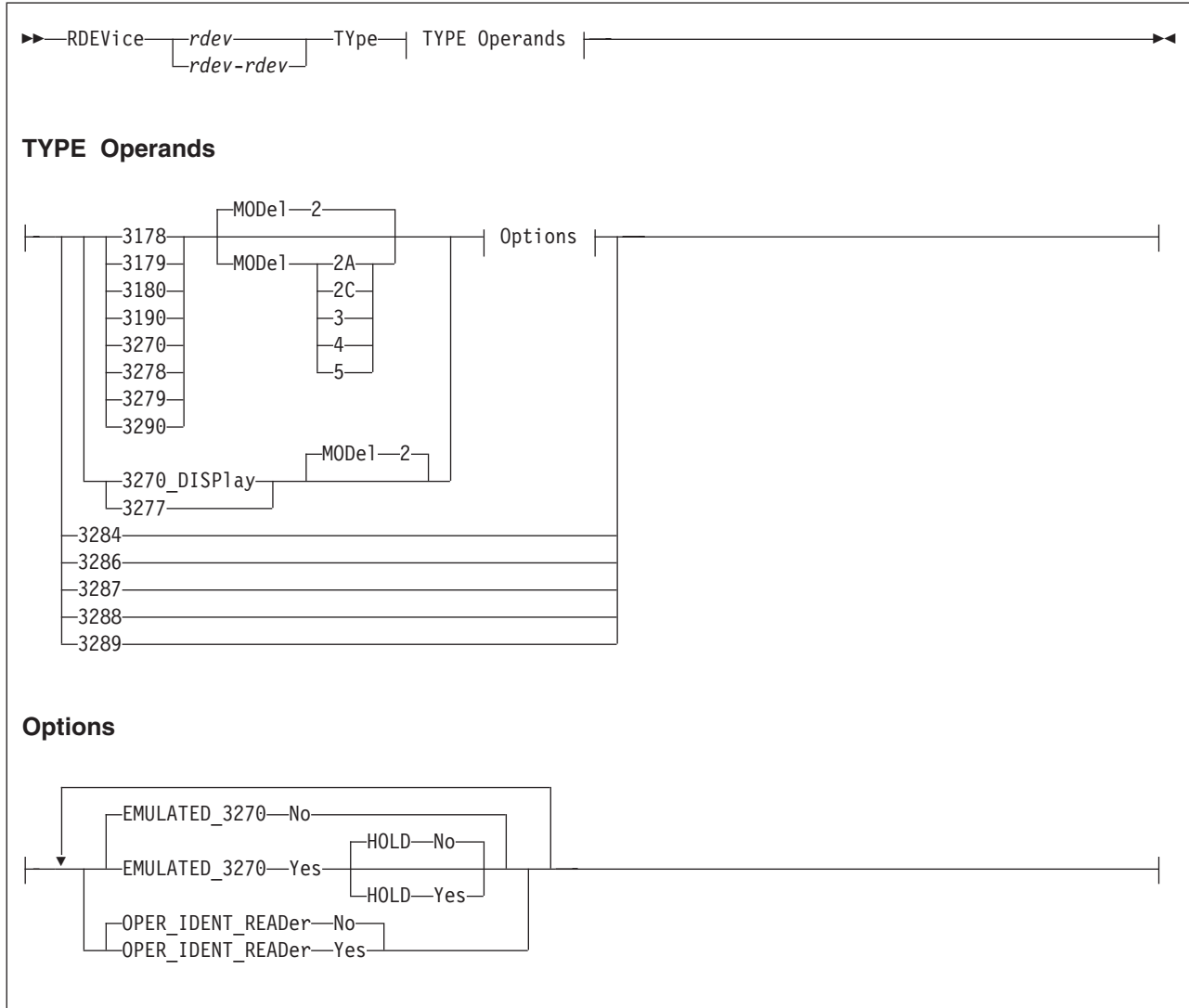
1. To define a shared DASD with minidisk caching off at device number 0410, use the following RDEVICE statement:  

```
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 non-shared DASD with minidisk caching on, no RDEVICE statement is necessary.
4. To define a non-shared 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'.

#### **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.

## RDEVICE (Graphic Display Devices)

### **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 327x display does not have an operator identification card reader.

### **OPER\_IDENT\_READER Yes**

tells CP that the 327x 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 supported for this release, see Chapter 5, "Defining I/O Devices," on page 45.
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.

## 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:

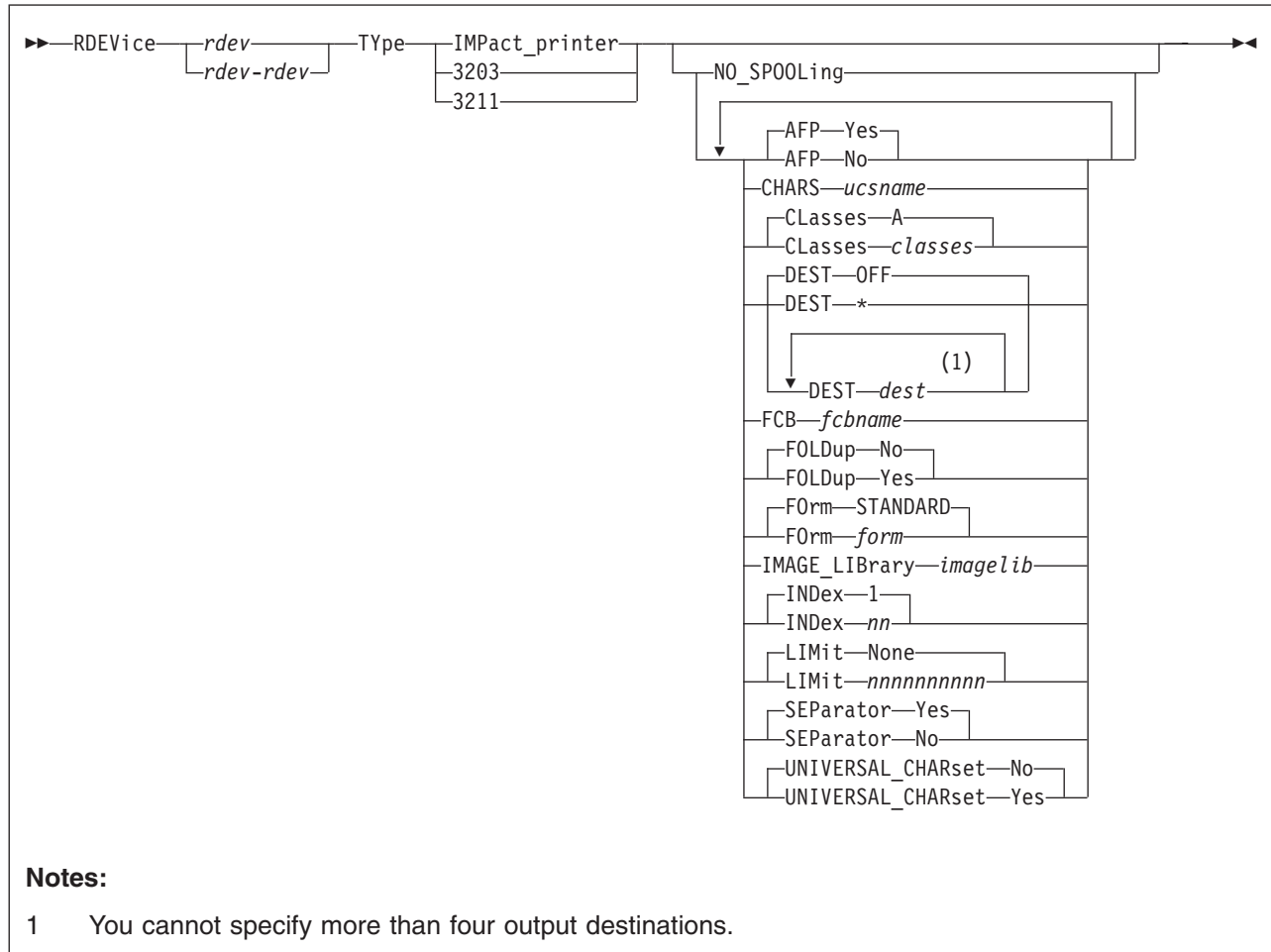
```
Rdevice 800-82f  Type 3270_Display /* 3174 controller */
/* ... with 32 local ports */
Rdevice 830-83f  Type 3277 /* 3272 controller */
/* ... with 16 3277 displays */
Rdevice 970-98e  Type 3278 /* 3274 controller */
/* ... with 31 3178s and 3278s */
```

## RDEVICE (Graphic Display Devices)

```
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)



## Purpose

Use this RDEVICE statement to define the following system-managed, channel-attached impact printers to CP:

3203 <sup>1</sup>	3211 <sup>2</sup>	3262	4245	4248
6262				

1. Only the 3203 Model 5 is supported.
2. 3211 printers are only supported as emulated by other printers.

## 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'.

**Type IMPact\_printer**

**Type 3202**

## RDEVICE (Impact Printers)

### **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 about the CP START UR command, see the *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 about the CP START UR command, see the *z/VM: CP Commands and Utilities Reference*.

### **FCB** *fcname*

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.

## RDEVICE (Impact Printers)

### 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

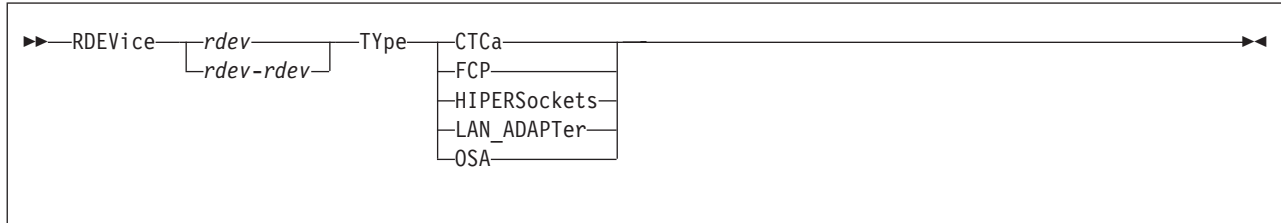
use the following RDEVICE statements:

```
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'.

#### **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 LAN\_ADAPTer**

tells CP that the real device (or devices) that you are defining is a channel-attached 8232 or 3172 representing a local area network.

#### **TYpe OSA**

tells CP that the real device (or devices) that you are defining is an Open-Systems-Adapter device.

### Usage Notes

1. CP can sense 3172 and 8232 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.

### 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 a local area network adapter at device number 0251, use the following RDEVICE statement:

## RDEVICE (Special Devices)

```
Rdevice 0251 Type LAN_Adapter
```

3. To define an Open-Systems-Adapter device at device number 0252, use the following RDEVICE statement:

```
Rdevice 0252 Type OSA
```

4. The following RDEVICE statements define:

- 48 channel-attached 3172s at device numbers 700 through 72F
- 48 channel-attached 8232s at device numbers 600 through 62F
- One channel-to-channel adapter at device address 250

```
/*-----*/  
/* 3172 LAN attached control unit - LAN adapter looks like a 3088 to */  
/*      CP. Note: you do not have to have a statement for 3088      */  
/*      looking devices ...                                          */  
/*-----*/
```

```
Rdevice 700-72f Type LAN_Adapter
```

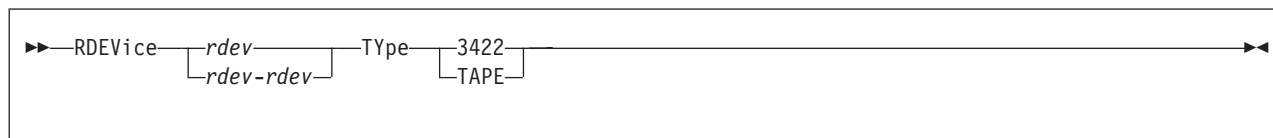
```
/*-----*/  
/* 8232 LAN attached control unit - Note: 8232s look like 3088s to */  
/*      CP but sometimes, the software does not respond correctly so */  
/*      you can be assured it is a 3088 device if you gen it as      */  
/*      LAN_Adapter                                                  */  
/*-----*/
```

```
Rdevice 600-62f Type LAN_Adapter
```

```
/*-----*/  
/* 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'.

#### **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, 3590, or 9348 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.

### 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
```

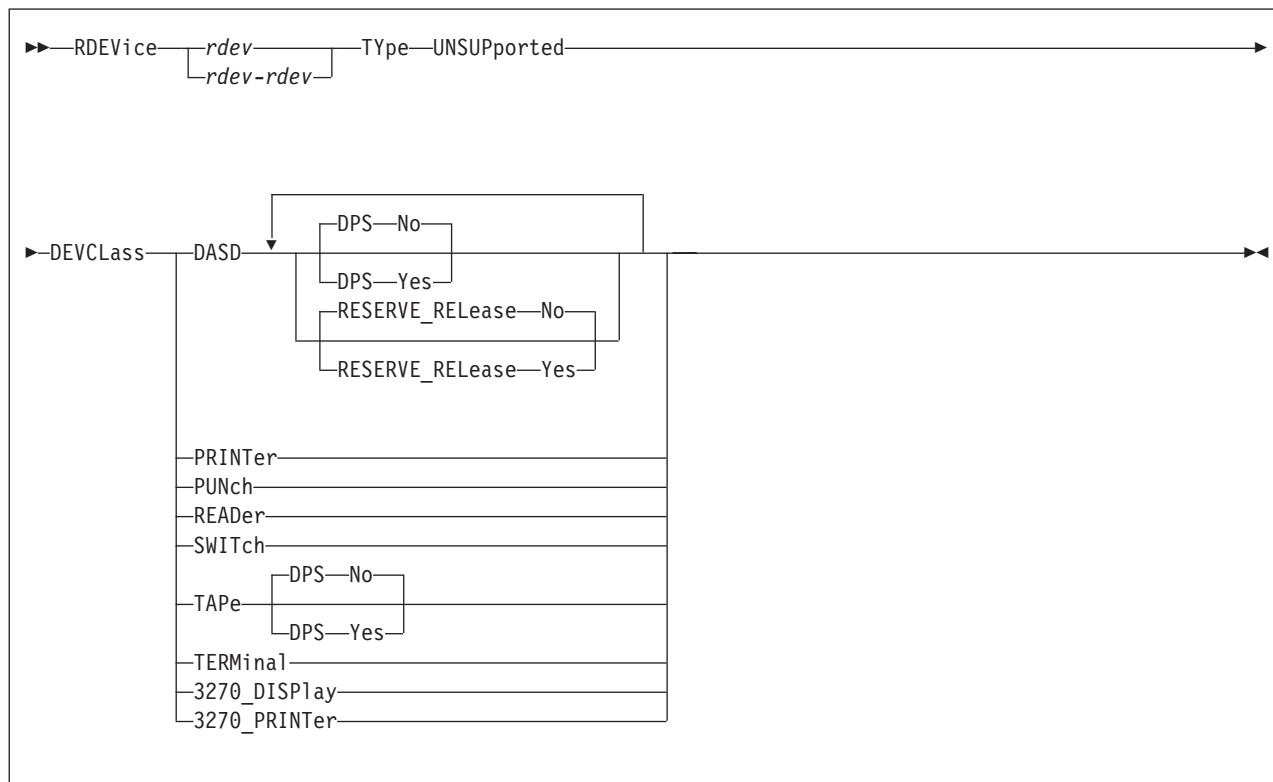
## RDEVICE (Tape Units)

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 (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 control statement in the virtual machine's directory statement or issue the CP ATTACH command. For more information about the DEDICATE statement, see page 477. For more information about the CP ATTACH command, see the *z/VM: CP Commands and Utilities Reference* book.

### 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'.

#### 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

## RDEVICE (Unsupported 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**

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 IBM's MVS/XA™ or MVS/ESA™, has an alternate-PGID capability so it can deal with the VM PGID on these channel paths.

### **DPS No**

(the default) 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**

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**

(the default) 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.

### 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:

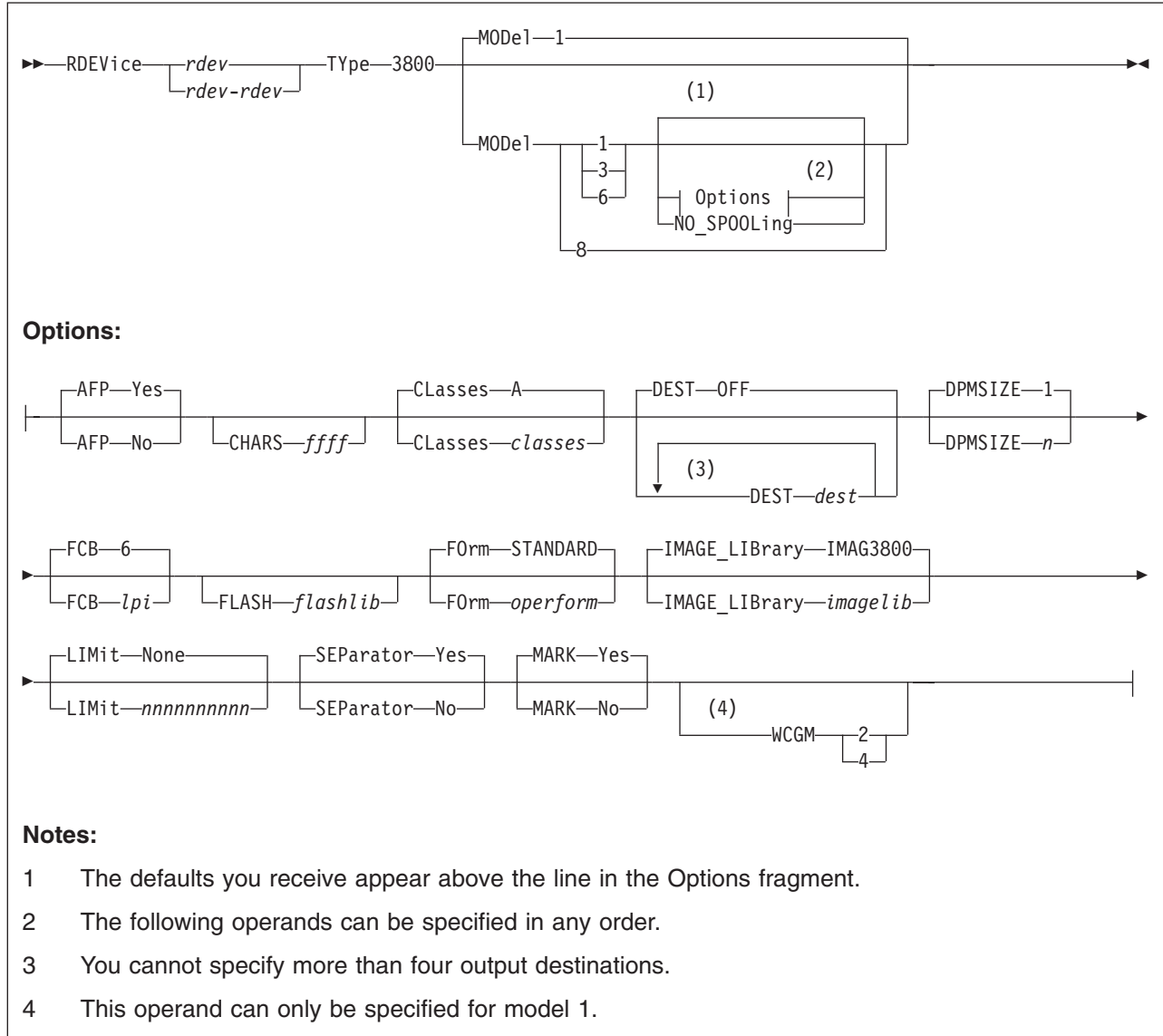
```
Rdevice 0fd0 Type UnSupported,  
              DevClass DASD,  
              DPS yes,  
              Reserve_Release yes
```

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)



## 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'.

### 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.

**Notes:**

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 the *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 about the CP START UR command, see the *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).

## RDEVICE (3800 Printers)

### Notes:

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 using CP Exit points that are provided in separator page processing. See the *z/VM: CP Exit Customization* for details.

### 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.

## Examples

- 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
```
- 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
```
- 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 */
```

## RDEVICE (3800 Printers)

```
/*                                                                    */
/* 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
```



## 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 quote 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

## SAY

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

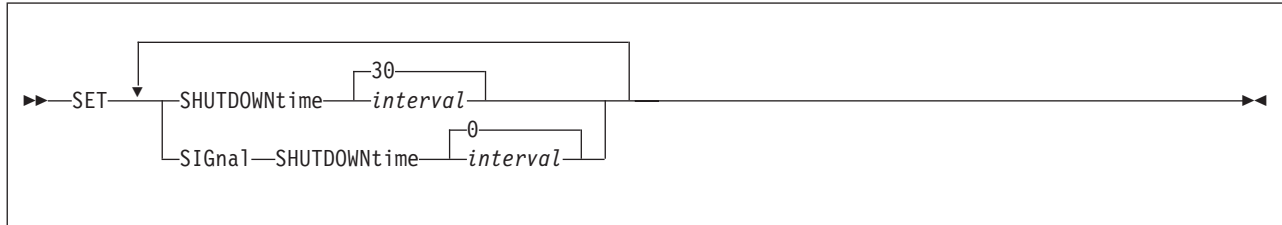
1. If your system configuration file contains the following:

```
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 Statement



### Purpose

Use the SET statement to define default system configuration values.

### Operands

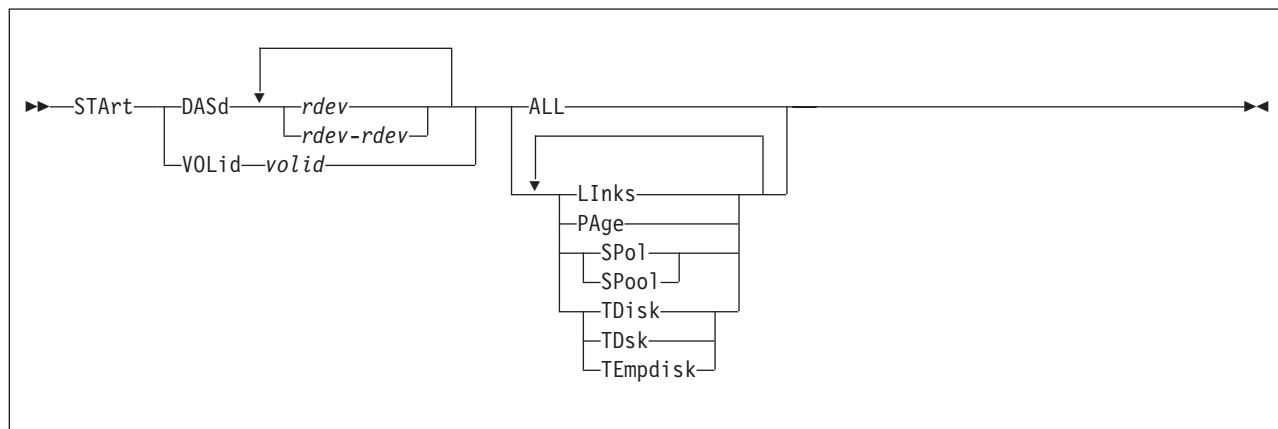
#### **SHUTDOWNtime** *interval*

defines the amount of time reserved for VM to shut down when a SHUTDOWN command is issued or a hardware termination signal is received. The *interval* specifies a number of seconds in the range 0 to 32767. If SET SHUTDOWNTIME is not specified in the system configuration file, the default is 30 seconds.

#### **SIGnal SHUTDOWNtime** *interval*

defines the duration of the default shutdown signal timeout interval in seconds. This value is used as the default timeout interval for SIGNAL, FORCE, and SHUTDOWN commands when no explicit interval is specified. The *interval* specifies a number of seconds in the range 0 to 32767. If SET SIGNAL SHUTDOWNTIME is not specified in the system configuration file, the default is 0, which suppresses guest shutdown signals.

## 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** *valid*

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 (page 73).

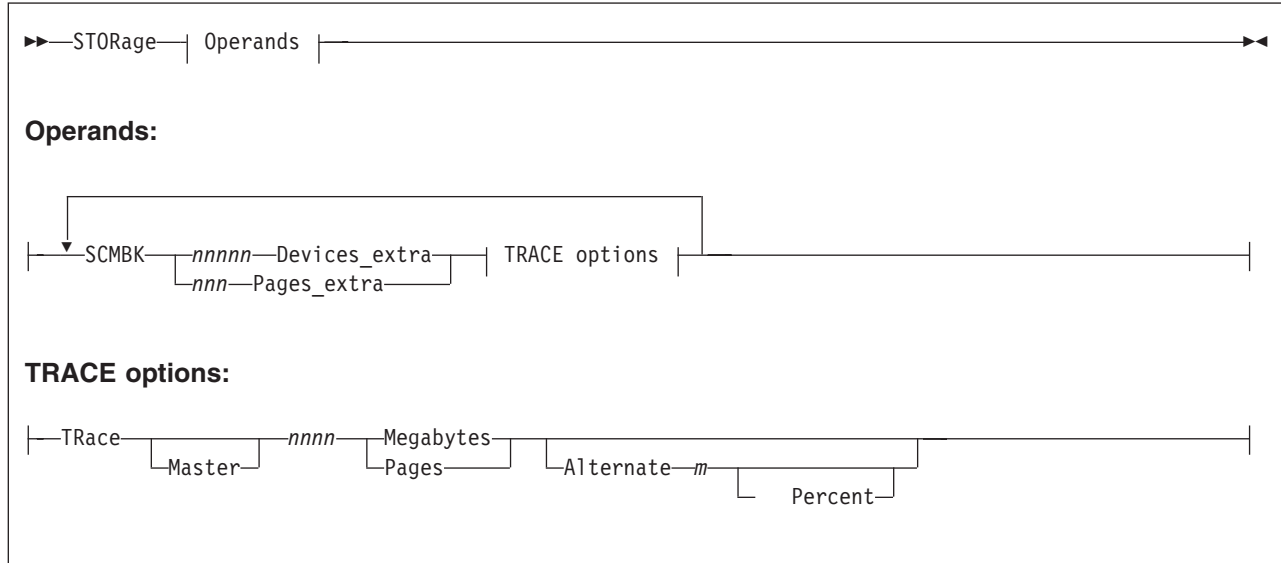
## Examples

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



### Purpose

Use the STORAGE statement to configure the use of real storage.

### 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.

### Operands

#### SCMBK *nnnnn* Devices\_extra

#### SCMBK *nnn* Pages\_extra

tells CP how much additional storage to allocate for subchannel measurement blocks (SCMBKs). CP allocates the SCMBK space in contiguous pages of storage. During initialization, CP allocates 1 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 to allocate to the internal trace table for the master processor and alternate processors in the configuration. 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 dyadic processor 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.

You can specify *nnnn* as any decimal number between 3 pages and less than one fourth of real memory.

## Usage Notes

1. What CP uses as the size of the real memory depends on what you have coded in HCPSYS and the amount of real storage installed on the system. If in HCPSYS you use the SYSSTORE macroinstruction to specify a real memory size smaller than what is actually available, CP will use the size specified in HCPSYS. We recommend that you not use SYSSTORE, or that you code it to tell CP that two gigabytes of memory (the largest amount possible) are available, so that CP itself can determine the actual amount of memory in the system. If you do not code the macroinstruction, CP will default to two gigabytes.
2. You can override what you or CP has determined to be the amount of memory available by using the IPL parameter screen of the Stand-Alone Program Loader (SAPL) when you IPL the system.
3. In storage configurations that have a large area of unaddressable storage (a hole) near the top of the storage address range, and with only a small amount of storage addressable after the hole, you may want to code the SYSSTORE macro to limit the system's use of storage to the large contiguous area below the hole. This will have two effects:
  - a. the QUERY (real) STORAGE response will be closer to the actual usable storage, and
  - b. the system will not map the unaddressable area of storage with control blocks which may use more storage than would be gained by the small amount of usable storage that is available above the hole.

One example of this type of storage configuration is a physically partitioned processor. The processor controller makes some of the storage available to one partition, but it doesn't always make contiguous address ranges available to one side or the other.

**Note:** Many processors do provide only contiguous storage ranges to each partition.

## STORAGE

4. The SCMBK operand will be ignored when format-1 measurement blocks are being used.

## Examples

1. The following STORAGE statement allocates:

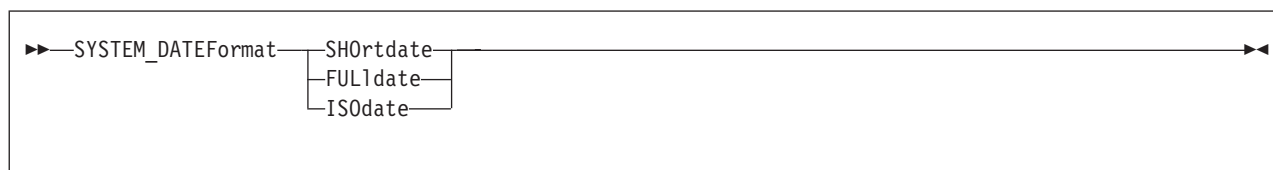
- 1 megabyte of storage for the internal trace table

```
Storage,                               /* Configure Use of Real Storage */
```

```
Trace Master 1 M Alternate 90 Percent,  
/* Allocate 1 Megabyte of space */  
/* Master Processors trace table */  
/* and 90% of that amount for */  
/* each of the other processors */  
/* in the complex */
```



## 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

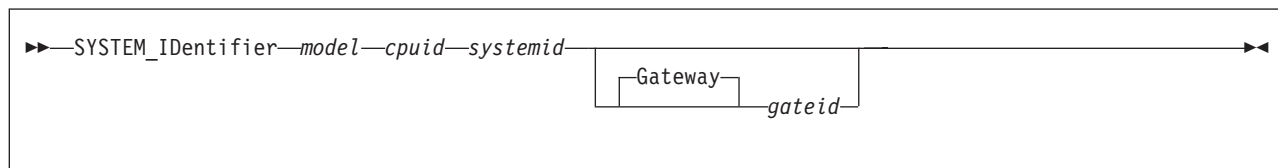
1. If the SYSTEM\_DATEFORMAT statement is not in the system configuration file, then the system-wide default date format is SHORTDATE.
2. The format of the dates, such as mm/dd/yy, mm/dd, and yy/mm/dd, are dependent upon the command or routine that displays or generates the date.

### 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



### Purpose

Use the SYSTEM\_IDENTIFIER statement to create a system identifier for the processor on which you run z/VM. If your installation has several processors, you can specify a SYSTEM\_IDENTIFIER statement for each one. At initialization, CP matches the processor model and CPU identification numbers with those specified with the SYSTEM\_IDENTIFIER statements. The selected system identifier 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 ID 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.

### Operands

#### *model*

is a character string with a length from 1 to 4 that specifies the processor unit model number. CP evaluates the string using pattern matching rules that are described below in Usage Note 2 on page 227. An asterisk (\*) tells CP to match any processor unit model number.

#### *cpuid*

is a character string with a length from 1 to 6 that specifies the processor identification number of a CPU in the processor complex. CP evaluates the string using pattern matching rules that are described below in Usage Note 2 on page 227. An asterisk (\*) tells CP to match any processor identification number.

#### *systemid*

is the 1- to 8-character system identification.

#### **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 *systemid* for the system. 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.

## Usage Notes

1. The EQUATE statement (page 133) defines nicknames that you can use as record qualifiers. You can also use the *systemid* on the SYSTEM\_IDENTIFIER statement as a record qualifier. Just be sure to define the *systemid* in a SYSTEM\_IDENTIFIER statement before you try to use it as a nickname.
2. 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.

## Examples

1. To define 3 9121 processors in a CSE complex with a system identifier of BOSTON3, use the following SYSTEM\_IDENTIFIER statements:

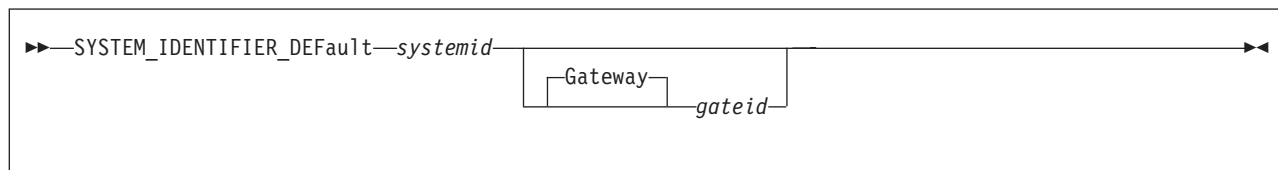
```
System_Identifier 9121 020721 Boston3 /* Set Boston3 as our */
System_Identifier 9121 120721 Boston3 /* system identifier */
System_Identifier 9121 220721 Boston3 /* for processor 0/1/2 */
/* in the complex */
```

2. Here is another way to do what Example 1 does. In addition, this method tells CP to match any processor model and any processor in that complex, even more than the 3 processors shown in Example 1.

```
System_Identifier *%20721 Boston3 /* Set Boston3as our */
/* system identifier */
/* for processor 0/1/2 */
/* in the complex */
```

---

## SYSTEM\_IDENTIFIER\_DEFAULT Statement



### Purpose

Use the `SYSTEM_IDENTIFIER_DEFAULT` statement to provide CP with a default system identifier. 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 identifier 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 ID 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.

### 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

*systemid*  
is the 1- to 8-character system identifier.

**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 *systemid* for the system. 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.

### Examples

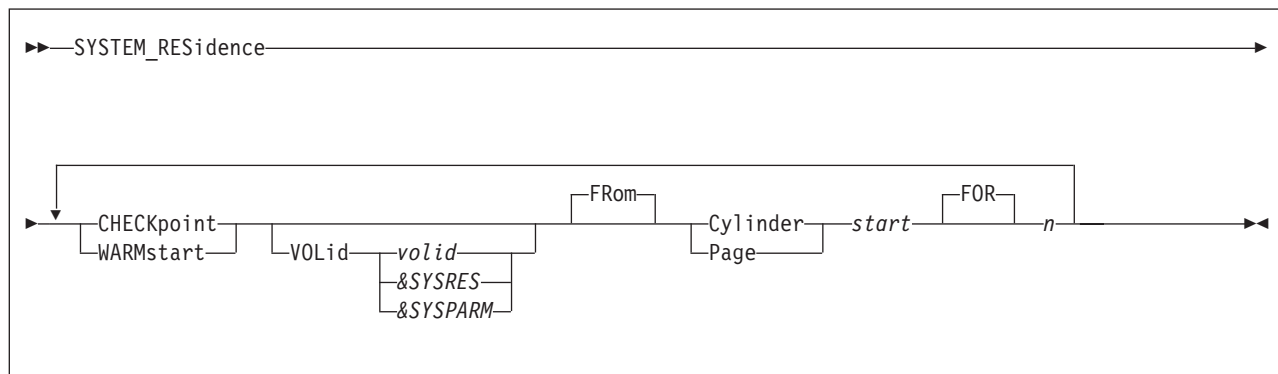
1. To define a default system identifier of `BOSTON3` for your system, use the following `SYSTEM_IDENTIFIER_DEFAULT` statement:

## SYSTEM\_IDENTIFIER\_DEFAULT

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 *valid*

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 65535. 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.
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 valid ESARES from Cyl 100 for 5
```

you could change it to:

```
SYSTEM_RESIDENCE CHECKpoint valid 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:

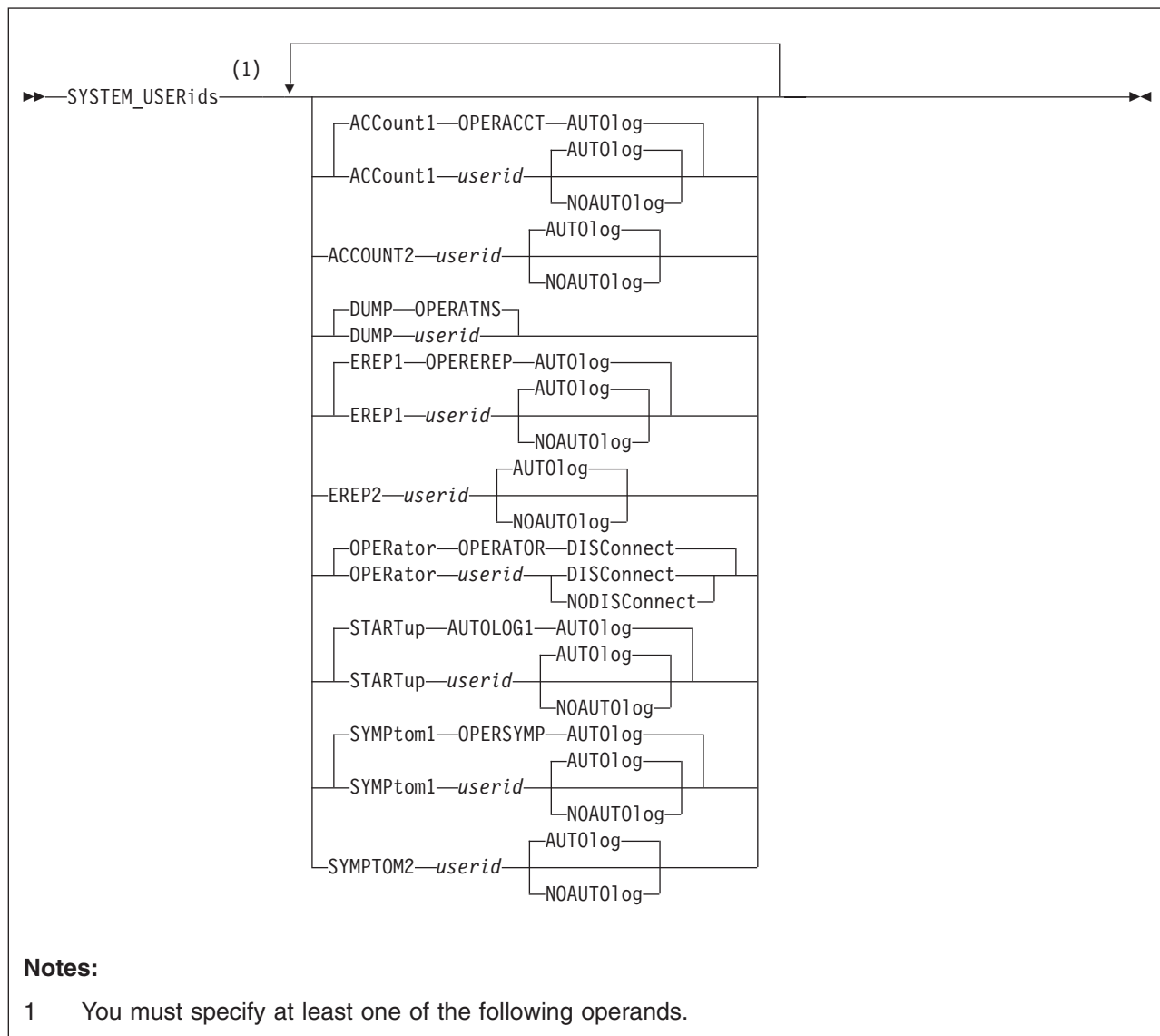
```
System_Residence,
  CheckPoint VolID sys002 from Cylinder 101 for 2,
  WarmStart  VolID sys001 from Cylinder 204 for 1
```

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



### 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 234.

**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 an entry in the user directory, that user ID is logged on at system initialization.

## SYSTEM\_USERIDS

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 an entry 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 an entry 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 about the ALTERNATE\_OPERATORS statement, see "ALTERNATE\_OPERATORS Statement" on page 58. For more information on defining operator privilege class, see the "PRIV\_CLASSES Statement" on page 184 or the "SYSFCN (Optional)" on page 673.

7. If the user ID specified on the SYSSYMP statement has an entry in the user directory at system initialization, the virtual machine designated by that user ID is logged on to the system.
8. If you are changing records relating to ACCOUNT, EREP or SYMPTOM, refer to the following sections: "Disassociating a User ID from the Retrieval of Accounting Records" on page 298, "Disassociating a User ID from the Retrieval of EREP Records" on page 317, and "Disassociating a User ID from the Retrieval of Symptom Records" on page 320.

## Examples

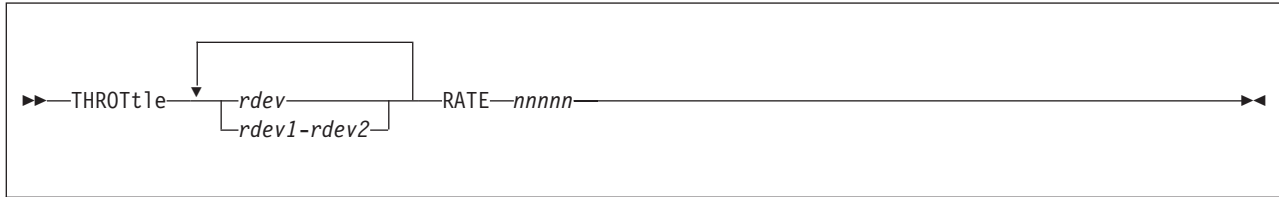
1. The following SYSTEM\_USERIDS statement specifies:
  - The system operator as user ID OPERATOR and to specify that CP should disconnect the system operator when a software re-IPL occurs and the system operator is not logged on to the system console
  - Dumps should go to the OPERATNS user ID
  - CP 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 */
```

## SYSTEM\_USERIDS

```
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 about the SET THROTTLE command, see the *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 about the QUERY THROTTLE command, see the *z/VM: CP Commands and Utilities Reference*.

### Examples

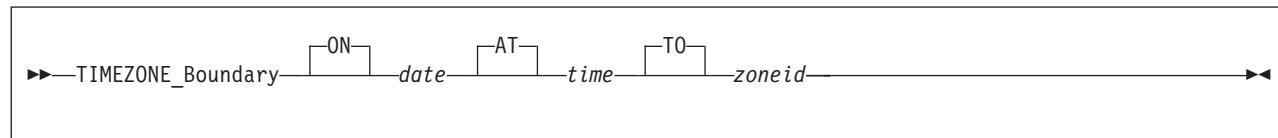
1. To limit all guest operating systems using 1c10 to 1 I/O operations per second, use the following THROTTLE statement:

```
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 3-character time zone definition established by an earlier TIMEZONE\_DEFINITION statement.

### Usage Notes

- 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 about the TIMEZONE\_DEFINITION statement, see page 239.
- 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 1992-04-05 at 02:00:00 to edt
TimeZone_Boundary on 1992-10-25 at 02:00:00 to est
TimeZone_Boundary on 1993-04-04 at 02:00:00 to edt
TimeZone_Boundary on 1993-10-31 at 02:00:00 to est
  
```

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 April 5, 1992, CP assumes that the local time is UTC, because nothing has told it otherwise. If you IPL between 02:00:00 UTC on April 5, 1992, and 02:00:00 EDT on October 25, 1992, CP assumes that EDT is the local time. If you IPL after 02:00:00 EDT on October 25, 1992, but before 02:00:00 EST April 4, 1993, CP takes EST as the local time; if you IPL any time after 02:00:00 EST on April 4, 1993, CP assumes that the local time is EDT. CP determines which time

## TIMEZONE\_BOUNDARY

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 about the TIMEZONE\_DEFINITION statement, see page 239. For more information about the CP SET TIMEZONE command, see the *z/VM: CP Commands and Utilities Reference*.

## Examples

1. To define eastern daylight time (EDT) and eastern standard time (EST) through the year 1999, use the following TIMEZONE\_BOUNDARY statements:

```
/*----- Timezone Boundary Condition Definitions -----*/
```

```
TimeZone_Boundary on 1992-04-05 at 02:00:00 to edt  
TimeZone_Boundary on 1992-10-25 at 02:00:00 to est
```

```
TimeZone_Boundary on 1993-04-04 at 02:00:00 to edt  
TimeZone_Boundary on 1993-10-31 at 02:00:00 to est
```

```
TimeZone_Boundary on 1994-04-03 at 02:00:00 to edt  
TimeZone_Boundary on 1994-10-30 at 02:00:00 to est
```

```
TimeZone_Boundary on 1995-04-02 at 02:00:00 to edt  
TimeZone_Boundary on 1995-10-29 at 02:00:00 to est
```

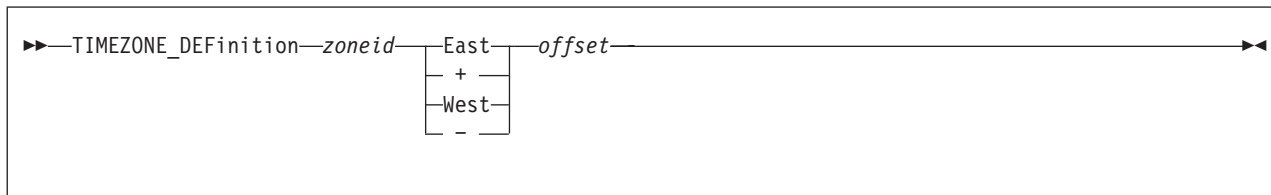
```
TimeZone_Boundary on 1996-04-07 at 02:00:00 to edt  
TimeZone_Boundary on 1996-10-27 at 02:00:00 to est
```

```
TimeZone_Boundary on 1997-04-06 at 02:00:00 to edt  
TimeZone_Boundary on 1997-10-26 at 02:00:00 to est
```

```
TimeZone_Boundary on 1998-04-05 at 02:00:00 to est  
TimeZone_Boundary on 1998-10-25 at 02:00:00 to est
```

```
TimeZone_Boundary on 1999-04-04 at 02:00:00 to edt  
TimeZone_Boundary on 1999-10-31 at 02:00:00 to est
```

## TIMEZONE\_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 3-letter time zone ID that you are defining. The variable *zoneid* must be a 1- to 3-character identifier.

#### 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 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

- 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 about the CP DEFINE TIMEZONE command, see the *z/VM: CP Commands and Utilities Reference*.

### Examples

- To define time zone:
  - EDT as 4 hours west of UTC

## TIMEZONE\_DEFINITION

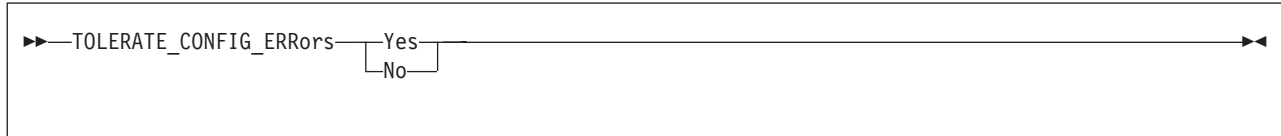
- EST as 5 hours west of UTC
- DRB as 5 ½ hours west of UTC
- RFC as 4 ½ hours west of UTC
- IKA as 3 ½ hours west of UTC
- DAC as 4 ½ hours east of UTC
- PDT as 7 hours west of UTC
- PST as 8 hours west of UTC

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

- 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.
- 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.
- 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

## TOLERATE\_CONFIG\_ERRORS

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:

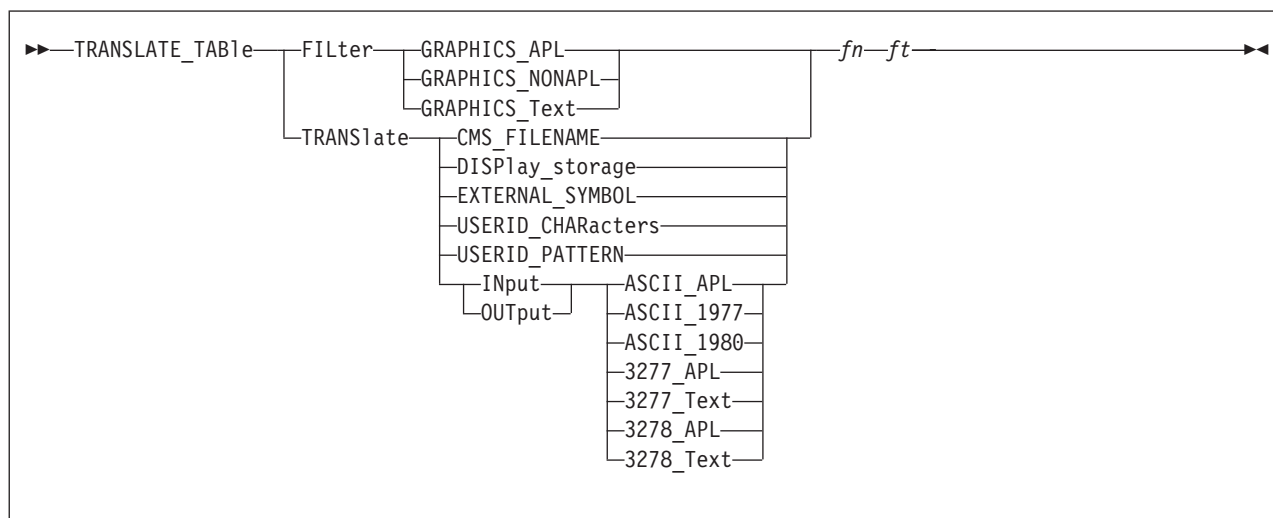
```
Tolerate_Config_Errors no /* Don't Ignore Any Errors in */  
/* the System Residence Stmt */
```

```
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 a 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.

## TRANSLATE\_TABLE

For more information about the CP DISPLAY and CPTYPE commands, see the *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. See “USER Directory Control Statement (Control)” on page 572 for points to consider in selecting valid characters for user IDs.

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 *
 *-----*/

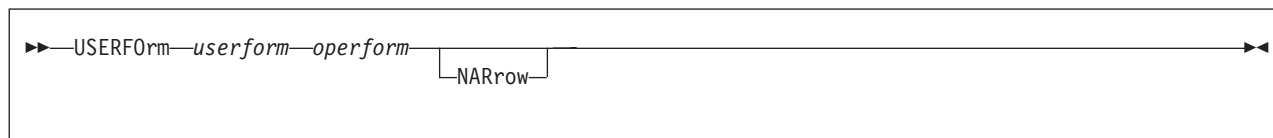
 4B4B4B4B 4B4B4B4B 4B4B4B4B 4B4B4B4B /* .... ..
 */
 4B4B4B4B 4B4B4B4B 4B4B4B4B 4B4B4B4B /* .... ..
 */
 4B4B4B4B 4B4B4B4B 4B4B4B4B 4B4B4B4B /* .... ..
 */
 4B4B4B4B 4B4B4B4B 4B4B4B4B 4B4B4B4B /* .... ..
 */
 404B4B4B 4B4B4B4B 4B4B4B4B 4C4D4E4F /* .... .. <(+|
 */
```

## TRANSLATE\_TABLE

```
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

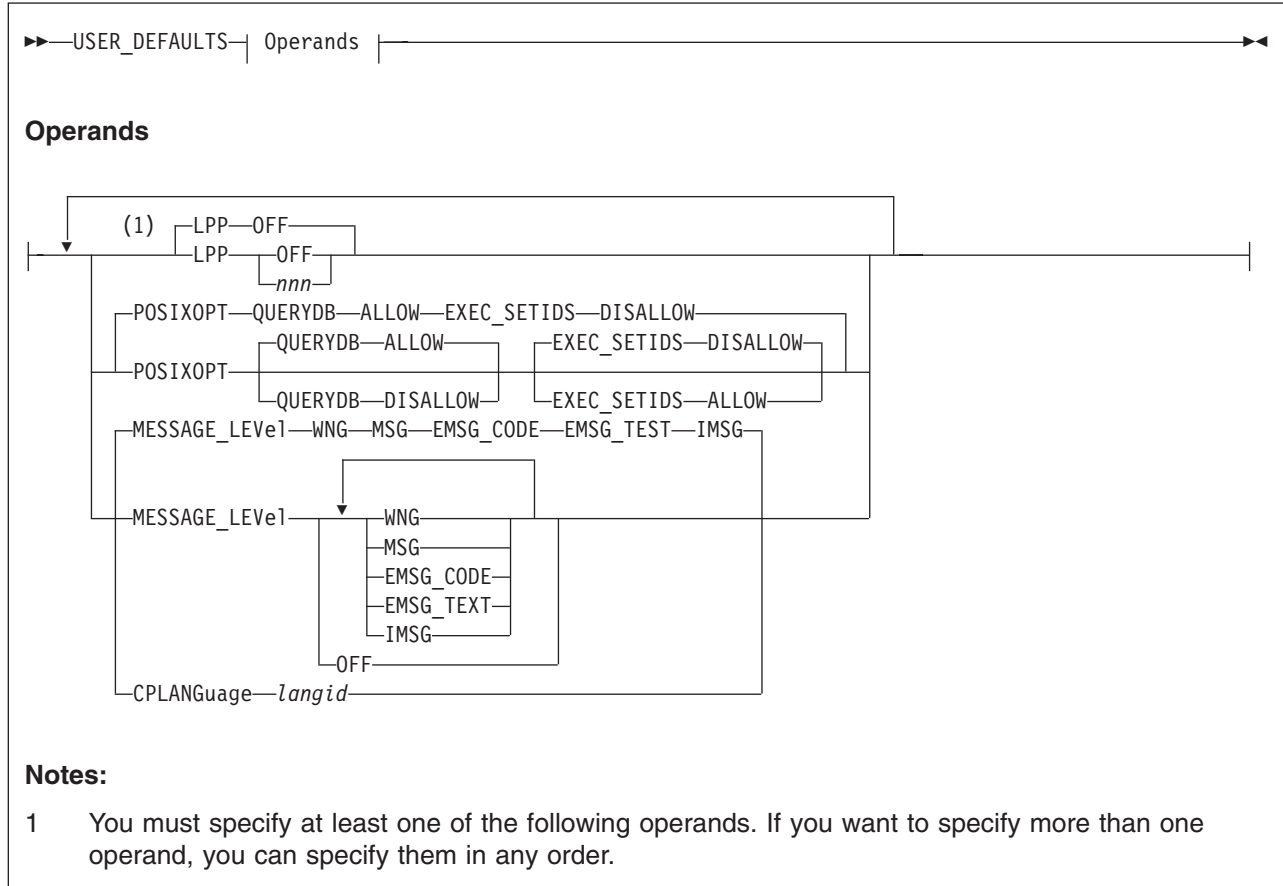
1. The *userform* and *operform* operands establish the correspondence between the user form and operator form. If you do not specify any user form and operator form pairs, CP does not generate a form list and does not distinguish between user and operator forms.

### 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 System Configuration File Statement



### 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 control statement ("POSIXOPT Directory Control Statement (General)" on page 548) to that users' directory entry.

**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.

## USER\_DEFAULTS

### **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
```

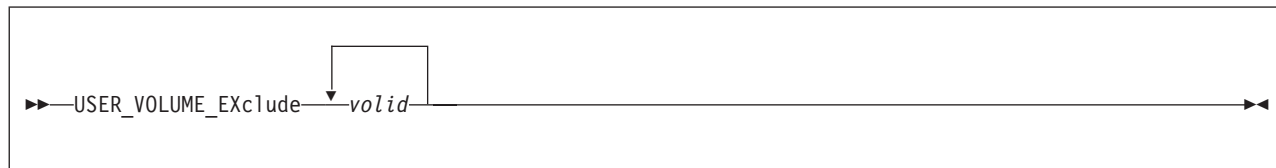
2. To specify that the default lines per page value for all virtual printers and virtual consoles is 56, use the following USER\_DEFAULTS statement:

```
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

#### *valid*

is a volume serial number or a generic volume serial number for a specific subset of volumes. The variable *valid* 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 *valid* 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 *valid* 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.

## USER\_VOLUME\_EXCLUDE

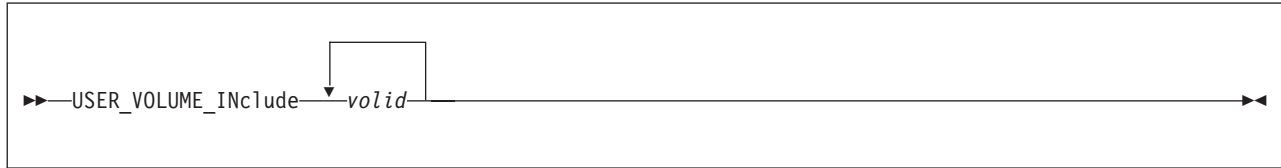
2. Dedicated devices are not user volumes. They must be attached to the user to which they are dedicated and not to the system. Therefore, if you specify USER\_VOLUME\_INCLUDE statements, you may want to specify dedicated devices in USER\_VOLUME\_EXCLUDE statements.

## Examples

1. To exclude volume XAUSRF from the user volume list, use the following USER\_VOLUME\_EXCLUDE statement:

```
User_Volume_Exclude  xausrf  /* Exclude the following User Volume  */
                        /* 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

#### *valid*

is a volume serial number or a generic volume serial number for a specific subset of volumes. The variable *valid* 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 (page 251).
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 *valid* 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

## USER\_VOLUME\_INCLUDE

It satisfies a USER\_VOLUME\_INCLUDE pattern and does not satisfy a USER\_VOLUME\_EXCLUDE statement.

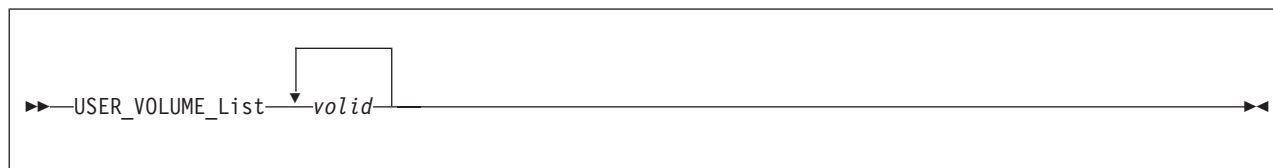
3. If a volume is normally attached to the system using a USER\_VOLUME\_INCLUDE statement, CP does not notify the operator if the volume is not mounted. If a user volume is necessary for normal system operation, specify it with a USER\_VOLUME\_LIST statement (page 255) so that the operator is notified during system initialization that the volume is not mounted.

## Examples

1. To include all volumes whose first 5 characters are XAUSR in the user volume list, use the following USER\_VOLUME\_INCLUDE statement:

```
User_Volume_Include  xausr*      /* Include the following User Volumes */
                               /* 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

*valid*

is the volume serial number of the DASD that you want included in the user volume list. The variable *valid* 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 about the CP ATTACH command, see the *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 (page 253).
4. When more than one DASD volume is found at system initialization (IPL) to have the same volume serial number (*valid*), and that volume serial number is in the CP or user volume list, the volume attached to the system is the one at the lowest device number. This rule does not apply to duplicates of the system residence volume. For more information about the system residence volume, see note 2 on page 231.
5. A duplicate *valid* causes the entire statement to be rejected.
6. A volume is automatically attached to the system if:
  - It is explicitly listed in the USER\_VOLUME\_LIST statement

## USER\_VOLUME\_LIST

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.

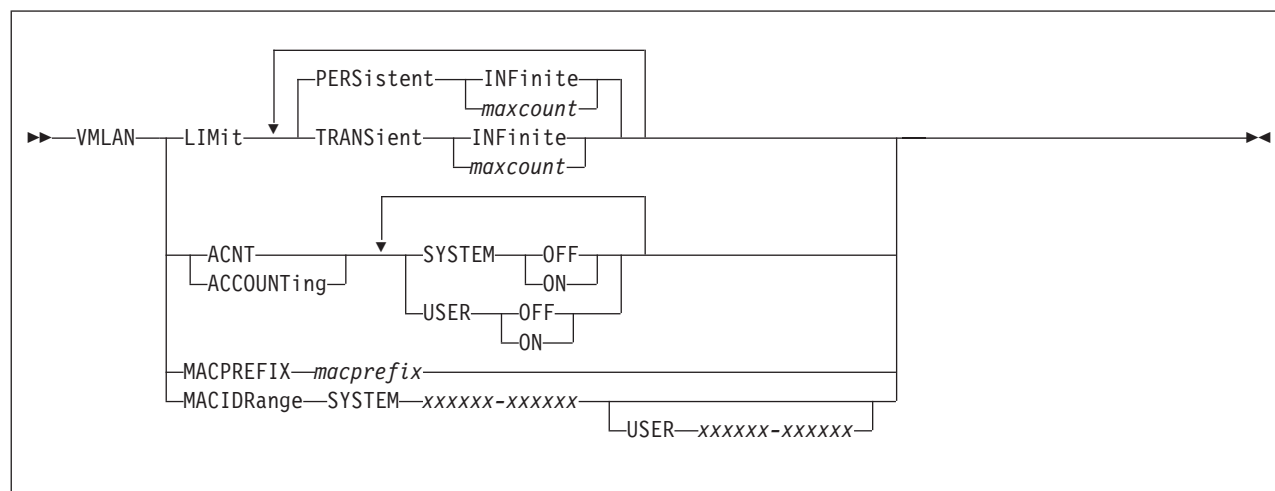
## Examples

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. */
```



## VMLAN Statement



## Purpose

Use the VMLAN statement to set control global attributes for VM Guest LAN support.

## Operands

### **PERsistent INFinite**

indicates the number of PERSISTENT VM LAN segments and Virtual Switches allowed on the CP Host system as infinite. This is the default.

### **PERsistent *maxcount***

is the maximum number of PERSISTENT VM LAN segments and Virtual Switches allowed on the CP Host system at one time. The value *maxcount* must be a decimal number in the range of 0 to 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 defines 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 defines a Virtual Switch.

### **TRANsient INFinite**

indicates the number of TRANSIENT VM LAN segments allowed on the CP Host system as infinite. This is the default.

### **TRANsient *maxcount***

is the maximum number of TRANSIENT VM LAN segments allowed on the CP Host system at one time. The value *maxcount* must be a decimal number in the range of 0 to 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.

### **ACNTIACCOUNTing SYSTEM ONIOFF**

Set the default accounting state for VM LAN segments 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.

## VMLAN

### ACNTIACCOUNTING USER ONIOFF

Set the default accounting state for VM LAN segments 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 three byte prefix (manufacturer ID) used when generating locally administered MAC addresses on the system. It must be 6 hexadecimal digits within the range of 020000 through 02FFFF (inclusive). In combination with the MAC ID used on the NICDEF directory statement, the MACPREFIX allows unique identification of virtual adapters within a network. If MACPREFIX is not specified, the default is 020000 (02-00-00).

### MACIDRange SYSTEM *xxxxxx-xxxxxx* USER *xxxxxx-xxxxxx*

is the range of identifiers (up to six hexadecimal digits each) to be used by CP when generating the unique identifier part (last six hexadecimal digits) of a virtual adapter MAC address. If a SYSTEM MACIDRANGE is not specified, CP creates unique identifiers in any range (000001-FFFFFF).

**USER** *xxxxxx-xxxxxx* is the subset of the SYSTEM range of identifiers that are reserved for user definition of MACIDs in the NICDEF directory statement. When specified, CP does not assign MACIDs within this USER range during creation of virtual adapters defined dynamically (DEFINE NIC) or with the NICDEF (or SPECIAL) directory statement without the MACID operand. In these cases, CP generates a unique identifier for the adapter outside of the USER range. Any MACID values specified on a NICDEF directory statement must be within the USER range or the virtual adapter is not defined during LOGON processing. If a USER MACIDRANGE is not specified, CP creates unique identifiers within the SYSTEM MACIDRANGE.

## 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 VM LAN segments is higher than the limit.
3. To prohibit users from defining any VM LAN segments on a given system, set the limit to zero (0). For example, "VMLAN LIMIT TRANSIENT 0" would prevent general users from creating VM LAN segments.
4. Use the MACPREFIX and MACIDRange parameters 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.

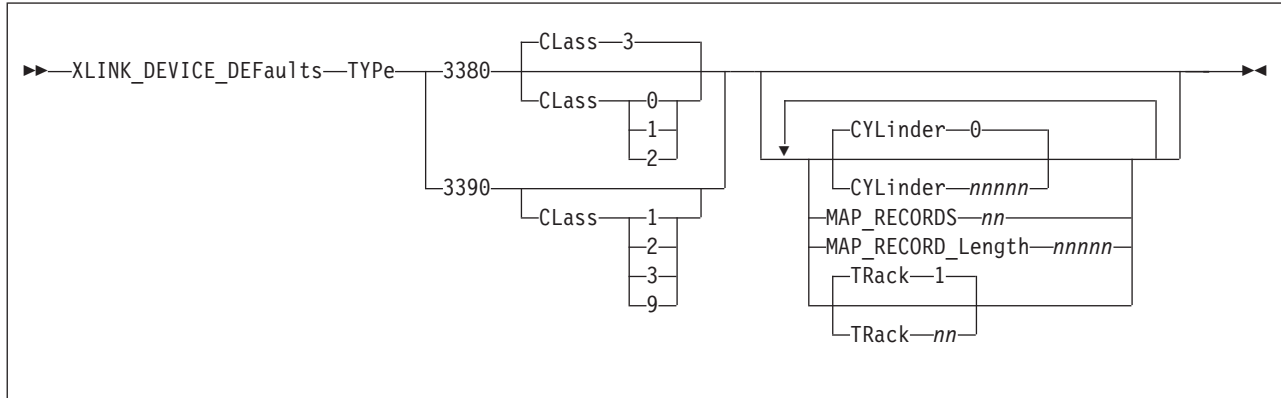
## Examples

1. To allow any number of PERSISTENT VM LAN segments, but set a limit of 20 VM LAN segments for use on a transient basis, specify the following:  

```
vm lan limit persistent infinite transient 20
```
2. To enable accounting for every user-owned VM LAN segment, specify the following:  

```
vm lan accounting user on
```

## XLINK\_DEVICE\_DEFAULTS Statement



### Purpose

Use the XLINK\_DEVICE\_DEFAULTS statement to change the defaults for the CSE track location and format for specific types of DASD.

**Note:** Although this statement allows you to change the definitions of cylinder, track, record length, and map records for the CSE track, we do not recommend that you do so. This 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.

### 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 3380

tells CP that you want to change the defaults for a 3380 DASD. Optionally, you can specify:

#### Class 0, 1, or 2

tells CP that this 3380 DASD belongs in the 1770 cylinder size group.

#### Class 3

(the default) tells CP that this 3380 DASD belongs in the 2655 cylinder size group.

If you are not sure which class to specify, we recommend that you choose a cylinder size group which is larger than the actual number of cylinders on the device or choose the default.

If you omit the CLASS operand for 3380 devices, CSE uses a value of 2655 cylinders for all 3380 type K devices and 1770 for for all 3380 non-type K devices.

## XLINK\_DEVICE\_DEFAULTS

### 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 *nnnn*

tells CP the number of the cylinder where the CSE track is located for the specified DASD type. The variable *nnnn* is a decimal number from 0 to 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 track, it is not recommended. This 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 track. The variable *nn* is a decimal number from 2 to 56. Specify as many records as will fit on one track. Table 8 shows the map record defaults for each device type.

Table 8. Map Record Defaults for Each DASD Type

Device Type	Number of Map Records	Map Record Length
3380 non-type K	11	1770
3380 type K	8	2655
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	10017 <sup>1</sup>
OTHER	8	1024

<sup>1</sup>Value can be up to the maximum number of cylinders on the DASD.

**Note:** Although this statement allows you to change the definition of map records for the CSE track, it is not recommended. This 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 CSE 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. Table 8 on page 260 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 CSE.

**Note:** Although this statement allows you to change the definitions of record length for the CSE track, it is not recommended. This 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 track. 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 track, it is not recommended. This 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. The CYLINDER, MAP\_RECORD, MAP\_RECORD\_LENGTH, and TRACK operands let you specify CSE track 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.

## XLINK\_DEVICE\_DEFAULTS

2. The default parameters for model K devices are those associated with 3380 CLASS=3, which are 2655 cylinders and 12 map records. If you want to make all 3380s use the same parameters as model Ks, add the three following statements to each system configuration file:

```
Xlink_Device_Defaults  Type 3380  Class 0,
                        MAP_Record 8,
                        MAP_Record_Length 2655
Xlink_Device_Defaults  Type 3380  Class 1,
                        MAP_Record 8,
                        MAP_Record_Length 2655
Xlink_Device_Defaults  Type 3380  Class 2,
                        MAP_Record 8,
                        MAP_Record_Length 2655
```

## Examples

1. Normal defaults for the placement of the CSE cross system link protection (XLINK) map 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 (q.v.), 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 the systems that CP is to exclude from cross system link (prevent from accessing each other's minidisks using cross system link). This statement has no effect on systems' participation in cross system spooling or cross system commands.

### 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 that CP is to exclude from cross system link. This statement has no effect on a system's participation in cross system spooling or cross system commands.

### Usage Notes

1. 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.
2. Fixed-Block Architecture (FBA) devices are supported by the CSE function for spooling only. The cross system link function of CSE is not supported for FBA devices.

### Examples

1. To exclude systems VM1, VM2, VM3, and VM4 from cross system link, 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 CSE systems will have */
Xlink_System_ExcLude  vm4    /* access to them.                */
```

## XLINK\_SYSTEM\_INCLUDE Statement

```
►► XLINK_SYSTEM_INCLUDE Slot n sysname ◄◄
```

### Purpose

Use the XLINK\_SYSTEM\_INCLUDE statement to specify the systems that CP is to include in cross system link. None of these systems need to share spool file access with any other system in the complex, nor need they send or receive cross system commands.

### 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 CSE cross system link. The maximum value of *n* depends on the type of DASD under CSE cross system link protection.

#### *sysname*

is the 1- to 8-character name of a system to be included in cross system link only.

### Usage Notes

1. 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.
2. 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.
3. The order of system names in XLINK\_SYSTEM\_INCLUDE statements for all systems participating in CSE 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.

### 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. Note: Non of these   */
/*   systems have to share spool or commands.     */
/*-----*/
```



## **XLINK\_SYSTEM\_INCLUDE**

```
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—valid—▶▶
```

### Purpose

Use the XLINK\_VOLUME\_EXCLUDE statement to define DASD volumes to be excluded from the cross system link operation. The volumes specified are not to be shared through cross system link. These volumes, if mounted and shared among systems, have no extended link protection.

### 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

*valid*

is a volume serial number or a generic volume serial number for a specific subset of volumes to be excluded from sharing by cross system link. The variable *valid* 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. If you do not specify XLINK\_VOLUME\_EXCLUDE and the system that you are IPLing has been specified in an XLINK\_SYSTEM\_INCLUDE statement, all DASD volumes are included in cross system link protection.
2. If you do not specify XLINK\_VOLUME\_EXCLUDE and XLINK\_VOLUME\_INCLUDE (page 268) statements, and the system that you are IPLing is specified on an XLINK\_SYSTEM 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.

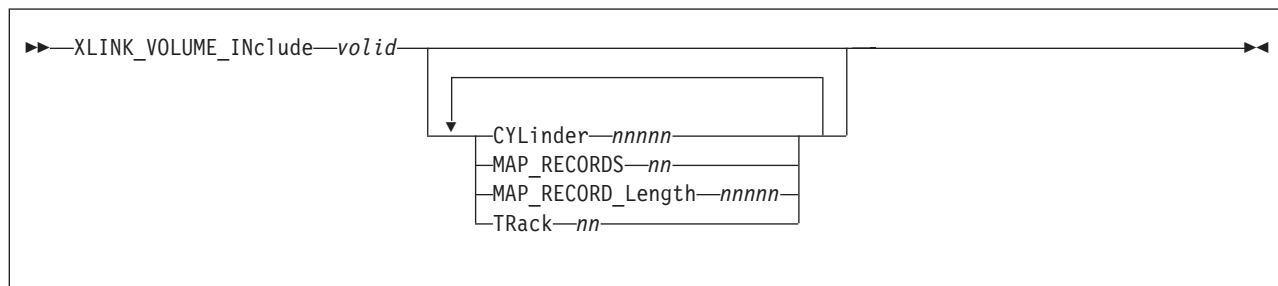
### Examples

1. To exclude volumes VMUSR4, VMUSR7, and VMUSR9 from cross system linking, use the following XLINK\_VOLUME\_EXCLUDE statements:

## XLINK\_VOLUME\_EXCLUDE

```
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 define the DASD volumes to be included in the cross system link operation. If you do not use this statement, all volumes are included.

### 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

#### *valid*

is a volume serial number or a generic volume serial number for a specific subset of volumes to be included in sharing by cross system link. The variable *valid* is a 1-character to 6-character alphanumeric string with no imbedded blanks. A generic volume serial number is a 1-character 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 track is located for every *valid*. The variable *nnnnn* is a decimal number from 0 to the maximum number of cylinders on the DASD. If omitted, the default is 0. If CP is to use a volume that is to be included in cross system link as an IPL device for the stand-alone dump utility, you must specify a cylinder other than the default of 0 for that volume.

#### **MAP\_RECORDS** *nn*

defines the number of map records on the CSE track. 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 9 on page 269 for the default values.

#### **MAP\_RECORD\_Length** *nnnnn*

defines the number of cylinders made available for CSE 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. See Table 9 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 CSE.

**TRack *nn***

defines the track number on which the CSE track is located for every CKD DASD volume matching *validn*. 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 *validn*.

## Usage Notes

1. If you do not specify XLINK\_VOLUME\_INCLUDE and XLINK\_VOLUME\_EXCLUDE (page 266) statements, and the system that you are IPLing is specified on an XLINK\_SYSTEM statement, all DASD volumes are included in cross system link protection.
2. 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.
3. 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.
4. Although the XLINK\_VOLUME\_INCLUDE enables you to change the definitions of cylinder, track, record length, and map records for the CSE track, we do not recommend that you do so. We have included this capability to provide link support for new devices with new geometries, should that support become necessary in the future. Table 9 shows the map record defaults for each device type.

*Table 9. Map Record Defaults for Each Device Type*

Device Type	Number of Map Records	Map Record Length
3380 non-type K	11	1770
3380 type K	8	2655
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

## XLINK\_VOLUME\_INCLUDE

Table 9. Map Record Defaults for Each Device Type (continued)

Device Type	Number of Map Records	Map Record Length
3390-9 (native)	56	10017 <sup>1</sup>
OTHER	8	1024

<sup>1</sup>Value can be up to the maximum number of cylinders on the DASD.

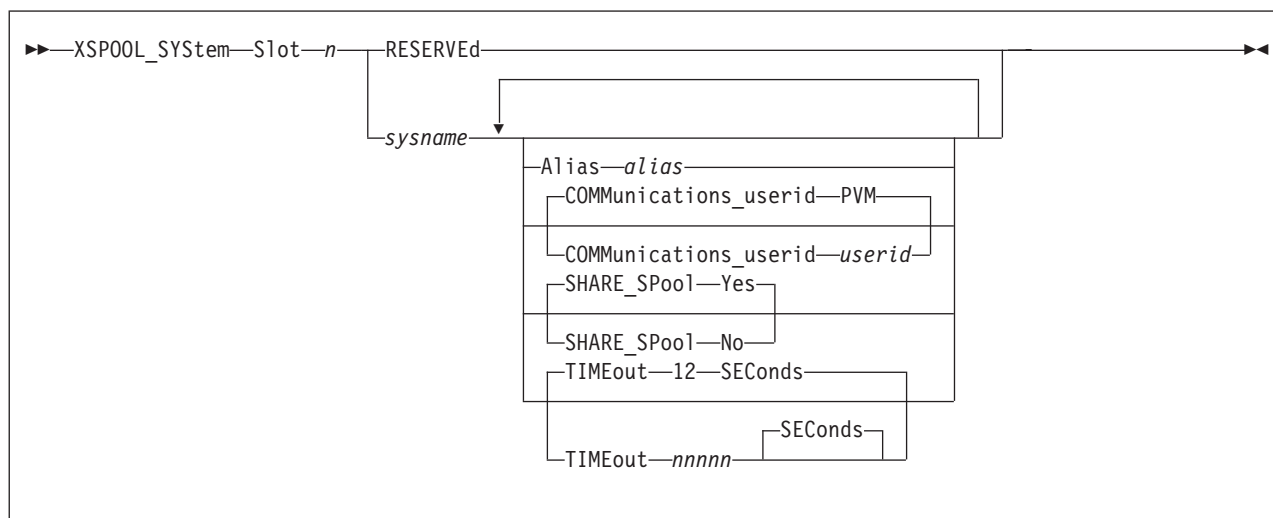
5. These defaults support the current CKD and ECKD-capable CKD DASD. The length of each map record is the maximum number of cylinders on any currently existing model of that device type. The number of records defined for other devices is 8, but the actual number of records used is limited to the number 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 ... */
```

## XSPool\_SYSTEM Statement



### Purpose

Use the XSPool\_SYSTEM statement to specify the systems that are to participate in cross system commands and spooling operations.

### How to Specify

Include as many statements as needed; they are optional. You can place XSPool\_SYSTEM 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 CSE cross system link. The variable *n* is a decimal number from 1 to 4 and the maximum value of *n* depends on the type of DASD under CSE cross system link protection.

#### RESERVED

tells CP to reserve the slot for future use.

#### *sysname*

is the 1- to 8-character system name.

#### Alias *alias*

is a 1- to 8-character nickname for *sysname* that people can use in CP commands instead of *sysname* to identify the system.

#### COMMunications\_userid *userid*

is the 1- to 8-character user ID of the communication virtual machine (CVM) for sharing spool files and cross system commands. If omitted, the default is PVM.

#### SHARE\_SPool Yes

(the default) tells CP that the specified system can share spool files with the system you are IPLing.

## XSPool\_SYSTEM

**Note:** If you have DIRMAINT Release 5 operating in your CSE complex and you specify SHARE\_SPOOL YES, you must specify GLOBAL\_SMSG=NO in the DIRMAINT DATA file.

### SHARE\_SPool No

tells CP that the specified system is excluded from sharing spool system continues to participate in cross system link and cross system commands.

### TIMEout *nnnnn* SEConds

tells CP how many seconds to wait for a response from another system. This waiting (or timeout) period applies to the overall status of the communication link, rather than being associated with a particular transaction. When the system you are IPLing sends a request to another system, CP waits for a response. If CP does not get a response before the timeout period expires, CP:

- Terminates the request with an error,
- Assumes that the CVM link to that system (or the system itself) is down, and
- Notifies the system operator (by message) of this event.

Until the system operator restores the connection, no cross system spool processing can take place between these two systems.

The variable *nnnnn* is any decimal number from 1 to 65535. If omitted, the default is 12 seconds.

## Usage Notes

1. A system name, *sysname*, must have the same slot number in all the lists on all the systems using cross system commands and shared spool operations.
2. Because some system in the list may not be using a particular slot number, you can use the RESERVED operand reserve a slot that is not used on the system that you are IPLing, but is used elsewhere in the complex.

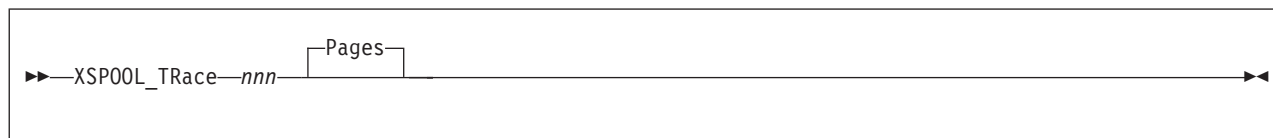
## Examples

1. To define two systems (BOSTON and NEWYORK) that will participate in cross system commands and spooling operations, use the following XSPool\_SYSTEM statements:

```
XSpool_System Slot 1 boston
XSpool_System Slot 2 newyork
```



## XSPool\_TRACE Statement



### Purpose

Use the XSPool\_TRACE statement to define the number of pages of storage that CP should allocate for the cross system spool (XSPool) trace tables.

### How to Specify

Include as many statements as needed; they are optional. You can place XSPool\_TRACE 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

#### *nnn* Pages

tells CP how many pages to allocate for the XSPool trace tables when you initialize the system. The variable *nnn* is a decimal number from 0 to 255. If omitted, CP allocates 4 pages. If specified as 0, CP does not allocate any pages for XSPool trace tables.

### Usage Notes

1. The first trace table has enough space to hold 16 synchronization requests and 111 CVM communication requests. Each additional page has space for 127 more CVM communication requests. When this number is exceeded, the trace tables wrap around, and the previously recorded entries are overwritten.

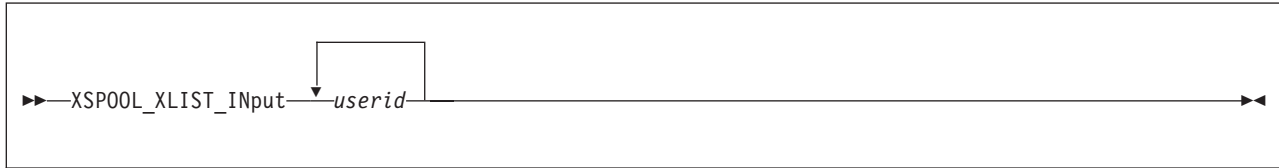
### Examples

1. To have CP allocate 10 pages of storage for the XSPool trace tables, use the following XSPool\_TRACE statement:

```
XSpool_Trace 10 Pages /* Allow for 10 CSE Trace Frames */
                    /* Each frame holds 16 Sync Requests */
                    /* and 111 CVM communication requests. */
```

---

## XSPOOL\_XLIST\_INPUT Statement



### Purpose

Use the XSPOOL\_XLIST\_INPUT statement to specify virtual machines whose input spool files will not participate in cross system spooling and cross system message and query commands. The other systems in the shared spool complex will not have access to input (reader) spool files owned by the user IDs specified in these statements.

### How to Specify

The XSPOOL\_XLIST\_INPUT statement is optional; if specified, you can include as many statements as you need as long as the total number of user IDs does not exceed 500. If you specify a generic user ID, CP counts each user that matches the pattern as one.

You can place XSPOOL\_XLIST\_INPUT 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*

is the 1- to 8-character user ID or generic user ID of a virtual machine whose input files are to be excluded from access to cross system spooling. A generic user ID 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:

```
Xspool_Xlist_Input us%vm*
```

creates a list that includes all user IDs that start with US and have VM as their fourth and fifth characters.

### Usage Notes

1. If your installation uses DIRMAINT Release 5, the DIRMAINT user ID and satellite machine user IDs must *not* be coded in XSPOOL\_XLIST\_INPUT statements.
2. User IDs listed in XSPOOL\_XLIST\_INPUT statements have operational characteristics that are different from the characteristics of those that are not listed. The users associated with the listed user IDs:
  - Can log on to more than one system in the complex at the same time. Users whose user IDs are not listed here cannot.
  - Receive messages generated by the CP MSG, MSGNOH, WNG, and SMSG commands, but only from users on the same system they are on, unless the user entering the CP command specifies the AT parameter.

Also, when these users:

- Enter a CP QUERY *userid* or QUERY NAMES command, only the system on which the command was entered is queried, unless the issuer specifies the AT parameter.
  - Are the target of a CP QUERY *userid* command, only the system on which the command was entered is queried, unless the issuer specifies the AT parameter.
3. CP automatically excludes the:
- Default system operator
  - Communication virtual machine (CVM).
- Do not code any XSPool\_XLIST\_INPUT statements for these virtual machines. You may decide that other virtual machines should also be excluded, but for ease of operation and administration of the shared spool complex, keep exclusions to a minimum.
4. The XSPool\_XLIST\_INPUT list for every system in the shared spool complex must be the same; the XSPool\_XLIST\_OUTPUT list for every system in the shared spool complex must be the same.

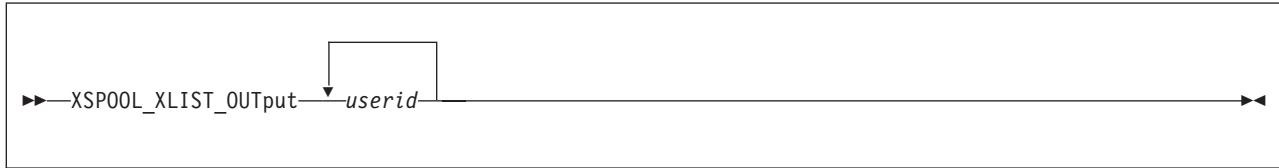
## Examples

1. To specify that the input spool files of user IDs OPERATOR, MAINT, DAWN, and RICH are excluded from cross system spooling and cross system message and query commands, use the following XSPool\_XLIST\_INPUT statements:

```
Xspool_Xlist_Input Operator      /* Userid Operator, Maint, Dawn */
Xspool_Xlist_Input Maint        /* and Rich's input files will */
Xspool_Xlist_Input Dawn         /* not participate in CSE      */
Xspool_Xlist_Input Rich         /* spooling.                   */
```

---

## XSPOOL\_XLIST\_OUTPUT Statement



### Purpose

Use the XSPOOL\_XLIST\_OUTPUT statement to specify virtual machines whose output spool files will not participate in cross system spooling and cross system message and query commands. The other systems in the shared spool complex will not have access to output (printer and punch) spool files owned by the user IDs specified in these statements.

### How to Specify

The XSPOOL\_XLIST\_OUTPUT statement is optional; if specified, you can include as many statements as you need as long as the total number of user IDs does not exceed 500. If you specify a generic user ID, CP counts each user that matches the pattern as one.

You can place XSPOOL\_XLIST\_OUTPUT 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*

is the 1- to 8-character user ID or generic user ID of a virtual machine whose output files are to be excluded from access to cross system spooling. A generic user ID 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:

```
Xspool_Xlist_Output us%vm*
```

creates a list that includes all user IDs that start with US and have VM as their fourth and fifth characters.

### Usage Notes

1. If you have DIRMAINT Release 5 installed, you cannot specify the DIRMAINT user ID and satellite machine user IDs on the XSPOOL\_XLIST\_OUTPUT statement.
2. User IDs you specify on the XSPOOL\_XLIST\_OUTPUT statement have operational characteristics that are different from the characteristics of the user IDs that are not listed. The users associated with these user IDs:
  - Can log on to more than one system in the complex at the same time. Users whose user IDs are not listed here cannot.
  - Receive messages generated by the CP MSG, MSGNOH, WNG, and SMSG commands, but only from users on the same system they are on, unless the user entering the CP command specifies the AT operand.

Also, when these users:

- Enter a CP QUERY *userid* or QUERY NAMES command, CP only queries the system on which the command was entered, unless the user specifies the AT operand.
  - Are the target of a CP QUERY *userid* command, CP only queries the system on which the command was entered, unless the person who issued the command specified the AT operand.
3. CP automatically excludes the:
- Default system operator
  - Communication virtual machine (CVM).
- Do not specify any XSPPOOL\_XLIST\_OUTPUT statements for these virtual machines. Your installation may decide that other virtual machines should also be excluded, but for ease of operation and administration of the shared spool complex, keep exclusions to a minimum.
4. The XSPPOOL\_XLIST\_INPUT list for every system in the shared spool complex must be the same; the XSPPOOL\_XLIST\_OUTPUT list for every system in the shared spool complex must be the same.

## Examples

1. To specify that the output spool files of user IDs OPERATOR, KERRY, DAVID, and MAINT are excluded from cross system spooling and cross system message and query commands, use the following XSPPOOL\_XLIST\_OUTPUT statements:

```
Xspool_Xlist_Output Operator      /* Userid Operator, Kerry, David */
Xspool_Xlist_Output Kerry        /* and Maint's output spool files */
Xspool_Xlist_Output David        /* will not participate in CSE */
Xspool_Xlist_Output Maint        /* spooling. */
```



---

## 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 10 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 10. Logo Configuration File Statements (LOGO CONFIG)*

Statement	Description	Page
CHOOSE_Logo	Defines the logo picture files that CP uses for specified terminals and for separator pages of printed output.	285
INPUT_AREA	Defines which file CP uses when displaying the input area of the logo.	291
ONLINE_MESSAge	Specifies the file whose text CP should use for the online message area of the logo.	292
STATUS	Defines the text of the status fields displayed in the bottom right corner of the screen.	293

---

### 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,

## Logo Configuration File

```
/*-----*
 * 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.

## Logo Configuration File

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. 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, enter:

```
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, highlighting, and so on) 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.

For the complete description of DRAWLOGO, see “DRAWLOGO” on page 642.

DRAWLOGO creates a new logo configuration file, but it does not put it into production. To replace the current system logo configuration file with LOCOTERM LOGO, use the CP REFRESH command:

1. Put LOCOTERM LOGO on a CMS-formatted minidisk accessed by CP.
2. Enter:

```
refresh logoinfo locoterm logo
```

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

## 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 11. 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.

## Logo Configuration File

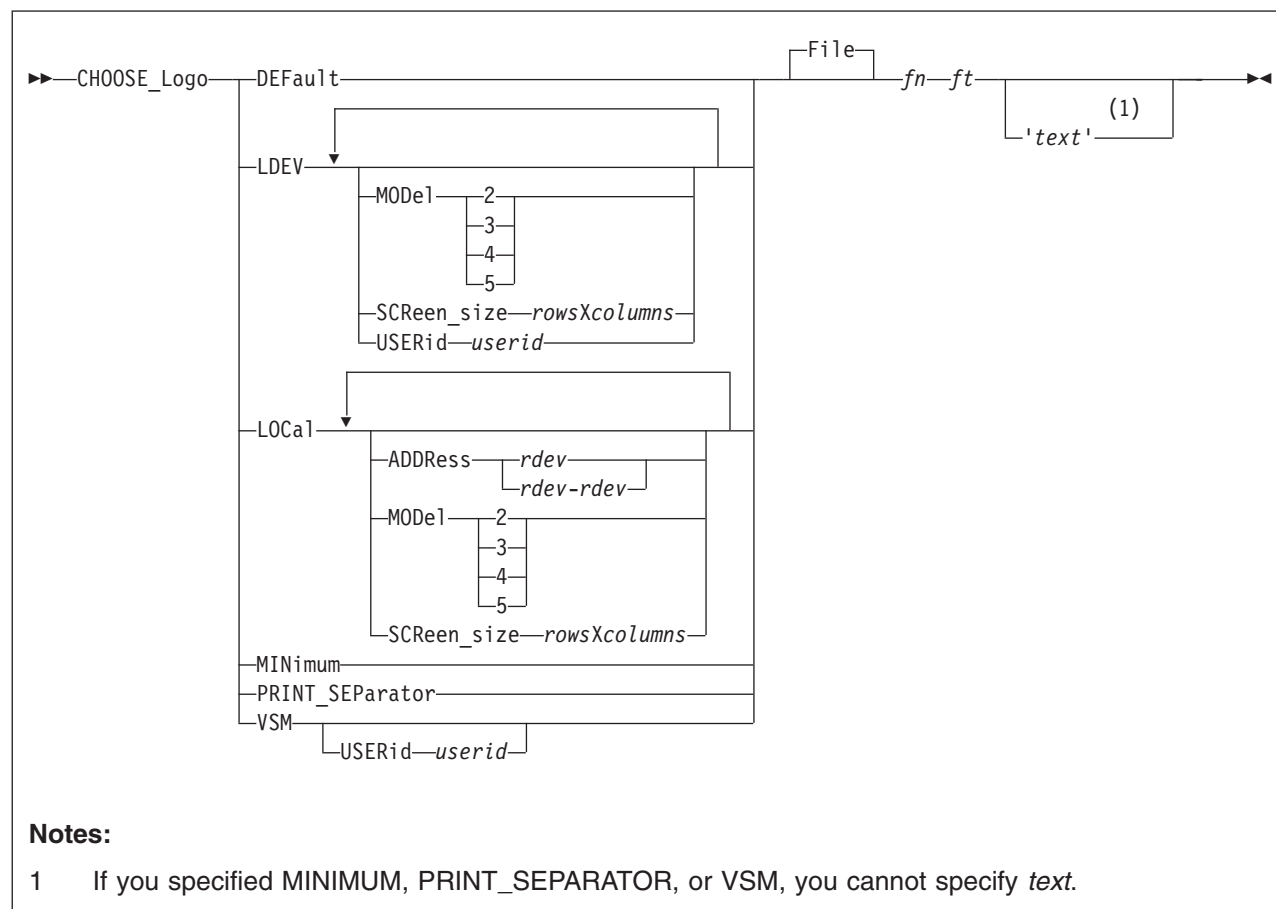
PF (program function) keys are also defined to assist in their use. A table of the different fields and their values follows.

*Table 12. Input Area Fields*

<b>Field ID</b>	<b>Hexadecimal Representation</b>
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



## 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.

## CHOOSE\_LOGO

### 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 *rowsXcolumns*

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** *rowsXcolumns*

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 quote marks (""") to represent a " character

## CHOOSE\_LOGO

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 than 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
```

## CHOOSE\_LOGO

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 641 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

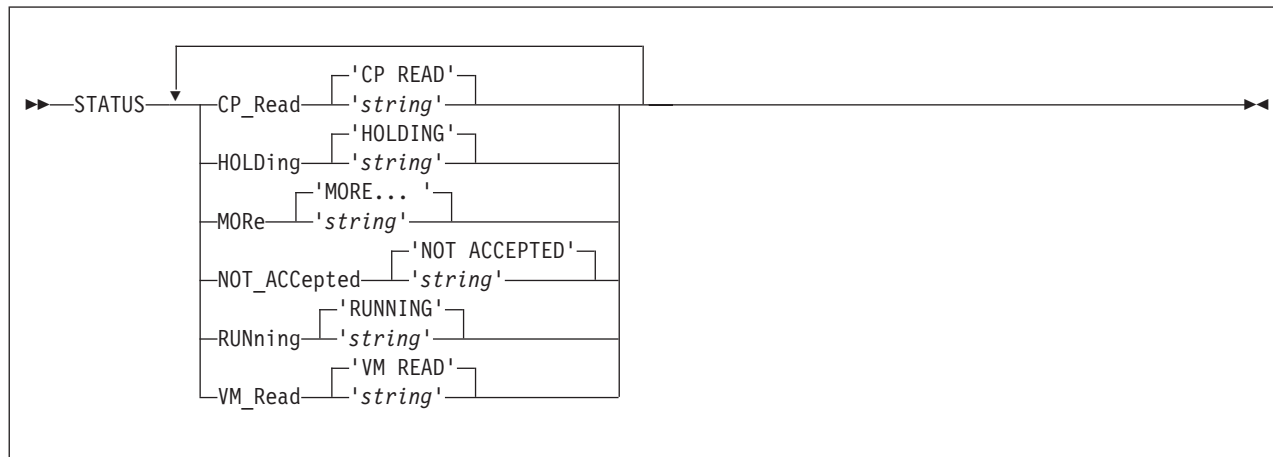
1. If the online message file contains more than one record or is wider than 78 characters, CP will use the online message defined in HCPBOX instead of the one defined in the file.
2. If the online message text is wider than the screen on which it will be displayed, CP truncates it.

### 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'.

#### **MOREe** *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'.

## STATUS

### **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 un 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 quote 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',  
       HoTding '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
- Print Services Facility/VM
- 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 user directory control 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 user directory entry 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 control 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 tapes or CD-ROM include:

- A sample directory entry for an accounting virtual machine. This entry contains the required IUCV authorization for connecting to the CP accounting system service.
- 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.

**Note:** One or two virtual machines can be set up as accounting virtual machines.

To set up a virtual machine to begin recording accounting information automatically, you must have the proper PROFILE EXEC and user directory set up. The following steps show this procedure:

1. **Log on as MAINT.** If you did this step as part of installation, you may already be logged on as MAINT, or the user ID you are installing with on your current operating system.

## Setting Up SVMs

- Determine the write password of the accounting virtual machine's 191 disk.** The accounting virtual machine has been specified in either the SYSTEM\_USERIDS ACCOUNT1 or ACCOUNT2 system configuration file statement or the SYSACNT macroinstruction. Before linking to the accounting virtual machine's 191 disk, find out its write password by examining its user directory entry. (As MAINT, you usually have access to the user directory.) If the accounting virtual machine's 191 disk does not have a write password, you must supply one and update the directory.

### Notes:

- To set up virtual machines for error recording and for symptom record recording, you must determine the write password of the 191 disk of the error recording virtual machines and symptom record recording virtual machines. To avoid having to update the directory three times, check the write password of the error recording virtual machine's and symptom record recording virtual machine's 191 disk at the same time you check the write password of the accounting virtual machine's 191 disk.
  - Verify that the directory entry for this virtual machine contains the required IUCV authorization for connecting to the CP accounting system service. For example, IUCV \*ACCOUNT.
- Link to the accounting virtual machine's 191 disk.** To do this, enter:

```
link to diskacnt 191 as 391 wr
```

When CP responds with ENTER WRITE PASSWORD: enter:

```
wpass
```

where *wpass* is the write password of the accounting virtual machine.

The accounting virtual machine's 191 disk is now your 391 disk.

- Access the 391 disk.** To do this, enter:  

```
access 391 x
```

If you receive a message that says X'391' DEVICE ERROR, you must format the 391 disk. To do this, enter:  

```
format 391 x
```

CMS responds with FORMAT WILL ERASE ALL FILES ON DISK X (391). DO YOU WISH TO CONTINUE? (YES|NO). You should enter:  

```
yes
```

When CP responds with ENTER DISK LABEL, enter:  

```
acct
```

or any 1- to 6-character label name.
- Copy a file named DVM PROFILE from MAINT's 191 disk to the 391 disk.** (DVM PROFILE is the PROFILE EXEC for the accounting, symptom record recording, and error recording virtual machines). The RETRIEVE utility, which does the IUCV connect to the \*ACCOUNT system service, is invoked from this PROFILE EXEC. To do this, enter:  

```
copyfile dvm profile a profile exec x
```
- Release and detach the 391 disk.** To do this, enter:  

```
release x (det
```
- 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 *diskacct* user ID, enter:  

```
xautolog diskacct
```



## 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 or the SYSACNT macroinstruction 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 an entry in the user directory, they will be logged on as part of CP initialization. See “SYSTEM\_USERIDS Statement” on page 232 for information about the SYSTEM\_USERIDS statement or “SYSACNT (Optional)” on page 661 for information about the SYSACNT macroinstruction.
- Additional virtual machines can also collect accounting records by following the instructions in sections:
  - See “Specifying a New Accounting Virtual Machine.”
  - See “Starting Manual Retrieval of Accounting Records.”

### 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 directory entry containing an IUCV directory control 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.”

### 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.

## Setting Up SVMs

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 CP 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 CP 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 CP 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 the SYSACNT macroinstruction is omitted and no user ID for accounting is specified in the system configuration file.

## Accounting Record Formats

### PI

CP produces the following types of accounting records:

- Virtual machine user records (record type 1).
- Records for devices dedicated to a virtual machine user (record type 2).
- Records for temporary disk space dedicated to a virtual machine user (record type 3).
- Records that are written when CP detects that a user has entered enough LOGON, AUTOLOG, XAUTOLOG, or APPCVM CONNECT invocations with an invalid password to reach or exceed an installation-defined threshold value (record type 04).
- Records that are written when CP detects that a user has successfully entered a LINK command to a protected minidisk not owned by the user (record type 05).
- Records that are written when CP detects that a user has entered enough LINK commands with an invalid password to reach or exceed an installation-defined threshold value (record type 06).
- Records generated whenever a user logs off or disconnects from a device controlled by the VCNA or VSCS (record type 07).
- Records that are written when CP detects that a user has successfully entered a LINK command to a protected minidisk not owned by that user's virtual machine (record type 08). Record type 08 is also generated when the user logs off or disconnects or when a SHUTDOWN or FORCE command causes a logged-on virtual machine to be forced off the system. Disconnected virtual machines do not have another 08 record generated for them if they are forced off.
- Records generated about ISFC (record type 09).
- Records logging changes to a user's privilege class (record type 0A)
- Records for virtual disk in storage space (record type B)
- Records for Network Data Transmissions (record type C)
- Records generated as a result of a user-initiated DIAGNOSE X'4C' instruction (record type C0).

**Note:** Record types 04, 05, 06, and 08 are generated only when LOGON, AUTOLOG, XAUTOLOG, LINK, and CONNECT journaling is on.

These records are all 80-character card images.

The size of the fields in the accounting records restricts the data limit that each field can contain. To find the available field size, refer to each record type. Since the fields are limited in their capacity, it is recommended that you cut accounting records at least once a day.

### Accounting Records for Virtual Machine Resource Usage (Record Type 1)

A type 1 accounting record is produced whenever a user logs off or whenever the ACNT command is entered. Columns 1 through 28 and 79 and 80 of this record contain character data; all other data is in hexadecimal form (the hexadecimal data is unprintable).

## Setting Up SVMs

Column	Contents
1–8	User ID
9–16	Account number
17–28	Date and time of accounting ( <i>mmddyymmss</i> )
29–32	Number of seconds connected to CP
33–36	Milliseconds of processor time used, including time for supervisor functions
37–40	Milliseconds of virtual CPU time used
41–44	Number of page reads
45–48	Number of page writes
49–52	Number of requested virtual I/O starts for non-spoiled I/O
53–56	Number of virtual punch cards sent to a virtual punch
57–60	Number of virtual print lines sent to a virtual printer (this includes one line for each carriage control command)
61–64	Number of virtual punch cards received from a virtual reader
65–68	Milliseconds of total Vector time
69–72	Milliseconds of virtual Vector time
73–76	Number of completed virtual I/O starts for non-spoiled I/O (except DIAGNOSE X'58' and DIAGNOSE X'98')
77–78	CPU address (for the SYSTEM VMDBK, this is the real processor address)
79	Card generator field (C if Diag 4C has been issued, 0 otherwise)
80	Accounting record identification code (1)

**Note:** User virtual machine time may be recorded in more than one record entry. For example:

- Processor time is accumulated under connect time (bits 29 - 32) and processor time (bits 33 - 40)
- Vector processing time is accumulated under connect time (bits 29 - 32), processor time (bits 33 - 40), and vector time (bits 65 - 72).

### Accounting Records for Dedicated Devices (Record Type 2)

A type 2 accounting record is produced whenever a virtual machine user releases a previously dedicated device. Columns 1 through 28 and 79 and 80 of this record contain character data; all other data is in hexadecimal form (the hexadecimal data is unprintable). Translation codes for columns 33 through 36 can be found in Appendix E, "Device Class and Type Codes," on page 773. See the documentation for each specific device for more complete information on model numbers.

Column	Contents
1–8	User ID
9–16	Account number
17–28	Date and time of accounting ( <i>mmddyymmss</i> )
29–32	Number of seconds since the virtual disk was created or the number of seconds since the last accounting record was cut for this virtual disk
33	Device class
34	Device type
35	Device model (if any)
36	Device features (if any)
37–78	Reserved
79	Card generator field (C if Diag 4C has been issued, 0 otherwise)
80	Accounting record identification code (2)

**Accounting Records for Temporary Disk Space (Record Type 3)**

A type 3 accounting record is produced whenever a virtual machine user releases temporary disk space. Columns 1 through 28 and 79 and 80 of this record contain character data; all other data is in hexadecimal form (the hexadecimal data is unprintable). See Appendix E, “Device Class and Type Codes,” on page 773 for information about translation codes for columns 33 through 36. See the documentation for each device for more information on model numbers.

<b>Column</b>	<b>Contents</b>
1–8	User ID
9–16	Account number
17–28	Date and time of accounting ( <i>mmddyymmss</i> )
29–32	Number of seconds since the TDISK was created or the number of seconds since the last accounting record was cut for this TDISK
33	Device class
34	Device type
35	Device model (if any)
36	Device features (if any)
37–38	<u>CKD and ECKD Only</u> Number of cylinders of temporary disk space used (only present for CKD or ECKD architected device types)
39–40	<u>CKD and ECKD Only</u> Reserved
37–40	<u>FBA Only</u> Number of FBA blocks used (only present for FBA architected device types)
41–44	Number of 4K pages used (present for all device types)
45–78	Reserved
79	Card generator field (C if Diag 4C has been issued, 0 otherwise)
80	Accounting record identification code (3)

**Accounting Records for Journaling (Record Types 04, 05, 06, 08, and 0I)**

When LOGON, AUTOLOG, XAUTOLOG, LINK, or APPCVM CONNECT journaling is on, CP may create type 04, type 05, type 06, type 08, and type 0I accounting records.

A type 04 accounting record is written whenever CP detects that a user has issued enough LOGON, AUTOLOG, XAUTOLOG, or APPCVM CONNECT invocations with an invalid password to reach or exceed an installation-defined threshold value. This record has the following format:

<b>Column</b>	<b>Contents</b>
1–8	User ID specified on the command
9–16	Reserved for IBM use
17–28	Date and time of accounting ( <i>mmddyymmss</i> )
29–32	Terminal address (see Note 1 on page 303)
33–40	Invalid password (see Note 2 on page 303)
41–48	User ID that entered AUTOLOG, XAUTOLOG, APPCVM CONNECT, or the BYUSER ID that entered LOGON (see Note 4 on page 303).
49–51	Reserved for IBM use
52–53	Current invalid password count in hexadecimal
54–55	Accounting record limit in hexadecimal
56	Blank
57–62	Reserved
63–70	Network qualifier for SNA terminal. Host virtual machine name for TCP/IP terminal.
71–78	LUNAME for SNA terminal. IP address for TCP/IP terminal (see Note 5 on page 303).

## Setting Up SVMs

### 79–80 Accounting card identification code (04)

A type 05 accounting record is produced whenever CP detects that a user has successfully entered a LINK command to a minidisk protected by a password and not owned by that user. This record is always produced when an external security manager authorizes the link. This record has the following format:

Column	Contents
1–8	User ID that entered the command
9–16	Account number
17–28	Date and time of accounting ( <i>mmddyymmss</i> )
29–32	Terminal address (see Note 1 on page 303)
33–40	Reserved for IBM use
41–48	User ID of the user that owns the minidisk
49–52	The minidisk address for which the LINK command was entered
53	Type of minidisk linked (see Note 3 on page 303)
54–55	Reserved for IBM use
56–57	Blank
58–62	Reserved
63–70	Network qualifier for SNA terminal. Host virtual machine name for TCP/IP terminal.
71–78	LUNAME for SNA terminal. IP address for TCP/IP terminal (see Note 5 on page 303).
79–80	Accounting card identification code (05)

A type 06 accounting record is produced whenever CP detects that a user has entered enough LINK commands with an invalid password to reach or exceed an installation-defined threshold value. This record has the following format:

Column	Contents
1–8	User ID that entered the command
9–16	Account number
17–28	Date and time of accounting ( <i>mmddyymmss</i> )
29–32	Terminal address (see Note 1 on page 303)
33–40	Invalid password (see Note 2 on page 303)
41–48	User ID of the user that owns the minidisk
49–51	Reserved for IBM use
52–53	Invalid password count in hexadecimal
54–55	Invalid password limit in hexadecimal
56	Blank
57–60	Minidisk address for which the LINK command was entered
61–62	Reserved
63–70	Network qualifier for SNA terminal. Host virtual machine name for TCP/IP terminal.
71–78	LUNAME for SNA terminal. IP address for TCP/IP terminal (see Note 5 on page 303).
79–80	Accounting card identification code (06)

A type 08 record is generated when a user logs off or disconnects or when a SHUTDOWN or FORCE command causes that logged-on user to be forced off the system. Disconnected users do not have another 08 record generated for them if they are forced off. This record has the following format:

Column	Contents
1–8	User ID
9–16	Account number
17–28	Date and time of accounting ( <i>mmddyymmss</i> )

29–48	Reserved
49–56	LUNAME for SNA terminal. IP address for TCP/IP terminal (see Note 5).
57–64	Network qualifier for SNA terminal. Host virtual machine name for TCP/IP terminal.
65–72	Terminal identification (logical device number, real device number, LUNAME for SNA terminal or NONE)
73–78	Reserved
79–80	Accounting record identification code (08)

### Notes:

- For the terminal address, columns 29 through 32 may contain one of the following:
  - NONE—if no terminal is found
  - SNA—if terminal is SNA (LUNAME is in columns 71–78)
  - A real or logical device number in the form *Lnnn*, where *nnn* is the logical device number, for all other cases.
- For the invalid password, columns 33 through 40 may contain one of the following:
  - Incorrect password
  - TOO LONG—if entered password is more than 8 characters.
- For the type of minidisk linked, column 53 may contain one of the following:
  - X'00'—if the link is to a user's minidisk
  - X'10'—if the link is to a full-pack overlay minidisk.
- A by-user is a user who logs on to a virtual machine using the BY operand of the LOGON command. The by-user's own password is used for LOGON authorization checking for the virtual machine, so the invalid password attempts are counted against the by-user ID, not the user ID of the virtual machine.
- IP addresses are normally written in dotted-decimal format (for example, 9.130.58.78). In journal records, each segment of the IP address is converted to a two-digit hexadecimal value. For example, 9 is converted to 09, and 130 is converted to 82. The result is an eight-byte string of four two-digit hexadecimal numbers in character form. So, 9.130.58.78 becomes the character string 09823A4E.

A type 0I record is generated when the user is logged on through a logical device with an associated IPv6 address. This record has the following format:

Column	Contents
1–8	VM user ID
9–16	Account number
17–28	Date and time of accounting ( <i>mmddyymmss</i> )
29–36	Host virtual machine name
37–68	IPv6 address (see note 1)
69–78	Reserved
79–80	Accounting record identification code (0I)

### Notes:

- IPv6 addresses are normally written in hexadecimal text representation.

**Example:** 0123:4567:89AB:CDEF:FEDC:BA98:7654:3210

In type 0I records, each segment of the IP address is converted from binary to character form. The result is a 32-byte string of sixteen two-digit hexadecimal numbers in character form. So, the address shown above becomes 0123456789ABCDEF FEDCBA9876543210.

### Accounting Records for SNA/CCS (Record Type 07)

A type 07 accounting record is produced whenever a user logs off or disconnects from a device controlled by VCNA or VSCS. The record indicates the user's share of the VCNA/VSCS resource used. Columns 1 through 16 and 79 and 80 of this record contain character data. See the *VCNA Installation and Terminal Use Guide* for details of VM/VCNA accounting records and the *ACF/VTAM\* Planning and Installation* book for details of VSCS accounting records.

Column	Contents
1–8	User ID
9–16	Account number
17–78	VSCS/VCNA accounting data
79–80	Accounting record identification code (07)

### Accounting Records for Inter-System Facility for Communications (Record Type 09)

A type 09 accounting record is produced when the ALL option of the ACNT command is entered. Accounting records are generated for all active conversations and all active links.

There are four different categories of ISFC accounting records:

- Initialization accounting records
- Conversation accounting records
- Link statistics accounting records
- Termination accounting records.

ISFC produces the initialization accounting record during its initialization on the z/VM system. This record, along with the termination accounting record indicates the time frame in which ISFC was active on the z/VM system.

The format of the initialization accounting record is:

Column	Contents
1–8	SYSISFC, indicating that this record was created by the z/VM domain controller
9–12	Initialization record identifier, ISFI
13–16	Reserved for IBM use
17–28	Date and time the accounting record is generated
29–78	Reserved for IBM use
79–80	ISFC accounting record identifier

The format of the conversation start accounting record is:

Column	Contents
1–8	SYSISFC, indicating that this record was created by the z/VM domain controller
9–12	Conversation start accounting record identifier, ISFS
13–16	Conversation ID
17–28	Date and time the accounting record is generated
29–36	User ID of the user that initiated the conversation
37–59	Reserved for IBM use
60	Type of name in bytes 61–68. R indicates a global resource. G indicates a gateway name. U indicates a private resource server virtual machine or workstation user ID. I indicates IUCV.



61–68	Resource name, a gateway name, the user ID of the private resource server virtual machine or workstation, or target userid for an IUCV CONNECT.
69–78	Reserved for IBM use
79–80	ISFC accounting record identifier

The format of the conversation active and conversation end accounting records is:

Column	Contents
1–8	SYSISFC, indicating that this record was created by the z/VM domain controller
9–12	Conversation record identifier. ISFA indicates a conversation active accounting record; ISFE, a conversation end accounting record.
13–16	Conversation ID
17–28	Date and time the accounting record is generated
29–32	Number of bytes received from the remote domain controller since the conversation started or since the last conversation active accounting record was issued.
33–36	Number of bytes sent to the remote domain controller since the conversation started or since the last conversation active accounting record was issued.
37–78	Reserved for IBM use.
79–80	ISFC accounting record identifier

The format of the link statistics accounting records is:

Column	Contents
1–8	SYSISFC, indicating that this record was created by the z/VM domain controller
9–12	Link statistics record identifier, ISFL
13–16	Reserved for IBM use
17–28	Date and time the accounting record is generated
29–32	Number of bytes of data received (unsigned binary fullword)
33–36	Number of bytes of data sent since the link came up or since the last accounting record was generated for this link (unsigned binary fullword)
37–40	Unit address of the link
41–44	Number of attention interrupts
45–48	Number of write operations that result in collisions
49–52	Number of successful write operations
53–56	Number of successful read operations
57–78	Reserved for IBM use.
79–80	ISFC accounting record identifier

The format of the termination accounting record is:

Column	Contents
1–8	SYSISFC, indicating that this record was created by the z/VM domain controller
9–12	Termination record identifier, ISFT.
13–16	Reserved for IBM use
17–28	Date and time the accounting record was generated
29–78	Reserved for IBM use
79–80	ISFC accounting record identifier

### Accounting Records for logging changes to a user's privilege (Record Type 0A)

A type 0A accounting record is produced whenever a SET PRIVCLASS command is successfully issued. The record tracks changes to a user's privilege class and their ability to change their privilege class settings.

There are 4 subtypes of this record.

Subtype	Description
L	SET PRIVCLASS LOCK has been issued
U	SET PRIVCLASS UNLOCK has been issued
C	The user's privilege class(es) have been changed via the SET PRIVCLASS command.
R	SET PRIVCLASS RESET has been issued

The format for subtypes 'L' (SET PRIVCLASS LOCK) and 'U' (SET PRIVCLASS UNLOCK) appears below. Columns 1 through 36 and 78 through 80 of this record contain character data; all other data is in hexadecimal form (the hexadecimal data is unprintable).

Column	Contents
1-8	User ID of the issuer of the SET PRIVCLASS command.
9-16	Account number
17-28	Date and time of accounting ( <i>mmddyymmss</i> )
29-36	Target user ID. User ID that is the target of the SET PRIVCLASS lock or unlock
37-77	Reserved
78	Subtype

Character	Subtype Meaning
L	SET PRIVCLASS LOCK has been issued
U	SET PRIVCLASS UNLOCK has been issued

79-80	Accounting record identification code (0A)
-------	--

The format for subtypes 'C' (SET PRIVCLASS change) and 'R' (SET PRIVCLASS RESET) appears below. Columns 1 through 36 and 78 through 80 of this record contain character data; all other data is in hexadecimal form (the hexadecimal data is unprintable). The privilege class information consists of bits representing privilege classes A-Z, and 1-6 respectively. The bit definitions are defined in HCPCLASS COPY.

Column	Contents
1-8	User ID of the issuer of the SET PRIVCLASS command
9-16	Account number
17-28	Date and time of accounting ( <i>mmddyymmss</i> )
29-36	Target user ID. User ID whose settings are being changed
37-40	User privilege classes before the change or reset
41-44	User privilege classes after the change or reset
45-48	User privilege classes as indicated in the directory
49-77	Reserved
78	Subtype

Character	Subtype Meaning
C	The User's privilege class has been changed
R	The User's privilege class has been reset

79-80	Accounting record identification code (0A)
-------	--

## Accounting Records for virtual disk in storage space (Record Type B)

A type B accounting record is produced for a virtual disk in storage. Columns 1 through 28 and 79 and 80 of this record contain character data; all other data is in hexadecimal form (the hexadecimal data is unprintable). Translation codes for columns 33 through 36 can be found in Appendix E, “Device Class and Type Codes,” on page 773. See the documentation for each specific device for more complete information on model numbers.

Column	Contents
1–8	User ID. This user ID is defined as follows: <ul style="list-style-type: none"> <li>• If the virtual disk in storage is defined in the directory, this is the user ID which contains the MDISK definition for this virtual disk in storage.</li> <li>• If the virtual disk in storage was defined using the CP DEFINE command, this is the user ID that issued the DEFINE command.</li> </ul>
9–16	Account number
17–28	Date and time of accounting ( <i>mmddyymmss</i> )
29–32	Number of seconds connected to CP
33	Device class
34	Device type
35	Device model (if any)
36	Device features (if any)
37–40	Number of FBA blocks used
41–44	Number of 4K pages used (present for all device types)
45	Sub-type of virtual disk accounting record. The only volume defined is zero. If non-zero, the contents of bytes 1-44 and 46-78 are undefined.
46–78	Reserved
79	Card generator field (C if Diag 4C has been issued, 0 otherwise)
80	Accounting record identification code (B)

## Accounting Records Network Data Transmissions (Record Type C)

Type C accounting records may be produced for any virtual machine user with NETAccounting or NETRouter specified as an option in its user directory. The account records are produced when accounting is performed (CP ACNT command), or when a field in the accounting record is about to overflow, and the user has sent or received data from a Network device. These records are only produced when fields containing byte counters for a device are non-zero.

Network devices to be included in these counts are:

- Virtual Network Interface Cards (NIC)
- Virtual Channel to Channel Adapters
- IUCV and APPC/VM connection paths

The contents of Network Data Transmission accounting records depends on the type of Network device being used. There are 3 different formats, designated by a subtype in byte 78. For subtype 00 (Virtual NIC records), an additional 4 subformats, designated in byte 77, indicate what type of network data is counted in the record.

Type C account record describing data transfer involving routers through a Virtual NIC (Network device type 00). Columns 1 through 28, 35 through 50, and 79 through 80 of this record contain character data; all other data is in hexadecimal form (the hexadecimal data is unprintable.)

## Setting Up SVMs

Column	Contents
1-8	VM Userid (Owner of the Virtual NIC)
9-16	Account number
17-28	Date and time of accounting (mmddyymmss)
29-30	Base VDEV address of the Virtual NIC.
31-34	Default IP Address for this network adapter.
35-42	LAN or Virtual Switch Owner
43-50	LAN Name or Virtual Switch Name
51-58	Bytes Sent (See Data Descriptor).
59-66	Bytes Received (See Data Descriptor).
67-76	Reserved
77	Data Descriptor:  00 - Bytes sent to or received from Routers 01 - Bytes sent to or received from non-Routers 02 - Bytes sent to or received via Broadcast 03 - Bytes sent to or received via Multicast
78	Network device type (00) Virtual NIC
79	Card generator field (C if Diag 4C has been issued, 0 otherwise)
80	Accounting record identification code (C)

### Notes:

1. Hardware headers which appear before the IP Header are not included in the byte counts reported for simulated network adapters.
2. IP addresses are normally written in dotted-decimal format (for example, 9.130.58.78). In journal records, each segment of the IP address is converted to a two-digit hexadecimal value. For example, 9 is converted to 09, and 130 is converted to 82. The result is an eight-byte string of four two-digit hexadecimal numbers in character form. So, 9.130.58.78 becomes the character string 09823A4E.

TYPE C accounting record for data transfer with a Virtual Channel to Channel Adapter. Columns 1 through 28, 35 through 42, and 79 through 80 of this record contain character data; all other data is in hexadecimal form (the hexadecimal data is unprintable.)

Column	Contents
1-8	VM Userid (owner of the local CTCA)
9-16	Account number
17-28	Date and time of accounting (mmddyymmss)
29-30	VDEV address of the local CTCA
31-34	Reserved
35-42	Userid of the owner of the remote CTCA
43-44	VDEV address of the remote CTCA
45-50	Reserved
51-58	Number of bytes sent to the remote CTCA.

<b>59-66</b>	Number of bytes received from the remote CTCA.
<b>67-77</b>	Reserved
<b>78</b>	Network device type (01) virtual CTCA
<b>79</b>	Card generator field (C if Diag 4C has been issued, 0 otherwise)
<b>80</b>	Accounting record identification code (C)

TYPE C account record for data transfer with an IUCV or APPC/VM connection. Columns 1 through 28, 35 through 42, and 79 through 80 of this record contain character data; all other data is in hexadecimal form (the hexadecimal data is unprintable.)

<b>Column</b>	<b>Contents</b>
<b>1-8</b>	VM Userid (Owner of the local connection.)
<b>9-16</b>	Account number
<b>17-28</b>	Date and time of accounting (mmddyymmss)
<b>29-34</b>	Reserved
<b>35-42</b>	Connected VM Userid (remote)
<b>43-50</b>	Reserved
<b>51-58</b>	Number of bytes sent to the remote user.
<b>59-66</b>	Number of bytes received from the remote user.
<b>67-77</b>	Reserved
<b>78</b>	Network device type (02) IUCV or APPC path
<b>79</b>	Card generator field (C if Diag 4C has been issued, 0 otherwise)
<b>80</b>	Accounting record identification code (C)

**Notes:**

1. Byte count values are reset to zero after an accounting record is created.
2. Byte count values represent data read or written to the device. They will not typically contain bytes transferred as a result of initialization, termination, or error recovery.
3. Accounting for a particular LAN may be controlled by the DEFINE LAN, and SET LAN commands, and default settings may be set by the VMLAN command or VMLAN statement in the system config file.

### **Accounting Records for CPU Capability (Record Type D)**

A Type D accounting record is produced to record the CPU capability of the processors in the system. A record is generated during VM system initialization and whenever the CPU capability changes.

All fields of the record contain character data.

<b>Column</b>	<b>Contents</b>
<b>1-16</b>	Reserved
<b>17-28</b>	Date and time of accounting (mmddyymmss)
<b>29-36</b>	CPU capability
<b>37-38</b>	Reserved
<b>79</b>	Card generator field (0)

## Adding Your Own Accounting Records and Source Code

CP allows you to customize the way it collects accounting records in two ways:

- A virtual machine can use the DIAGNOSE instruction to initiate the generation of a virtual machine accounting record.
- You can add your own source code to the CP accounting exit module.

### User-Initiated Accounting Records (Record Type C0)

A virtual machine user can initiate the creation of an accounting record that contains up to 70 bytes of information of the user's choosing. To do this, the user enters a DIAGNOSE code X'4C' instruction with the following operands (the *z/VM: CP Programming Services* book describes how to enter a DIAGNOSE code):

- The address of a data area in virtual storage that contains the information that the user wants in columns 9 through 78 of the card image record. (This information is placed in the accounting record exactly as it appears in the data area. If more than 70 bytes of data are included, only the first 70 bytes appear in the accounting record.)
- A function code of X'10'.
- The length of the data area in bytes.

The information on this type of accounting record is as follows:

Column	Contents
1–8	User ID
9–78	User-formatted data
79–80	Accounting record identification code (C0)

**PI** end

File spool service virtual machines can generate accounting records in this category. For details, see the *z/VM: CMS File Pool Planning, Administration, and Operation* book.

### CP Accounting Exit (Module HCPACU)

**PI**

You can make system-wide changes to the accounting records that CP collects by adding source code to module HCPACU. HCPACU is an installation-wide exit.

HCPACU supports installation-supplied code for the IBM 3277 Operator Identification Card Reader feature and for examining and modifying CP accounting records. HCPACU is included in the CP nucleus at system generation. Later, during system operation, the CP accounting function calls HCPACU whenever any of the following occurs:

- A virtual machine is started or stopped
- Accounting commands are entered by an operator or a virtual machine
- CP does checkpointing operations.

However, if you do not add any code to HCPACU, the module performs no meaningful function.

HCPACU contains two entry points, HCPACUON and HCPACUOF. Figure 4 provides an overview of the IBM-supplied HCPACU module, showing the entry points and the areas where you can add your own source code.

```

          COPY  HCPLOPTNS
HCPACU   HCPPROLG ATTR=(RESIDENT),BASE=(R12)
*
          COPY  AOFPARAM           parameter list for HCPACUOF
          COPY  AONPARAM           parameter list for HCPACUON
          COPY  HCPEQUAT           common system equates
          COPY  HCPPFXPG           prefix page
          COPY  HCPSAVBK           savearea block
*
          HCPUSING PFXPG,0
*
HCPACUON HCPENTER CALL,SAVE=DYNAMIC
          HCPUSING AONPARAM,R1
*****
*
* Installation-supplied code can be inserted here.
*
*****
          HCPDROP R1
          SLR   R15,R15           Set return code of 0
ACUONEXT DS    0H
          ST   R15,SAVER15       Store return code
          L    R11,SAVER11       Restore register 11
          HCPEXIT EP=HCPACUON
          HCPDROP R13
*
HCPACUOF HCPENTER CALL,SAVE=PFXBALSV
          HCPUSING AOFPARAM,R1
          LA   R13,PFXBALSV      Establish savearea address
          HCPUSING SAVBK,R13
*****
*
* Installation-supplied code can be inserted here.
*
*****
          HCPDROP R1
          SLR   R15,R15           Set return code of 0
ACUOFEXT DS    0H
          ST   R15,SAVER15       Store return code
*          L    R11,SAVER11       HCPEXIT does actual load
          HCPEXIT EP=HCPACUOF
          HCPDROP R0,R13
*
          HCPEPILG

```

Figure 4. Module HCPACU Overview

HCPACU has the following requirements and restrictions:

1. HCPACU is written in Assembler H coding language.
2. HCPPROLG, HCPUSING, HCPENTER, HCPDROP, HCPEXIT, and HCPEPILG are IBM-supplied macros (in HCPPSI MACLIB) that must be preserved in HCPACU and used only as shown.
3. The attributes of HCPACU must be REENTRANT and RESIDENT, because HCPACU is called during CP initialization and during CP checkpointing operations.
4. Links and accesses from HCPACU to other modules, data areas, and control blocks are not supported.

## Setting Up SVMs

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 both HCPACUON and HCPACUOF, register 13 points to a save area mapped by HCPSAVBK, shown in Figure 5. 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.

SAVBK	DSECT		
	DS	6F	Reserved for IBM use
SAVEREGS	DS	0XL64	CALLER'S REGISTERS - R0 to R15
SAVER0	DS	F	CALLER'S SAVED REGISTER 0
SAVER1	DS	F	CALLER'S SAVED REGISTER 1
SAVER2	DS	F	CALLER'S SAVED REGISTER 2
SAVER3	DS	F	CALLER'S SAVED REGISTER 3
SAVER4	DS	F	CALLER'S SAVED REGISTER 4
SAVER5	DS	F	CALLER'S SAVED REGISTER 5
SAVER6	DS	F	CALLER'S SAVED REGISTER 6
SAVER7	DS	F	CALLER'S SAVED REGISTER 7
SAVER8	DS	F	CALLER'S SAVED REGISTER 8
SAVER9	DS	F	CALLER'S SAVED REGISTER 9
SAVER10	DS	F	CALLER'S SAVED REGISTER 10
SAVER11	DS	F	CALLER'S SAVED REGISTER 11
SAVER12	DS	F	CALLER'S SAVED REGISTER 12
SAVER13	DS	F	CALLER'S SAVED REGISTER 13
SAVER14	DS	F	CALLER'S SAVED REGISTER 14
SAVER15	DS	F	CALLER'S SAVED REGISTER 15
SAVEWRK	DS	0XL40	WORKAREA FOR CALLEE
SAVEWRK0	DS	F	WORKAREA FOR CALLEE
SAVEWRK1	DS	F	WORKAREA FOR CALLEE
SAVEWRK2	DS	F	WORKAREA FOR CALLEE
SAVEWRK3	DS	F	WORKAREA FOR CALLEE
SAVEWRK4	DS	F	WORKAREA FOR CALLEE
SAVEWRK5	DS	F	WORKAREA FOR CALLEE
SAVEWRK6	DS	F	WORKAREA FOR CALLEE
SAVEWRK7	DS	F	WORKAREA FOR CALLEE
SAVEWRK8	DS	F	WORKAREA FOR CALLEE
SAVEWRK9	DS	F	WORKAREA FOR CALLEE

Figure 5. HCPSAVBK Save Area

**Entry Point HCPACUON:** Entry point HCPACUON supports installation-supplied code for the IBM 3277 Operator Identification Card Reader feature. You can add code to examine or validate the data read from an identification card and to accept or reject operator connection to the system.

HCPACUON is called when a LOGON command is entered from a virtual machine, or during autolog processing.

The CP accounting function uses general-purpose register 1 for input to entry point HCPACUON. Register 1 contains the address of a parameter list, called AONPARM, that is described in Figure 6 on page 313. The DSECT for this parameter list must be preserved as shown.



```

AONPARM DSECT ,
****   AONPARM -
*
*
*   0   [-----] AONBUFF
*
*   8
*
****   AONPARM -
*****
*
*   Parameters for IBM 3277 Operator Identification Card
*   Reader Feature.
*
*****
AONBUFF DC   F'0'           Contains one of the following:
*
*                               - Address of a buffer containing
*                               up to 130 bytes of data that
*                               has been read from the IBM 3277
*                               Operator Identification Card
*                               Reader Feature. The data is
*                               user-defined for terminal
*                               operator identification.
*
*                               - 0, if the reader feature was
*                               not available
*
*                               - 4, if an unsuccessful attempt
*                               to read this reader feature
*                               was made.
*
AONSIZE EQU   (*-AONPARM+7)/8   AONPARM size in doublewords

```

Figure 6. AONPARM Parameter List for Entry Point HCPACUON

Upon return to the CP accounting function from entry point HCPACUON, general-purpose register 15 must contain one of the following return codes:

Value	Meaning
X'00000000'	Normal processing continues.
X'00000004'	The terminal operator should be forced off the system. However, if the terminal operator was automatically logged on during initialization, you cannot force the operator off the system.

**Entry Point HCPACUOF:** HCPACUOF supports the addition of installation-supplied code for examining and modifying accounting records. HCPACUOF is called when any of the following occurs:

- A virtual machine operator or system operator enters a LOGOFF or FORCE command
- The system operator enters an ACNT or SET ACCOUNT command
- CP starts the checkpointing function
- A virtual machine issues DIAGNOSE code X'4C' to create a user-initiated accounting record.

For the first three conditions listed above, CP generates accounting record type 1, 2, 3, B, or C. For the last condition, CP generates type C0 accounting records. HCPACUOF is not called during the generation of other types of accounting records.

## Setting Up SVMs

The CP accounting function uses general-purpose register 1 for input to entry point HCPACUOF. Register 1 contains the address of a parameter list, called AOFPARAM, that is described in Figure 7. The DSECT for this parameter list must be preserved as shown.

```

AOFPARAM DSECT ,
****    AOFPARAM -
*
*
*      0      [-----] AOFAREC [-----]
*
*      8
*
****    AOFPARAM -
*****
*
*    ACCOUNTING RECORD.
*
*****
AOFAREC DC    F'0'          Contains the address of a
*
*                                completed accounting record.
AOFSIZE EQU    (*-AOFPARAM+7)/8    AOFPARAM size in doublewords

```

Figure 7. AOFPARAM Parameter List for Entry Point HCPACUOF

Upon return to the CP accounting function from entry point HCPACUOF, general-purpose register 15 must contain following return code:

Value	Meaning
X'00000000'	Normal processing continues

**PI end**

## 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 tapes or CD-ROM include:

- A sample directory entry for one error recording virtual machine.
- A sample system configuration file that defines the user ID for an error recording virtual machine as OPEREREP. The user ID for an error recording virtual machine is defined as part of the SYSTEM\_USERIDS statement in the system configuration file. Records are queued for it and it is automatically logged on by CP initialization.
- A sample PROFILE EXEC for one error recording virtual machine.

**Note:** The sample directory entry for an error recording virtual machine (OPEREREP) supplied on the z/VM System DDR tapes or CD-ROM contains the required IUCV authorization to connect to the CP \*LOGREC system service.

To set up a virtual machine to begin error recording automatically, you must have the proper PROFILE EXEC and user directory set up. The following steps show this procedure:

1. **Log on as MAINT.** If you did this step as part of installation, you may already be logged on as MAINT, or the user ID you are installing with on your current operating system.
2. **Determine the write password of the error recording virtual machine's 191 disk.** Before you can link to the error recording virtual machine's 191 disk, you need to know its write password. To find out the write password, examine the user directory entry of the error recording virtual machine. (As MAINT, you usually have access to the user directory.)  
If the error recording virtual machine's 191 disk does not have a write password, you must supply one and update the directory.

**Note:** Verify that the directory entry for this virtual machine contains the required IUCV authorization for connecting to the CP error recording system service. For example, IUCV \*LOGREC.

3. **Link to the error recording virtual machine's 191 disk.** To do this, enter:

```
link to opererep 191 as 391 wr
```

When CP responds with ENTER WRITE PASSWORD: enter:

```
wpass
```

where *wpass* is the write password of the error recording virtual machine.

4. **Access the 391 disk.** To do this, enter:

```
access 391 x
```

If you receive a message that says X'391' DEVICE ERROR, you must format the 391 disk. To do this, enter:

```
format 391 x
```

CMS responds with FORMAT WILL ERASE ALL FILES ON DISK X (391). DO YOU WISH TO CONTINUE? (YES|NO). You should enter:

```
yes
```

CP responds with ENTER DISK LABEL. You should enter:

```
erep
```

or any 1- to 6-character label name.

5. **Copy a file named DVM PROFILE from MAINT's 191 disk to the 391 disk.** (DVM PROFILE is the PROFILE EXEC for the accounting and error recording virtual machines). The RETRIEVE utility, which does the IUCV connect to the \*LOGREC system service, is invoked from this PROFILE EXEC. To do this, enter:

```
copyfile dvm profile a profile exec x
```

6. **Release and detach the 391 disk.** To do this, enter:

```
release x (det
```

7. **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 *opererep* user ID, enter:

```
xautolog opererep
```

---

## 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:

## Setting Up SVMs

- At CP generation time in the system configuration file, code the SYSTEM\_USERIDS statement or the SYSEREP macroinstruction to specify the virtual machine IDs (up to two) for which the CP accounting system service (\*LOGREC) is to accumulate records. If these users have an entry in the user directory, they will be logged on as part of CP initialization. See “SYSTEM\_USERIDS Statement” on page 232 for information about the SYSTEM\_USERIDS statement or “SYSACNT (Optional)” on page 661 for information about the SYSEREP macroinstruction.
- Additional virtual machines can also collect error recording data by following the instructions in sections:
  - See “Specifying a New EREP Virtual Machine.”
  - See “Starting EREP Record Retrieval Manually.”

## 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 directory entry of the user entering the RETRIEVE utility must have an IUCV directory control 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.”

## 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’s 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. Then erase the old files to free space on the A disk.

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 CP 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 CP 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 CP 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 the SYSEREP macroinstruction is omitted and no user ID for EREP is specified in the system configuration file.

---

### 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 control statement in its user directory entry 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 DDR tapes or CD-ROM include:

- A sample directory entry 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.

To set up a virtual machine so that it automatically begins symptom record recording, you must have the proper PROFILE EXEC and user directory set up. The following steps show this procedure:

1. **Log on as MAINT.** If you did this step as part of installation, you may already be logged on as MAINT, or the user ID you are installing with on your current operating system.
2. **Determine the write password of the symptom record recording virtual machine's 191 disk.** Before you can link to the symptom record recording virtual machine's 191 disk, you need to know its write password. To find out the write password, examine the user directory entry of the symptom record recording virtual machine. (As MAINT, you usually have access to the user directory.)

If the symptom record recording virtual machine's 191 disk does not have a write password, you must supply one and update the directory.

**Note:** Verify that the directory entry for this virtual machine contains the required IUCV authorization for connecting to the CP accounting system service. For example, IUCV \*SYMPTOM.

3. **Link to the symptom record recording virtual machine's 191 disk.** To do this, enter:

```
link to opersymp 191 as 391 wr
```

When CP responds with ENTER WRITE PASSWORD: enter:

```
wpass
```

where *wpass* is the write password of the symptom record recording virtual machine.

4. **Access the 391 disk.** To do this, enter:
 

```
access 391 x
```

If you receive a message that says X'391' DEVICE ERROR, you must format the 391 disk. To do this, enter:

```
format 391 x
```

CMS responds with `FORMAT WILL ERASE ALL FILES ON DISK X (391). DO YOU WISH TO CONTINUE? (YES|NO)`. You should enter:

```
yes
```

CP responds with `ENTER DISK LABEL`. You should enter:

```
symptom
```

or any 1- to 6-character label name.
5. **Copy a file named DVM PROFILE from MAINT's 191 disk to the 391 disk.** (DVM PROFILE is the PROFILE EXEC for the accounting, error recording, and symptom record recording virtual machines.) The RETRIEVE utility, which does the IUCV connect to the \*SYMPTOM system service, is invoked from this PROFILE EXEC. To do this, enter:
 

```
copyfile dvm profile a profile exec x
```
6. **Release and detach the 391 disk.** To do this, enter:
 

```
release x (det
```
7. **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 or the `SYSSYMP` macroinstruction 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 an entry in the user directory, they will be logged on as part of CP initialization. See “SYSTEM\_USERIDS Statement” on page 232 for information about the `SYSTEM_USERIDS` statement or “SYSACNT (Optional)” on page 661 for information about the `SYSSYMP` macroinstruction.
- Additional virtual machines can also collect symptom records by following the instructions in sections:
  - See “Specifying a New Symptom Record Recording Virtual Machine.”
  - See “Starting Manual Retrieval of Symptom Records” on page 320.

### 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.

## Setting Up SVMs

- 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 control 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.”

## 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 CP 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 CP 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 CP 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 the SYSSYMP macroinstruction is omitted and 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 3705, 3720, or 3725 communication controller, set up a special service virtual machine. Unlike other service virtual machines such as the accounting virtual machine, the z/VM System DDR tapes or CD-ROM do not include a sample directory entry 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 the *z/VM: CP Commands and Utilities Reference*.

## Setting Up SVMs

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 directory entry for each communication controller. Use the directory entry in Figure 8 on page 323 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 3380 900 40 XADISK MR
** Provides space for dumps of the communication controller.

```

Figure 8. Sample Directory Entry for a Communication Controller

2. Set up a PROFILE EXEC for the EP0F7 user ID. Use the sample in Figure 9 as a guideline.

```

access 343 b/a
** Access the ACF/SSP disk.
**
queue 'RUN0F7'
** Stack a command to execute the CCLOAD command.
**

```

Figure 9. 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(msg 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.

## Setting Up SVMs

The PVM configuration file shown in Figure 10 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 10. Sample PVM Configuration File*

PROFILE PVM additions to use this support are shown in Figure 11.

```
* 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 11. PROFILE PVM Additions*

The EP0F7 PVM file shown in Figure 12 on page 325 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 12. Sample EP0F7 File

When the 37xx 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 37xx 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:

```

HCPATRI22E 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 the *z/VM: Connectivity* book. The application that controls the service pool is called the service pool manager. The

## Setting Up SVMs

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 user directory control statement. This assigns a directory entry for each virtual machine to be included in the pool. See “POOL Directory Control Statement (General)” on page 540 for information on coding the POOL user directory control statement. If you code the POOL statement in a user’s directory entry, it is assigned the only control statement other than a USER statement in that user’s directory entry. 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 user directory control statements that apply to the range of virtual machines in the pool in a profile entry in the directory (see “PROFILE Directory Control Statement (Control)” on page 551). 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 Print Services Facility/VM Virtual Machines

The Print Services Facility/VM (PSF/VM) licensed program processes the spooled output for advanced function printers. It also uses defaults and options specified in a PSF command and CP SPOOL command to generate printer commands to produce formatted output. PSF/VM supports the following printers:

- 3800 Printing Subsystem Models 3 and 8
- 3812 Page Printer Model 2
- 3820 Page Printer
- 3827 Page Printer
- 3835 Page Printer
- 4224 Printer.

PSF/VM is composed of the following components:

- The PSF command, which places a print file and resource files in a printer spool file for a page printer and places the processing options in an external attribute buffer (XAB).
- The spool file conversion machine (SFCM), which converts the spooled print files into the relevant printer command stream for a specified type of printer driver machine (PDM) (for example, a 3800 PDM or a 3820 PDM) and builds a set of control files for use during processing of the print file.
- The 3800 PDM, which drives a 3800 using the printer command stream and control files created by the SFCM.
- The 3820 PDM, which processes files either for a 3820 or for one or more printers attached through RSCS. For the 3820, the 3820 PDM drives the printer using the printer command stream created by the SFCM.
- The Group3 PDM, which processes files for a Group3 printer using the printer command stream and control files created by the SFCM.

You must define one virtual machine for the SFCM and for each PDM. Depending on the requirements of your installation, you might define a virtual machine owned by the system programmer for maintaining PSF/VM.

For more information about PSF/VM and how to set up the virtual machines required for this product, see the *Print Services Facility/VM System Programmer's Guide*.

---

## 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 ACL 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 user directory control 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 387. In addition, further information on access modes can be found in “LINK Directory Control Statement (Device)” on page 500 and “MDISK Directory Control Statement (Device)” on page 513. 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 directory entry for the virtual machine that contains the VSM:

- Must authorize the VSM to use IUCV to communicate with SNA/CCS using the IUCV directory control 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 control 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 control statement.
- May specify that a named saved system or device is to be loaded when the virtual machine is logged on using the IPL control statement.
- May specify a secondary user ID on the CONSOLE directory control 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 279; for details, see “CHOOSE\_LOGO Statement” on page 285.

### 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.

**DSPACE**

is the pacing value to be used with display terminals.

**KSPACE**

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 page 555. 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. Setting Up Cross System Extensions (CSE)

This chapter contains an overview of cross system extensions (CSE) and tells how to:

- Plan for CSE
- Enable CSE
- Administer CSE
- Define a virtual CSE system to be used during hardware outages.

---

### Overview

VM provides programming support for cross system extension (CSE). With CSE, end users can participate in a multisystem environment. This environment can include up to four z/VM systems connected in a complex. Systems running different releases of z/VM may be connected in the same CSE complex.

Figure 13 depicts an example of a CSE complex consisting of four z/VM systems.

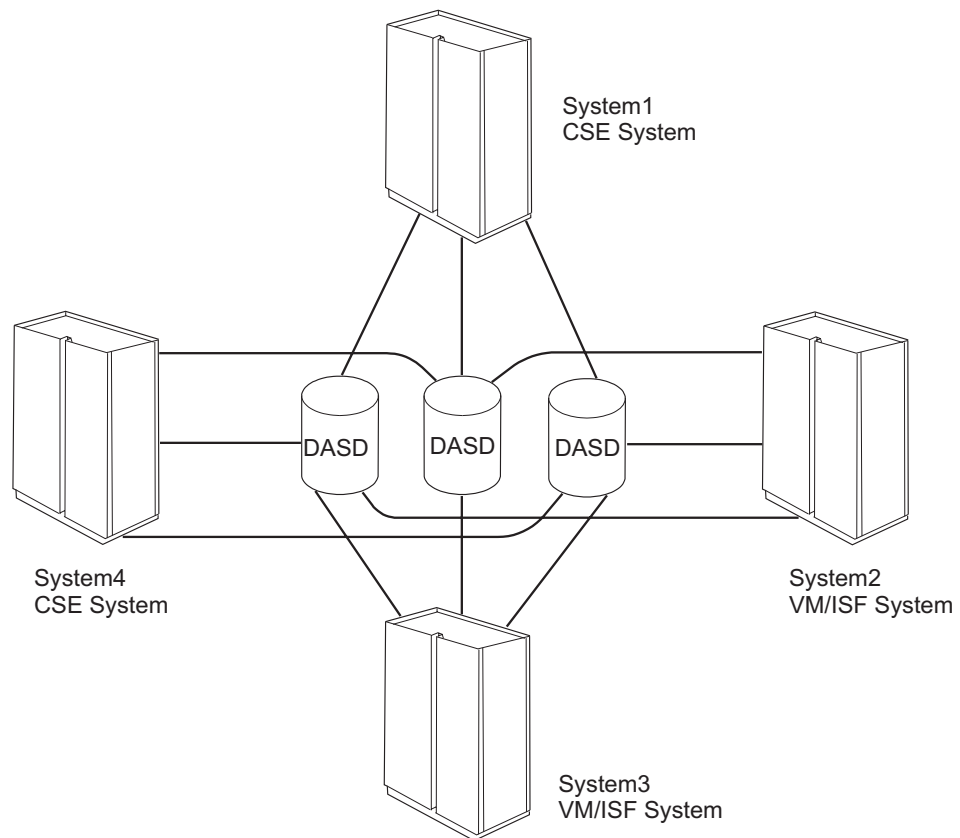


Figure 13. A Four-System CSE Complex

### CSE Capabilities

CSE consists of three related, but independent, parts:

- Cross system link

This facility extends CP link protocols across the CSE complex. It controls normal, stable, and exclusive read or read/write access across multiple VM

## CSE Overview

systems for minidisks on shared count-key-data direct access storage devices (CKD DASD), including those devices that accept the extended-count-key-data (ECKD) channel program protocol.

- Cross system spool  
This facility extends CP spooling capabilities. It contains and manages spool files for users across multiple VM systems by using shared DASD and a communication vehicle. The communication vehicle is the Virtual Machine/Pass-Through Facility with modifications. You can set up your system to have cross system linking without cross system spooling.
- Cross system MESSAGE and QUERY commands.  
With CSE, MESSAGE and QUERY command support is extended to allow their use across multiple VM systems. Refer to “Supported Features” on page 335 for features of z/VM that are supported across system boundaries.

Cross system spool is supported among all CSE systems. The maximum number of systems which may be coupled using CSE functions depends on the specific functions being used:

- Shared spool: 4
- Shared minidisks: up to 56, depending on the physical devices
- Shared source directory: 4

## Users Supported

A CSE complex can support 95% of the total number of users that could be supported by each system without CSE. For example, if each system could support 1000 users, a four-system CSE complex could support 3800 users, as shown by the equation:

$$\begin{array}{ccccccc} 1000 & \times & .95 & \times & 4 & = & 3800 \\ \text{(users)} & & \text{(percent)} & & \text{(systems)} & & \text{(users supported)} \end{array}$$

For general users, this CSE complex functions as a single larger system. For system operators, programmers, and administrators, the single image is less pronounced. These users tend to view the CSE complex as up to four systems that must be managed as before.

## Setting Up a CSE Complex

Setting up a CSE complex consists of:

- Providing the hardware connections between the processors
- Installing the IBM VM/Pass-Through Facility licensed program
- Configuring z/VM and the supporting programs with the information they need to run the CSE complex.

---

## Planning for CSE

Before you install CSE, you need to identify the requirements unique to your installation in a detailed plan.

## Requirements

Using CSE requires the following:

- A single source directory. A single source directory requires z/VM Directory Maintenance Facility (DirMaint) to synchronize the user directories of all systems in the CSE complex. You may, if you wish, choose another means of creating a single source directory, but that is outside the scope of CSE.

- A single version of DIRECTXA. The most current version of DIRECTXA must be used on all systems in the CSE complex.
- Multiple z/VM systems.
- The systems in the complex must be connected by means of channel-to-channel adapters (CTCA) or an IBM 3088 Multisystem Channel Communications Unit. The VM/Pass-Through Facility virtual machines used by CSE communicate over these connections.
- The systems must share CKD-formatted DASDs. Fixed block architecture (FBA) devices are not supported by the cross system link facility.
- The same release of the IBM VM/Pass-Through Facility (Program Number 5684-100) licensed program must be installed on all systems in the complex. The VM/Pass-Through Facility is needed to support cross system spool and CSE message support among all systems.

You need detailed information about z/VM. (See the “Bibliography” on page 787 for a list of z/VM books.)

## Restrictions

System paging space and minidisks cannot reside on the same DASD device. This will avoid a possible deadlock situation.

An installation should not define users with a password of LBYONLY, unless all systems in the CSE complex support the LBYONLY operand of the USER directory control statement. For more information about this function, see the “USER Directory Control Statement (Control)” on page 572 and the LOGON command in the *z/VM: CP Commands and Utilities Reference*.

## Supported Features

CSE supports the following features of z/VM across system boundaries:

- LINK command for count-key-data (CKD) DASD, including those DASD that accept the extended-count-key-data (ECKD) channel program protocol
- Spool files
- Special message function (SMSG)
- Message (MSG, MSGNOH) and warning (WNG) commands
- RACF<sup>(™)</sup> licensed program for VM

**Note:** If the RACF licensed program is on one system in the complex, it must be on all systems in the complex.

- QUERY commands
  - QUERY LINKS
  - QUERY NAMES
  - QUERY USERS
  - QUERY USERS *userid*
- DIRMAINT support for general users
- Other commands
  - CHANGE
  - INDICATE LOAD
  - ORDER
  - PURGE
  - TAG

## Planning CSE

- TRANSFER.

Some features of z/VM, although unchanged within any system in the CSE complex, are *not* supported across system boundaries by CSE. These are:

- Virtual machine communication facility (VMCF)
- Inter-user communication vehicle (IUCV)
  - Monitoring data collectors
  - Accounting
- Virtual reserve/release
- System data files
- Virtual disks in storage.

Applications that contain the Transparent Services Access Facility (TSAF) can use TSAF functions within a CSE complex.

**Note:** The VMUDQ macro, used to query the CP user directory, examines only the object directory of the system on which it is being run. For DASD that are not totally shared and globally defined among member systems of a CSE complex, you must run VMUDQ on each of those systems to obtain a complete picture of the target volume. For more information about the VMUDQ macro, see the *z/VM: CP Programming Services* book.

## Sharing DASD Volumes

Device I/O activity may increase significantly when DASD are shared among multiple VM systems. Devices can serve only one system at any given instant. If two systems require service at the same time, one system will find the device busy and must wait. The length of this wait depends upon the type of I/O request already in progress. If the first I/O is limited to a single track, as are most read and write operations on behalf of the CMS file system, the wait is not long. If the I/O potentially affects all tracks on the same cylinder, for such activity as dumping, restoring, CMS and CP formatting, or CP paging and spooling, the time spent waiting can seriously affect the performance of the second system.

Furthermore, if additional work is queued for the device in the first system (for example, during periods of heavy CP paging and spooling loads), when the current I/O operation ends, both systems race to engage the device and start the next operation. The second system is at a disadvantage in this race because it must execute a longer code path. That is, fewer instructions are required to redrive a device with an already-queued I/O request after the operation in progress terminates than are required to recognize that the busy device is now free and that another I/O operation can be started for it.

An additional complication arises because DASD control units do not, in general, respond to each of their channel interfaces in a round-robin or fair-share manner. Thus, the system connected to the highest-priority channel interface of the control unit receives more service than any other during periods of high contention. In extreme cases, this can result in a lockout.

Because of these considerations, you should follow these general guidelines:

- Assign storage devices, control units, and channel paths to avoid unnecessary CSE I/O contention.



- Cable the fastest system to the lowest priority port and the slowest system to the highest priority port on the control units when DASD is shared among systems of unequal power.
- Check that all devices are online after the system is IPLed.

### Avoiding Shared DASD I/O Bottlenecks

Since multiple systems can now cause I/O operations on the same devices and control units, bottlenecks that did not exist (or that were not obvious) in a single system can cause serious performance complications in a CSE environment. Careful planning will eliminate unnecessary performance problems by preventing undue I/O interference caused by CSE activity. In general, the aim of such planning is to isolate I/O operations by type and by system.

The kinds of shared DASD that you should consider when planning for CSE are DASD for:

- Shared spool files
- Backup and recovery
- Shared access to minidisks.

Each of these has somewhat different characteristics, which are explained as follows.

**DASD for Shared Spool Files:** For shared access to spool files, all CP-owned DASD on which spooling space has been allocated must be physically and logically shared by all systems in the complex. Proper cabling of channels and control units creates *physical sharing*. The volume ID of each spooling DASD in each system's list of CP-owned DASD (created by the CP\_OWNED statement in the system configuration file) creates *logical sharing*.

**Performance Considerations:** To reduce the performance impact of DASD sharing for spool files:

- Isolate by system—Do not intermix the CP-owned spooling DASD of different systems on the same string, control unit, or channel path. If this rule is followed, systems will not compete with each other when writing, checkpointing, or purging spool files. CSE contention is possible only when systems are reading spool files other than their own.
- Isolate by type—Do not mix DASD spooling space with other high-activity DASD space, either on the same device or on the same control unit. Paging space, temporary disk space, and frequently used CMS minidisks should not be allowed to contend with spooling.
- Review how you have organized your CP-owned DASD.

Each system has its own spooling volumes. In the list of each system's CP-owned volumes, you must include the volumes used for spooling in the other systems, but you must designate them as shared volumes.

Use the SHARED option on CP\_OWNED statements in your system configuration file. You do not have to list the volumes in any particular order as long as you use the SLOT operand to give the same slot number to the same volume on each system.

**DASD for Backup and Recovery:** When one system in the CSE complex is down for scheduled or unscheduled maintenance, spool files created on that system can be made available to users on other systems by operating CSE in a virtual machine. Refer to the *z/VM: System Operation* book.

For this to be effective, the spooling and paging DASD owned by the system must be available to at least one other system in the complex. Spooling DASD are shared as a matter of course and do not present a problem. Paging space, temporary disk space, and the system residence volume should not be online to (logically shared by) any other system during normal operation. Since I/O operations to these spaces are initiated only by the system to which the space is dedicated, intermixing paging or T-disk space belonging to different systems on the same device, control unit, or channel path introduces unnecessary CSE contention. Such contention could degrade the performance of all systems without providing any compensating functional benefit.

To allow DASD sharing for backup and recovery without creating performance problems, try to:

- Share physically the DASD required for backup operation. To provide the most flexibility, all DASD should be accessible by all systems in the complex.
- Isolate the DASD that is dedicated to a single system. Do not intermix such a DASD on the same device, control unit, or channel path with shared DASDs.
- Avoid logically sharing DASD space that is dedicated to a particular system during normal operation. DASD that contain the following should be online to the system where used and offline to all others:
  - Paging space
  - Directory space
  - T-disk space
  - System residence volume.

When the required DASD is operating in backup mode, it should be switched and varied online to the backup system.

***DASD for Shared Access to Minidisks:*** For CSE to control links between the multiple VM systems in the complex, 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 system configuration files and IOCPs if needed), and identical lists of DASD volumes (in XLINK\_VOLUME\_INCLUDE and EXCLUDE statements in system configuration files) of all systems.

***Maximum Number of Systems Supported for Cross System Link:*** The maximum number of systems supported for cross system link is 56 if the CSE complex uses only shared ECKD capable CKD devices. If any shared CKD devices are used, then the maximum number of systems supported depends on the device type. See Table 13 on page 365 for the maximum number of systems supported for each CKD device type. See “Default Values for the CSE Area” on page 365 for further explanation.

***Choosing Which Minidisks to Share:*** When multiple systems access the same real disks, the probability that CP will find the device busy when an I/O command is entered is increased. Therefore, heavily accessed disks like the CMS S and Y disks should not be shared by all systems. Multiple copies of these minidisks (on separate real volumes) can be provided so that each system can access its own S or Y disk without CSE contention. These volumes can still be included in cross system link for maintenance, but general users cannot share them across VM systems.

*Performance Considerations for CMS Minidisks:* In a CSE environment, 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:

1. Use 4 KB blocking on all CMS minidisks. Since the block size used by the CMS FORMAT command defaults to 2 KB for 3350s and 4 KB for 3380s, you must take positive action to achieve 4 KB blocking on 3350s.
2. Use a 3880-13, 3880-23, or 3990-3,6 to cache heavily used *read-only* minidisks in the controller. This should dramatically reduce the volume's busy time.
3. Put heavily used minidisks on separate volumes and assign users to these minidisks so that the load is primarily from a single system. For example, as you plan the use of the CMS Y disk, consider creating the CP directory entries for MAINT 19E on each system that point to a different real device and cylinder offset, so that each system uses a different Y disk. Another user ID — CMSMAINT, for example — could have minidisk entries defined for all the Y disks in the complex. You would link to the CMSMAINT entries to update all the Y disks from one system.
4. Avoid using the minidisk caching feature for minidisks on shared DASDs. In principle, all general-user minidisks should be shared through cross system linking in the CSE environment. Minidisks that contain the single source directory and serve as preliminary A disks for user IDs such as OPERATOR *must* be shared throughout the complex. Consider minidisks owned by service machines individually, depending on the activity level of each minidisk and how it is used.

**Note:** Using MDC with CSE may cause data integrity problems. See Minidisk Caching Feature (MDC) below.

*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 in the complex. CP and the directory program do not prevent you from defining minidisks that overlap. A CP system in a CSE complex 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.

Correctly protecting an installation's minidisks across a CSE complex means keeping CP directory entries for shared volumes exactly synchronized. This support is provided by DIRMAINT Release 5 with multiple CPU support.

*Minidisk Caching Feature (MDC):* CP can cache CMS minidisks in Expanded or Main Storage. This is a definite performance advantage in a CMS-intensive environment. However, because MDC does not address cross-cache invalidation, the scope of caching is limited to one system only. The MDC feature is therefore incompatible with DASD sharing in any form, with or without CSE. **Using MDC with CSE may cause data integrity problems.**

Since CACHING=ON is the default, the real devices on which all shared minidisks reside can be identified by the SHARED YES specification on their RDEVICE statements in the system configuration file. This specification prevents MDC for these devices. Alternatively, you can suppress MDC at a more granular level by specifying NOMDC on the MINIOPT statement for shared minidisks in the directory and also by using the SET MDC command.

### Placing of the CSE Area

CSE keeps control information about the state of CSE links to minidisks on a special area (called the *CSE area*) of the real volume where the minidisks reside. (In earlier releases this area was referred to as the *CSE track*.) During installation, the XLINK utility initializes this area. This information must be updated 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 during system definition, CSE initializes the CSE area starting on cylinder 0, track 1, when you enter the XLINK FORMAT command. This is described under “Generating the Cross System Link Facility” on page 349. Since 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 Device Support Facilities program (ICKDSF). The XLINK FORMAT command allows these volumes to be shared by formatting the CSE area. For more information about the XLINK FORMAT command, see the *z/VM: CP Commands and Utilities Reference book*.

**Note:** Do not define user minidisks in the space reserved for the CSE area, except for the purpose described under “Generating the Cross System Link Facility” on page 349. For some device types, the CSE area resides on multiple cylinders. See Table 13 on page 365 describing the number of cylinders used for each device. When reserving space for the CSE area, these cylinders must also be considered.

### Defining Volumes Containing Minidisks to be Shared

You can include and exclude volumes containing minidisks to be shared. Use the XLINK\_VOLUME\_INCLUDE statement (see page 268) to include volumes and the XLINK\_VOLUME\_EXCLUDE statement (see page 266) to exclude volumes.

### Cross System Linking and Synchronization of Directories

To protect your installation’s minidisks across the CSE complex, the CP directory entries for shared DASD volumes must be kept exactly synchronized. One method for synchronizing these directories is with the IBM z/VM Directory Maintenance Facility (DirMaint).

So that all CSE systems can link to the same minidisks, the MDISK directory entries for all shared volumes must be identical on all systems in the CSE complex. Directory entries for user IDs that are allowed to be simultaneously logged on to every system 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 VM directory are also defined, minidisks become available to every system.

You can use the OPTION, MDISK, or LINK directory control 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 control 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 control 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 Control Statement (General)” on page 533. For more information about link accesses, see “LINK Directory Control Statement (Device)” on page 500 or “MDISK Directory Control Statement (Device)” on page 513.

## Spool File Directory Statements

In a CSE complex, the MAXSPOOL operand of the SPOOLFILE directory statement limits the number of spool files that a user may have on any given system. Since the CSE complex operates on a single source directory, this limit is the same for all systems. To define different MAXSPOOL limits for different systems, use the system affinity records.

The maximum number of spool files that a user may create on each system in the complex is determined by dividing the number of files declared with MAXSPOOL by the number of systems in the complex. For example, if the MAXSPOOL operand is defined as 300 on a three-system complex, that user may create 100 spool files on *each* of the three systems. If that user uses the conditional statement facilities and chooses to define MAXSPOOL limits of 150 spool files on the first system, 210 spool files on the second system, and 300 spool files on the third system, that user may create up to 50 on the first system, 70 on the second system, and 100 on the third system.

## Preparing for CSE

To prepare for CSE, you must:

- Cable the systems' hardware paths.
- Prepare the existing directories for a merger into a single source directory.
- Put the same version of DIRECTXA on every system in the CSE complex.
- Define user IDs that concurrently log onto all systems so that they have one directory entry (for example, OPERATOR).
- Decide which volumes should be shared. See “Sharing DASD Volumes” on page 336.
- Decide where to place the CSE area on each of the shared volumes. (The default is cylinder 0, track 1.)
- Plan how to set up the communication virtual machine (CVM). See “Installing the VM/Pass-Through Facility and Its Modifications” on page 345 and “Preparing the Communication Virtual Machine (CVM)” on page 350.
- Prepare to deal with outages. Each system should have a properly configured virtual machine in its user directory able to run a guest system in the event another system is disabled. See the *z/VM: System Operation* book for an example.

### Cabling the Hardware

Use the operator's guides for your hardware devices to plan the best cabling for your hardware. Ensure that shared volumes are accessible to all the systems in the CSE complex.

### Preparing for a Single Source Directory

Review the directories of the current systems. Minidisks that are defined in all the directories *must* have identical volume identifiers and cylinder extents, since only one definition of them is in the single source directory. To avoid overlapping extents, you need to:

- Identify identical virtual machines in existing directories
- Define a policy for managing identical identifiers in the existing directories.

**Note:** When using different levels of CMS, it may be necessary to generate CMS on a disk other than the standard 190 disk.

### Nonshared Directory Entries

Some directory entries define user IDs that may be logged on to more than one system within the CSE complex at the same time. These user IDs include:

- The system operator
- The communication virtual machine (CVM)
- The service machines that have identical names and run concurrently on more than one system.

Refer to the z/VM Directory Maintenance (DirMaint) library for information and examples of directory entries.

**Example of a Directory Entry:** Figure 14 is an example of a directory entry. This entry is appropriate for the system operator user ID in each system in the CSE complex. The format is standard for any similar user IDs.

**Note:** You can use either the pre-PROFILE EXECs (see Figure 15 on page 343 and Figure 16 on page 344) or the SYSAFFIN control statement of DIRMAINT Release 5 to set up your directory statements.

```
*****
*      SYSTEM OPERATOR      *
*****
USER OPERATOR XXXXXX 4M   16M ABCDEG
ACCOUNT ACT1 35H:0253
  CONSOLE 009 3215
  SPOOL 00C 2540 READER A
  SPOOL 00D 2540 PUNCH A
  SPOOL 00E 1403 A
  LINK CMSPACKS 190 190 RR
  LINK CMSPACKS 19E 19E RR
  LINK ADISKS 207 191 RR
  LINK ADISKS 204 204 MR
  LINK ADISKS 304 304 MR
  LINK ADISKS 404 404 MR
  LINK ADISKS 504 504 MR
```

Figure 14. Example of a Directory Entry

The ADISKS user ID is an arbitrarily chosen name for the entry that defines the pool of read/write (R/W) CMS minidisks from which each of the common user IDs receives a particular minidisk.

**Note:** The preliminary A disk, linked as 191, will be in read-only (R/O) mode, but each of the real A disks, linked initially as 204, 304, 404, and 504, will be linked in R/W mode (unless some other user already has them in R/W mode). Only one of these disks is required, depending upon which system is being logged on to. The secondary pre-PROFILE EXEC for this user should detach the disks that are not required, so that the disks can be used by the same user ID on an associated system.

**Example of a Pre-PROFILE EXEC:** Figure 15 on page 343 is an example of a pre-PROFILE EXEC for a four-system complex. This sample exec configures OPERATOR and other common user IDs.

```

/*-----*/
/* Definitions for this installation follow. These names are the */
/* same as those used in the combination of XSPool_SYSTEM, */
/* XLINK_SYSTEM_INCLUDE, XLINK_SYSTEM_EXCLUDE statements in a system */
/* configuration file. */
/* */
/**/ sysname1 = "SYSTEM1" /**/
/**/ sysname2 = "SYSTEM2" /**/
/**/ sysname3 = "SYSTEM3" /**/
/**/ sysname4 = "SYSTEM4" /**/
/* */
/**/ userid1 = "OPERATOR" /**/
/**/ userid2 = "PVM" /**/
/**/ userid3 = "none" /**/
/*-----*/
MAKEBUF
'EXECIO * CP (STRING QUERY USERID'
PULL userid junk sysid
SAY userid junk sysid

IF sysid = sysname1 THEN DO; CALL CSE1; EXIT; END;
IF sysid = sysname2 THEN DO; CALL CSE2; EXIT; END;
IF sysid = sysname3 THEN DO; CALL CSE3; EXIT; END;
IF sysid = sysname4 THEN DO; CALL CSE4; EXIT; END;
CALL sysname_bad;

CSE1:
IF userid = userid1 THEN DO; 'EXEC CSE1USR1' ; RETURN; END;
IF userid = userid2 THEN DO; 'EXEC CSE1USR2' ; RETURN; END;
IF userid = userid3 THEN DO; 'EXEC CSE1USR3' ; RETURN; END;
CALL userid_bad;

CSE2:
IF userid = userid1 THEN DO; 'EXEC CSE2USR1' ; RETURN; END;
IF userid = userid2 THEN DO; 'EXEC CSE2USR2' ; RETURN; END;
IF userid = userid3 THEN DO; 'EXEC CSE2USR3' ; RETURN; END;
CALL userid_bad;

CSE3:
IF userid = userid1 THEN DO; 'EXEC CSE3USR1' ; RETURN; END;
IF userid = userid2 THEN DO; 'EXEC CSE3USR2' ; RETURN; END;
IF userid = userid3 THEN DO; 'EXEC CSE3USR3' ; RETURN; END;
CALL userid_bad;

CSE4:
IF userid = userid1 THEN DO; 'EXEC CSE4USR1' ; RETURN; END;
IF userid = userid2 THEN DO; 'EXEC CSE4USR2' ; RETURN; END;
IF userid = userid3 THEN DO; 'EXEC CSE4USR3' ; RETURN; END;
CALL userid_bad;

```

Figure 15. Example of a Pre-PROFILE EXEC (Part 1 of 2)

```

/*-----*/
/* System name not recognized - Cannot continue. */
/*-----*/
Sysname_bad:
SAY "ERROR IN INITIALIZATION OF "||userid||" VIRTUAL MACHINE: "
SAY "CSE SYSTEM NAME "||sysid||" IS UNDEFINED IN THIS PROFILE."
EXIT 999

/*-----*/
/* User ID not recognized - Cannot continue. */
/*-----*/
Userid_bad:
SAY "ERROR IN INITIALIZATION OF "||userid||" VIRTUAL MACHINE: "
SAY "USER ID "||userid||" IS UNDEFINED IN THIS PROFILE."
EXIT 998

```

Figure 15. Example of a Pre-PROFILE EXEC (Part 2 of 2)

**Example of a Secondary Pre-PROFILE EXEC:** Figure 16 is an example of a secondary pre-PROFILE EXEC for a four-system complex. This entry is appropriate for the user ID OPERATOR on system SYSTEM1 defined in Figure 15 on page 343.

```

/*-----*/
/* Secondary pre-PROFILE EXEC for OPERATOR on system SYSTEM1. */
/* */
/* The purpose of this EXEC is to link to and access the proper */
/* R/W A disk for the above user ID and system. To tailor this */
/* EXEC, perform the following 3 steps: */
/* */
/* 1). Modify the following CP LINK command to link to the */
/* proper R/W A disk as defined in the single source directory */
/* for the CSE complex. For SYSTEM2, SYSTEM3, and SYSTEM4, */
/* make comparable changes. */
/* */
/**/'CP LINK * 204 204 MR' /* System SYSTEM1 OPERATOR's A disk */
/* */
/* 2). The following statement will retry the step if the link */
/* was not obtained. */
/* */
/**/ IF rc <> 0 THEN DO;'CP SLEEP 5 SEC'; PUSH 'PROFILE'; EXIT; END /**/
/* */
/* 3). Modify the following CP DETACH commands to detach the other */
/* user's R/W A disks that were also defined in the single */
/* source directory entry for this user ID. These disks were */
/* defined as multiaccess (MR) to eliminate the need for */
/* passwords in this EXEC, and may have been linked R/W when */
/* this user ID logged on. They belong to the same user ID on */
/* another system, and must be detached from this one. Make */
/* comparable changes for SYSTEM2, SYSTEM3, and SYSTEM4. */
/* */
/**/'CP DETACH 304' /* System SYSTEM2 OPERATOR's A disk */
/**/'CP DETACH 404' /* System SYSTEM3 OPERATOR's A disk */
/**/'CP DETACH 504' /* System SYSTEM4 OPERATOR's A disk */
/*-----*/

'RELEASE 191 (DET' /* Detach the preliminary A disk. */
'CP DEFINE 204 191' /* Make the "real" A disk 191. */
'ACCESS 191 A' /* Make it the A disk. */
'EXEC PROFILE' /* Execute the "real" profile. */

```

Figure 16. Secondary Pre-PROFILE EXEC for System1



## Enabling CSE

To enable CSE:

1. Create the single source directory (page 341).
2. Install tapes (page 345):
  - a. VM/Pass-Through Facility (Release 4 with modifications, or later)
  - b. If using DIRMAINT, refer to the *z/VM: Directory Maintenance Facility Commands Reference* book.
3. Create or update the system configuration file (page 346). Add XLINK family statements (and possibly XSPPOOL family statements) to the system configuration file for each system. Add SHARED operands to appropriate CP\_OWNED statements.
4. Initialize the volumes for linking (page 349):
  - a. Generate the XLINK utility program.
  - b. Initialize the CSE areas.

**Note:** If you wish to initialize the CSE areas to something other than the defaults, you must IPL the system prior to running XLINK format.

5. Prepare the communication virtual machine (page 350).
6. Perform an initial program load.
7. Verify the cross system links (page 353).
8. Optionally initialize cross system spool:
  - a. Join the systems (page 353).
  - b. Verify the cross system spool status (page 354).
9. Phase CSE into the production environment (page 356).

## Installing the VM/Pass-Through Facility and Its Modifications

The communication virtual machine (CVM) is the means by which the VM control programs communicate within a CSE complex. The CVM runs the VM/Pass-Through Facility licensed program.

The CVM requires 3 MB for two IPL systems plus 1 MB for each additional system in the complex.

To provide general functions of a pass-through virtual machine, you may use the CVM, or you may use a copy of the VM/Pass-Through Facility running in another virtual machine.

If you use the CVM to provide general pass-through services and CSE communications, you can assign to it the customary user ID for the pass-through virtual machine, PVM. If you use a virtual machine other than the CVM to provide general pass-through services, you should give the user ID PVM to that virtual machine and select another user ID for CVM.

Specify the user ID of the CVM using the COMMUNICATIONS\_USERID operand of the XSPPOOL\_SYSTEM statement (page 271). Install VM/Pass-Through Facility Version 2 Release 1.1 or later. Although the procedures include all the information you need, you must be aware of the layout of the CSE complex and may need to modify:

- The PVM CONFIG statement
- The PROFILE PVM statement

## Enabling CSE

- The entry in the directory that defines the CVM
- The PROFILE EXEC of the CVM.

**Note:** All systems in the CSE complex *must* run the same level of PVM.

These modifications must be completed before the systems in the CSE complex can communicate. Procedures for establishing communication are described under “Preparing the Communication Virtual Machine (CVM)” on page 350.

## Defining CP for Each System

### System Definition for CSE

If CSE is not being used, system definition from the user’s point of view is completely unchanged. A default set of CSE tables that define a one-system complex is generated automatically when the HCPSYS ASSEMBLE file is processed.

**Note:** This is the only way that a one-system complex can be defined. If the XLINK or XSPPOOL families of statements are used in a system configuration file, the CSE complex must be composed of at least two systems.

If CSE is used, you must make the following entries in the system configuration file:

- The CP\_OWNED statement must be coded to identify each CP-owned volume that will contain spool space that belongs to any associated system in the CSE complex.
- XLINK\_SYSTEM\_INCLUDE statements are required.
- The following statements are optional. They may be coded as needed for a particular installation. (Notice that cross system spooling has been separated from cross system linking; separate statements now control these functions.)

#### **XLINK\_DEVICE\_DEFAULTS**

modifies certain attributes of the cross system link operation on a device-type basis. (See page 259.)

#### **XLINK\_SYSTEM\_EXCLUDE**

identifies a system to be excluded from the CSE complex. (See page 263.)

#### **XLINK\_VOLUME\_INCLUDE**

identifies which DASD volumes will be controlled by the cross system link feature of CSE. (See page 268.)

#### **XLINK\_VOLUME\_EXCLUDE**

identifies which DASD volumes will be excluded from control by the (See page 266.) cross system link feature of CSE.

#### **XSPPOOL\_TRACE**

controls the number of free storage pages to be allocated and used for the CSE trace tables. (See page 273.)

#### **XSPPOOL\_XLIST\_INPUT**

identifies certain virtual machines whose input files will not participate in cross system spooling and cross system message and query commands. (See page 274.)

**XSPPOOL\_XLIST\_OUTPUT**

identifies certain virtual machines whose output files will not participate in cross system spooling and cross system message and query commands. (See page 276.)

**CP\_OWNED Statements**

The specific DASD volumes to be used for spool file space are described by CP\_OWNED statements.

Code CP\_OWNED statements to define a list of up to 255 CP-owned DASD volumes. CP-owned volumes are the CP system residence volume and volumes that contain real system paging, spooling, dump, directory, or temporary disk space.

When CSE is not used, all the volumes listed on CP\_OWNED statements that contain spool space may be used to read and write spool files. (Spool files are written only once, when they are created.)

When CSE is active, shared access to spool files for all systems in the complex is provided by physically sharing the spooling DASD. Although physically this is both read and write sharing, logically the sharing is on a read-only basis. Thus, some volumes must be identified as available for writing the spool files created on this system, and others must be identified as available only for reading spool files created on other systems. To do this, specify the CP\_OWNED operands OWN and SHARED as needed.

**Attention:** It is recommended that the system residence volumes and paging volumes not be shared among systems in a CSE complex. If you share system residence and paging volumes, results are unpredictable.

For the format and complete description of the CP\_OWNED statement, refer to “CP\_OWNED Statement” on page 73.

**Example of a System Configuration File**

Figure 17 on page 348 shows portions of a system configuration file for a four-system complex that includes significant information for CSE.

## Enabling CSE

```
/*
Timezone Definitions
*/
Timezone_Definition EDT West 04.00.00

/*
Checkpoint and Warmstart Information
*/
System_Residence ,
Checkpoint Valid KS3RES From Cylinder 2 For 2 ,
Warmstart Valid KS3RES From Cylinder 4 For 2

/*
CP_Owned Volume Statements
*/
CP_Owned Slot 1 KS3RES OWN
CP_Owned Slot 2 MD-SP0 Share
CP_Owned Slot 3 SPOOL0 Share
CP_Owned Slot 4 MD-SP1 Share
CP_Owned Slot 5 SPOOL1 Share
CP_Owned Slot 6 MD-SP2 OWN
CP_Owned Slot 7 SPOOL2 OWN
CP_Owned Slot 8 LDUMP0 Share
CP_Owned Slot 9 LDUMP1 Share
CP_Owned Slot 10 LDUMP2 OWN
CP_Owned Slot 11 LDUMP3 Share

/*
System_Identifier Information
*/
System_Identifier_Default KEN3

/*
User_Volume_List
*/
User_Volume_List KLS191
User_Volume_List CMS55
User_Volume_List UTTOOLS
User_Volume_List CMS195
User_Volume_List XSPACE
User_Volume_List PVMDSK
User_Volume_List PVM191
User_Volume_List TEMP
User_Volume_List MD-SP3
```

Figure 17. Portions of a Configuration File for a CSE Four-System Complex (Part 1 of 2)

```

/*****
/*          XSPPOOL_System Statements          */
/*****
XSPPOOL_System Slot 1  KEN1  Alias K1 ,
      Communications_Userid PVM Timeout 20 Seconds Share_Spool Yes
XSPPOOL_System Slot 2  KEN2  Alias K2 ,
      Communications_Userid PVM Timeout 20 Seconds Share_Spool Yes
XSPPOOL_System Slot 3  KEN3  Alias K3 ,
      Communications_Userid PVM Timeout 20 Seconds Share_Spool Yes
XSPPOOL_System Slot 4  KEN4  Alias K4 ,
      Communications_Userid PVM Timeout 20 Seconds Share_Spool No

/*****
/*          XLINK_Volume_Include Information    */
/*****
XLINK_Volume_Include MD1
XLINK_Volume_Include MD2
XLINK_Volume_Include MD3
XLINK_Volume_Include MD-SP*

/*****
/*          XLINK_Volume_Exclude Information    */
/*****
XLINK_Volume_Exclude MD-SP3

/*****
/*          XSPPOOL_Userid_Input Information    */
/*****
XSPPOOL_Xlist_Input  OPERATNS MAINT KEN IN

/*****
/*          XSPPOOL_Userid_Output Information   */
/*****
XSPPOOL_Xlist_Output OPERATNS MAINT KEN OUT

/*****
/*          XLINK_Device_Defaults              */
/*****
XLINK_Device_Defaults Type 3390 Class 3,
      Cylinder 300 Track 12 Map_Record_Length 203 Map_Records 2

/*****
/*          System Userids                    */
/*****
System_Userids ,
      Operator OPERATOR disconnect ,
      Dump      OPERATNS

```

Figure 17. Portions of a Configuration File for a CSE Four-System Complex (Part 2 of 2)

## Initializing the Volumes for Cross System Link

The CSE area on each DASD volume contains link information about the minidisk on that volume. You need to format this area to prepare DASD for sharing.

### Generating the Cross System Link Facility

The cross system link facility utility contains the XLINK DISPLAY and XLINK FORMAT commands. To ensure that the DASD parameters used by these commands are the same as those used by CP, you should generate a separate version of XLINK for each system in the CSE complex. Use the following command to generate the XLINK MODULE for each system:

```
VMFBLD PPF ppfname compname HCPXLOAD XLINK (ALL
```

## Enabling CSE

*ppfname*

is the name of the product parameter file

*compnname*

is the name of the component

See the *z/VM: VMSES/E Introduction and Reference* book for information about the VMFBLD EXEC.

Now that the XLINK MODULE has been generated, it may be used to format the CSE areas on the DASD volumes that cross system link will control. You must format the areas before the cross system link facility can be activated.

**Note:** The XLINK MODULE should be placed on a nonshared minidisk of the user who will enter the XLINK FORMAT or XLINK DISPLAY command from the CSE system.

For more information about the XLINK FORMAT or XLINK DISPLAY commands, see the *z/VM: CP Commands and Utilities Reference* book.

### Initializing the CSE Area on Shared Minidisk Volumes

To initialize the CSE area on shared minidisk volumes:

1. Connect the volume to be formatted to the virtual machine by one of the following methods:
  - a. LINK, 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. Since this method does not expose any data beyond the CSE area, it is preferred.  
  
**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 13 on page 365 for devices that use multiple cylinders for the CSE area.
  - b. ATTACH—to attach the real volume to the virtual machine.
  - c. LINK, in R/W mode—to link to a minidisk that defines the entire real volume (a full-pack minidisk).
2. Enter an XLINK FORMAT command that specifies the virtual address and the real volume identification of the volume to be formatted. 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. It does require, however, write access to the device as *vdev*.

You should run XLINK FORMAT before you load the new CP module. However, if you wish to initialize the CSE areas to something other than the defaults, you must IPL the system prior to running XLINK format.

For the format, description, and responses of XLINK FORMAT, refer to the *z/VM: CP Commands and Utilities Reference* book.

## Preparing the Communication Virtual Machine (CVM)

To establish proper connections between the CVM of one system in the CSE complex and the CVM of the other systems in the CSE complex, you need to define the addresses of the channel-to-channel links that connect them.

**Note:** When using the 3088 Multisystem Channel Communications Unit, refer to the VM/Pass-Through Facility library.

Figure 18 shows an example that is based on channel-to-channel linking for a four-system complex.

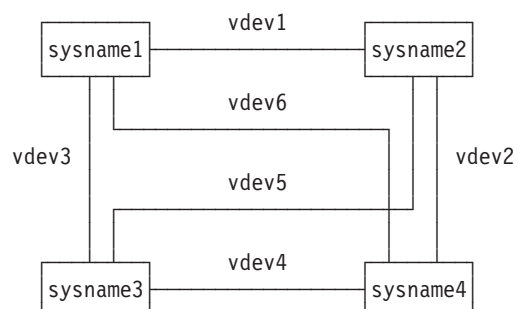


Figure 18. Connecting the Communication Virtual Machines

For the example shown in Figure 18, the directory entry for the CVM user ID should include:

```
DEDICATE vdev1 rdev1
DEDICATE vdev2 rdev2
DEDICATE vdev3 rdev3
DEDICATE vdev4 rdev4
DEDICATE vdev5 rdev5
DEDICATE vdev6 rdev6
```

Modify the PVM CONFIG and PROFILE PVM from each system to include the necessary statements.

For the first system:

- PVM CONFIG should include:

```
LOCAL sysname1
LANG AMENG
LINK vdev1 sysname2 CTCA
LINK vdev3 sysname3 CTCA
LINK vdev6 sysname4 CTCA
```

- PROFILE PVM should include:

```
START LINE vdev1
START LINE vdev3
START LINE vdev6
```

For the second system:

- PVM CONFIG should include:

```
LOCAL sysname2
LANG AMENG
LINK vdev1 sysname1 CTCA
LINK vdev5 sysname3 CTCA
LINK vdev2 sysname4 CTCA
```

- PROFILE PVM should include:

## Enabling CSE

```
START LINE vdev1
START LINE vdev2
START LINE vdev5
```

For the third system:

- PVM CONFIG should include:

```
LOCAL sysname3
LANG AMENG
LINK vdev3 sysname1 CTCA
LINK vdev5 sysname2 CTCA
LINK vdev4 sysname4 CTCA
```

- PROFILE PVM should include:

```
START LINE vdev3
START LINE vdev4
START LINE vdev5
```

For the fourth system:

- PVM CONFIG should include:

```
LOCAL sysname4
LANG AMENG
LINK vdev6 sysname1 CTCA
LINK vdev2 sysname2 CTCA
LINK vdev4 sysname3 CTCA
```

- PROFILE PVM should include:

```
START LINE vdev2
START LINE vdev4
START LINE vdev6
```

**where** (as shown in Figure 18 on page 351):

**sysname1**

is the name of the first system.

**sysname2**

is the name of the second system.

**sysname3**

is the name of the third system.

**sysname4**

is the name of the fourth system.

**vdev1** is the 1- to 3-character virtual device number of the channel-to-channel link for *sysname1* and *sysname2*.

**vdev2** is the 1- to 3-character virtual device number of the channel-to-channel link for *sysname2* and *sysname4*.

**vdev3** is the 1- to 3-character virtual device number of the channel-to-channel link for *sysname1* and *sysname3*.

**vdev4** is the 1- to 3-character virtual device number of the channel-to-channel link for *sysname3* and *sysname4*.



**vdev5** is the 1- to 3-character virtual device number of the channel-to-channel link for *sysname2* and *sysname3*.

**vdev6** is the 1- to 3-character virtual device number of the channel-to-channel link for *sysname1* and *sysname4*.

**rdev1** is the 1- to 4-character real device number of the channel-to-channel link for *sysname1* and *sysname2*.

**rdev2** is the 1- to 4-character real device number of the channel-to-channel link for *sysname2* and *sysname4*.

**rdev3** is the 1- to 4-character real device number of the channel-to-channel link for *sysname1* and *sysname3*.

**rdev4** is the 1- to 4-character real device number of the channel-to-channel link for *sysname3* and *sysname4*.

**rdev5** is the 1- to 4-character real device number of the channel-to-channel link for *sysname2* and *sysname3*.

**rdev6** is the 1- to 4-character real device number of the channel-to-channel link for *sysname1* and *sysname4*.

For more information about routing, links, the START and DROP commands, and 3088 connections, refer to the *VM/Pass-Through Facility Managing and Using* book for Release 4 of the VM/Pass-Through Facility.

## Verifying Your CSE Configuration

1. IPL your z/VM system.
2. Perform your installation's customary verification testing.

## Verifying Cross System Link

Use the XLINK CHECK command when you want to know whether one or more volumes are under cross system link control. You can execute this command after you have completed the IPL of the CSE system. For the format and a complete description of this command, refer to the *z/VM: CP Commands and Utilities Reference* book.

## Enabling Cross System Spool

### Starting CSE Communication

To initialize CSE and bring it into full operation, you need to join each system to CSE by following these procedures:

1. IPL the system and continue with initialization as described in the *z/VM: CP Commands and Utilities Reference* book.

**Note:** Do not start the output spooling devices and do not enable user terminals now.

2. XAUTOLOG the CVM.
3. Use the SMSG command to inform the CVM to start the CSECOM task. CSECOM is the task that joins the systems together. Your installation may have an exec you can invoke to enter this command with fewer keystrokes.

Use the START command to start a CSE communication task.

## Enabling CSE

▶▶—START—CSEcom—*system*—▶▶

### CSEcom

indicates that the CSE communication task is to be started.

**Note:** To join any two systems in the CSE complex, the CSECOM task must be started from *each* of those systems specifying the system to be joined as the target system.

### *system*

indicates the target system with which the session is to be initiated. The target for START CSECOM must be another CSE system.

In the examples that follow, these are *sysname1*, *sysname2*, *sysname3*, and *sysname4*, which specify the names of the systems in the CSE complex defined in the system configuration file. In these examples, *cvm* is the user ID of the communication virtual machine as defined in the system configuration file.

**Example:** The following is an example for the four-system complex shown in Figure 18 on page 351:

1. From the first system, enter the following commands to join the appropriate systems to the CSE complex:

```
SMSG cvm START CSECOM sysname2
SMSG cvm START CSECOM sysname3
SMSG cvm START CSECOM sysname4
```

2. From the second system, enter the following commands to join the appropriate systems to the CSE complex:

```
SMSG cvm START CSECOM sysname1
SMSG cvm START CSECOM sysname3
SMSG cvm START CSECOM sysname4
```

3. From the third system, enter the following commands to join the appropriate systems to the CSE complex:

```
SMSG cvm START CSECOM sysname1
SMSG cvm START CSECOM sysname2
SMSG cvm START CSECOM sysname4
```

4. From the fourth system, enter the following commands to join the appropriate systems to the CSE complex:

```
SMSG cvm START CSECOM sysname1
SMSG cvm START CSECOM sysname2
SMSG cvm START CSECOM sysname3
```

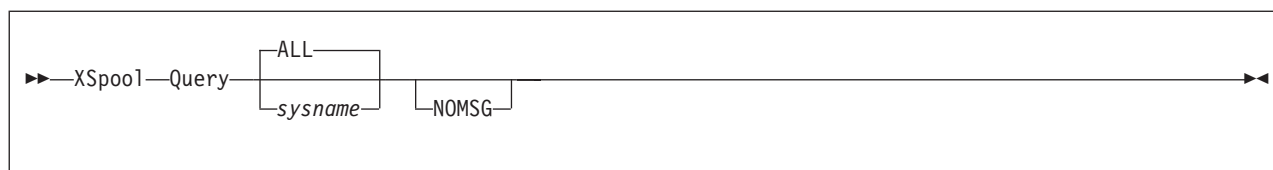
5. After you have started the CSECOM task, enable the lines and terminals and then start the output spooling devices, if any.

**Responses:** The normal responses to a START CSECOM command are:

```
DVMCSE0851I CSE sysnamex CONNECTED TO NODE sysnamey userid
DVMCSE0853I SYSTEM sysnamex SYNCHRONIZED WITH SYSTEM sysnamey
userid
```

### Verifying Cross System Spool

Use the XSPOOL QUERY command to request system spool status.

**sysname**

is the name of the CSE system. If you specify a system name, the status of that system is displayed.

**ALL**

displays the status of all the cross system pool systems. ALL is the default.

**NOMSG**

suppresses the normal command response; however, any error messages are displayed.

**Responses:** The normal response to the XSPOOL QUERY command is:

```
HCPXXS2987I Associated system sysname is active
```

If any of the systems is not active, the response for that system is:

```
HCPXXS2982I Associated system sysname is not active
```

If the CVM link used by CSE is not in operation, the response is:

```
HCPXXS2980I The CSE Communication Virtual Machine is not active
```

**Note:** For a system to be listed as active, the CSECOM task must be started from both the system issuing the XSPOOL QUERY command and the system being queried.

**Activating Spool for Systems Not Active after Start CSECOM**

Use the following commands to activate cross system spool for systems that are listed as not active after starting the CSECOM task for that system.

The following is an example of how to activate the CVM in a four-system complex (shown in Figure 18 on page 351):

1. From *sysname1*, enter the following commands for the systems that are not active:

```
msg sysname1 cvm drop line vdev1
msg sysname1 cvm drop csecom sysname2
msg sysname1 cvm start line vdev1
msg sysname1 cvm start csecom sysname2
```

```
msg sysname1 cvm drop line vdev3
msg sysname1 cvm drop csecom sysname3
msg sysname1 cvm start line vdev3
msg sysname1 cvm start csecom sysname3
```

```
msg sysname1 cvm drop line vdev6
msg sysname1 cvm drop csecom sysname4
msg sysname1 cvm start line vdev6
msg sysname1 cvm start csecom sysname4
```

2. From *sysname2*, enter the following commands for the systems that are not active:

```
msg sysname2 cvm drop line vdev1
msg sysname2 cvm drop csecom sysname1
msg sysname2 cvm start line vdev1
msg sysname2 cvm start csecom sysname1
```

## Enabling CSE

```
smsg cvm drop line vdev5
smsg cvm drop csecom sysname3
smsg cvm start line vdev5
smsg cvm start csecom sysname3
```

```
smsg cvm drop line vdev2
smsg cvm drop csecom sysname4
smsg cvm start line vdev2
smsg cvm start csecom sysname4
```

3. From *sysname3*, enter the following commands for the systems that are not active:

```
smsg cvm drop line vdev3
smsg cvm drop csecom sysname1
smsg cvm start line vdev3
smsg cvm start csecom sysname1
```

```
smsg cvm drop line vdev5
smsg cvm drop csecom sysname2
smsg cvm start line vdev5
smsg cvm start csecom sysname2
```

```
smsg cvm drop line vdev4
smsg cvm drop csecom sysname4
smsg cvm start line vdev4
smsg cvm start csecom sysname4
```

4. From *sysname4*, enter the following commands for the systems that are not active:

```
smsg cvm drop line vdev6
smsg cvm drop csecom sysname1
smsg cvm start line vdev6
smsg cvm start csecom sysname1
```

```
smsg cvm drop line vdev2
smsg cvm drop csecom sysname2
smsg cvm start line vdev2
smsg cvm start csecom sysname2
```

```
smsg cvm drop line vdev4
smsg cvm drop csecom sysname3
smsg cvm start line vdev4
smsg cvm start csecom sysname3
```

**Note:** If you find a problem using the START and DROP process and the lines are still not active, you can use another method to activate cross system spool. To use this alternative method, use the CP FORCE command with the CVM user ID and then re-XAUTOLOG each CVM user ID on all systems.

## Phasing CSE into the Production Environment

You should introduce CSE into your environment in a manner that builds confidence in the stability of the complex before you create dependencies on its functions and capabilities.

The following steps may vary according to installation policy and experience:

1. IPL one system in the complex but do not activate cross system link and spool. To keep from activating cross system link:
  - Do not start the CSECOM task of the communication virtual machine.
  - Do not bring any shared minidisk volumes online to the other systems in the CSE complex.

- Ensure that the current non-CSE system has the same list of CP-owned volumes with dummy volumes replacing those volumes having the share option. (This list is from the CP\_OWNED statements in the system configuration file).
2. When you have confidence in the stability of the system:
    - a. IPL CP on each of the systems.
    - b. Introduce the cross system link facility by bringing online all shared minidisk volumes, after they have been formatted by the cross system link facility utility program.
    - c. Start the CSECOM task in the CVMs of all the systems in the CSE complex to initiate the cross system spool facility.

**Note:** You may place the START CSECOM command in the PROFILE PVM file on the CVM user ID at a later time.

### Networking Implications of CSE

A single VM system may be a node in many different kinds of networks, all of which require their own independent form of definition and control. CSE does not introduce any new networking mechanism, but it does have an effect on the definition of network nodes. In general terms, the following kinds of networks are typically found on VM systems:

- RSCS
- PVM
- SNA.

**RSCS Networks:** Users communicate with each other over an RSCS network by using RSCS service virtual machines, and they communicate with these service virtual machines by means of spool files and the SMSG mechanism. Since both of these modes are supported across systems by CSE, an entire CSE complex may appear as a single node on an RSCS network.

Users can enter CMS commands such as SENDFILE and TELL to communicate conveniently with users on the same node or on different nodes in an RSCS network. These commands and RSCS network communicating programs use the IDENTIFY command to determine the user ID of the RSCS virtual machine and the RSCS node name of the system on which they are operating. The source of the data supplied by the IDENTIFY command is a CMS file, usually found on the S disk, named SYSTEM NETID. If your installation uses a single RSCS node, this file is the same for all systems in the CSE complex. Commands and programs such as SENDFILE and TELL operate the same way in the complex as they do on a single system.

There is no systematic relationship between the RSCS node name (as specified in SYSTEM NETID) and any of the node name specifications of the other networking mechanisms that may be in use, or any of the CSE complex system names.

**PVM Networks:** PVM uses the logical device facility to allow users to log on to host machines. Since this mode is not supported across a CSE complex, each system in the complex appears to the PVM network as a single node. In other words, for PVM purposes the CSE complex does not exist.

Network node definitions for PVM are in the PVM CONFIG file on the PVM service machine's A disk. Again, there is no systematic relationship between the PVM node name and any of the node name specifications of the other networking

## Enabling CSE

mechanisms, or any of the CSE complex system names. It does make sense, however, to define the PVM node names for the systems in a CSE complex so that they are the same as their CSE system names.

**SNA Networks:** VM/VTAM uses CP console services (CCS) to allow users to log on to host machines. Since this mode is not supported across a CSE complex, each system in the complex appears to the SNA network as a single node, and for SNA purposes the CSE complex does not exist.

Network node definitions for SNA are contained in data files managed by VTAM and have no systematic relationship with any of the node name specifications of other networking mechanisms or with any of the CSE complex system names. As in the case of PVM, it makes sense to define the SNA node names for the systems in a CSE complex so that they are the same as their CSE system names.

---

## Administering CSE

This section is to help you manage CSE as part of your system resources. The information describes the cross system link, the CSE area, and cross system spooling.

### How Linking Works in z/VM

The unit of data control on z/VM systems is the minidisk. A minidisk is defined to CP by specifying in the CP directory the DASD volume on which it is located, the cylinder on which it starts, and the number of cylinders it encompasses. You should avoid defining minidisk extents that overlap.

The CP directory controls access to all minidisks. The CP LINK command is the mechanism by which virtual machines are connected to a particular minidisk. Access may be read-only (R/O) or read/write (R/W). A user with the necessary authorization may use stable read, stable write, exclusive read, or exclusive write access.

By linking to another user's minidisk, users can share data, usually CMS files. The access method that controls the updating of data on the minidisk is not part of CP, but runs in each user's virtual machine. That access method does not support data sharing. Therefore, any updating of data on minidisks can result in read or write failures or even data loss.

When a link is requested, CP finds the status of existing links. It then grants or denies the requested new link depending on CP link protocols. CP checks all links to that same minidisk and all minidisks that overlap the cylinders of the requested minidisk with these exceptions:

- If you are requesting a full-pack minidisk, CP ignores any smaller minidisks that may be active on that same DASD.
- If you are requesting a smaller-than-full-pack minidisk, CP ignores any full-pack minidisks that may be active on that same DASD.

Defining full-pack/smaller-than-full-pack minidisk overlap compromises the data integrity otherwise guaranteed by the stable and exclusive link access modes, because a full-pack minidisk can be exclusive write at the same time that a smaller minidisk on that same DASD is exclusive write. If you define such overlap, you assume responsibility for data integrity.

DASD RESERVE/RELEASE processing is not designed to handle overlapping minidisks and will issue a virtual reserve for each of the two overlapping minidisks. If you define such overlap, you assume responsibility for data integrity.

To maintain data integrity you must control the use of the command DEFINE MDISK and of DIAGNOSE code X'E4' for full-pack overlay. These are two powerful variants of LINK which must be used carefully or not at all to maintain data integrity.

## How CSE Extends Linking

Cross system link extends the CP LINK command. When multiple VM systems are connected in a CSE complex, CP LINK protection includes all minidisks defined in common across the CSE complex. Thus, users on any of these systems can enter LINK commands without regard to which system they are actually logged on to. From the general user's point of view, cross system link functions in the same manner that the current CP LINK command does and is therefore transparent.

To record and interrogate the types of links outstanding on all interconnected systems, CSE uses channel programs that work with CKD DASD, including ECKD capable CKD DASD. Cross system link does not work with fixed-block architecture (FBA) DASDs.

The hardware facility reserve/release is *not* used.

It is important that the CSE area not be updated inadvertently by any program other than CP. If you do not specify a particular location in the system configuration file, CSE initializes the CSE area starting on cylinder 0, track 1, when you enter the XLINK FORMAT command. This is described under "Generating the Cross System Link Facility" on page 349. Since 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 Device Support Facilities program (ICKDSF). The XLINK FORMAT command allows these volumes to be shared by assigning and formatting the CSE area.

**Note:** For some device types, the CSE area resides on multiple cylinders. See Table 13 on page 365 describing the number of cylinders used for each device. Be careful not to define user minidisks in the space reserved for the CSE area, except for the purpose described under "Generating the Cross System Link Facility" on page 349.

### Controlling Cross System Link

The lists of shared and excluded volumes reside in the system configuration file. The list of shared volumes is in the XLINK\_VOLUME\_INCLUDE statements. The list of volumes to be excluded from sharing is in the XLINK\_VOLUME\_EXCLUDE statements.

The XLINK RESET command is provided to recover links and prevent lockouts if one of the systems in the CSE complex fails. This should be used with the cautions described in the *z/VM: CP Commands and Utilities Reference* book.

## CSE Area

Information about the state of current links is kept on special tracks (called the CSE area) on each DASD volume being shared. To see these link indicators, use the XLINK DISPLAY command (see the *z/VM: CMS Commands and Utilities Reference* book).

## Administering CSE

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. Since 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 19 on page 362 for an example of the format of a CKD formatted CSE area, and see Figure 20 on page 363 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. For more details on the XLINK FORMAT command, see the *z/VM: CP Commands and Utilities Reference* book. A series of channel programs are used to interrogate and update the link flag. See Figure 21 on page 363 for an example of an ECKD-formatted CSE area, and Figure 22 on page 364 for an example of link records for an ECKD-formatted CSE area.

Each link state is signified by the following flags:

- A NOLINK flag indicates that no links exist for the minidisk.
- A READ flag indicates that read links exist for the minidisk, but write links do not.
- A WRITE flag indicates that at least one write link, and possibly one or more read links, exist for the minidisk.
- A STABLE READ flag indicates that 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.
- A STABLE WRITE flag indicates that 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.
- An EXCLUSIVE READ flag indicates that one exclusive read link exists for the minidisk. No links will be granted while the EXCLUSIVE READ flag exists for the minidisk.
- An EXCLUSIVE WRITE flag indicates that one exclusive write link exists for the minidisk. No links will be granted while the EXCLUSIVE WRITE flag exists for the minidisk.

To operate cross system link, 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 cross system link facility utility (XLINK FORMAT command). Cross system link cannot support DASD that have not been formatted by this program or that are attached to virtual machines.

The cross system link facility 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. No program other than CP should have write access to the CSE area of any volume attached to the system, lest it be inadvertently changed.

Since 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. A DASD hardware failure on a CSE complex has no greater impact than that on a single VM system without CSE.

### 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, for the CSE area. CP keeps the volume label and allocation maps on cylinder 0, track 0. Therefore, the remaining tracks on cylinder 1 are available. Regardless of the location of the CSE area, it should be allocated as PERM space. For some device types, the CSE area resides on multiple cylinders. See Table 13 on page 365 describing the number of cylinders used for each device.

CSE needs to update information about links 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 may not.

To reduce the need for, and frequency of, input/output (I/O) to CSE areas, the primary read-only (RR) link mode should be used in LINK commands and directory control statements whenever possible. Since RR links are always granted once 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 may 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 user's minidisk space, it may 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.

### Placing of the CSE Area

Figure 19 on page 362 illustrates the general format of a CKD-formatted CSE area in an *n*-system complex.

## Administering CSE

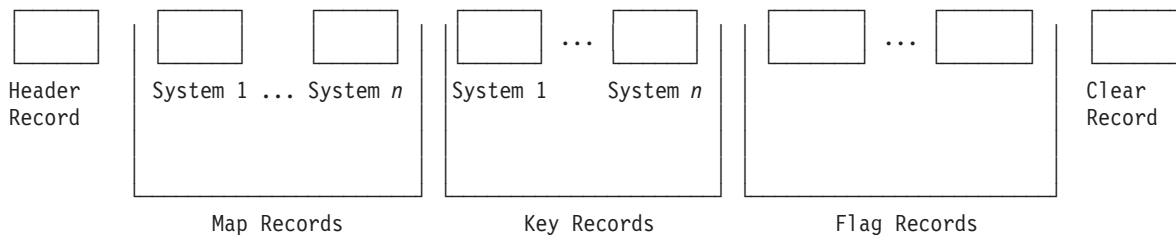


Figure 19. Format of a CKD-Formatted CSE Area for an  $n$ -System Complex

<b>1 Header Record</b>	Contains 240 bytes of cross system link general information.
<b><math>n</math> Map Records</b>	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.
<b><math>n</math> Key Records</b>	One record per system, each containing 1 byte of irrelevant information. The key area is used as a comparison argument.
<b>7 Flag Records</b>	Each record contains a byte representing NOLINK, READ, WRITE, STABLE READ, STABLE WRITE, EXCLUSIVE READ, or EXCLUSIVE WRITE link flags.
<b>1 Clear Record</b>	Contains in all bytes NOLINK link flags. 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 RECS= operand on the XLINK\_DEVICE\_DEFAULTS statement (page "XLINK\_DEVICE\_DEFAULTS Statement" on page 259) and the CYLINDER operand on the XLINK\_VOLUME\_INCLUDE statement (page "XLINK\_VOLUME\_INCLUDE Statement" on page 268).

The number of map records used should equal the number of systems that your CSE complex supports.

Map records show the state of the existing links on the CSE systems. Figure 20 on page 363 is an example.

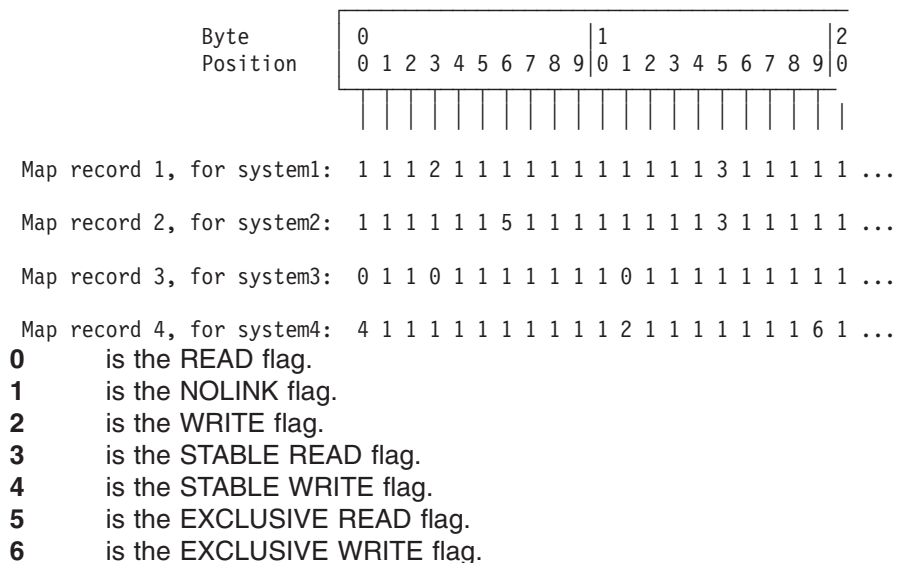


Figure 20. Sample Map Records for a CKD-Formatted CSE Area for a Four-System Complex

These map records show that:

- 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.

Figure 21 illustrates the general format of an ECKD-formatted CSE area in an *n*-system complex.

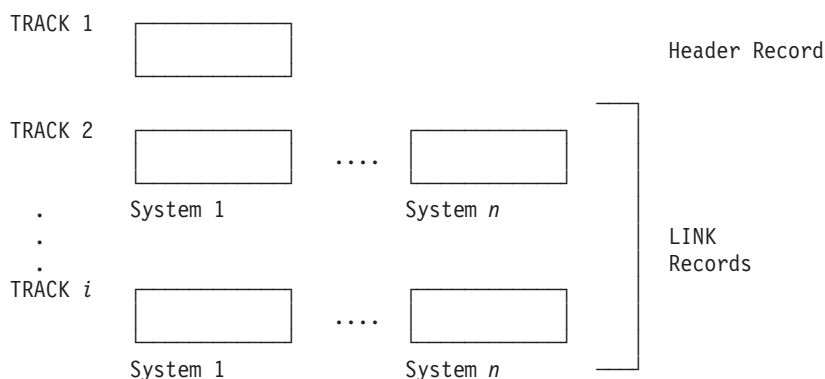


Figure 21. Format of a CKD-Formatted CSE Area for an *n*-System Complex

**1 Header Record**

Contains 56 bytes of cross system link general information.

***n*\**i* LINK Records**

Of length 256 bytes. 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.

*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 RECS= operand on the XLINK\_DEVICE\_DEFAULTS statement (page “XLINK\_DEVICE\_DEFAULTS Statement” on page 259) and the MAP\_RECORDS operand on the XLINK\_VOLUME\_INCLUDE statement (page “XLINK\_VOLUME\_INCLUDE Statement” on page 268).

**Note:** For some devices, the CSE area does not fit on one cylinder so multiple cylinders are used. See Table 13 on page 365 describing 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 RECL= operand on the XLINK\_DEVICE\_DEFAULTS statement (page “XLINK\_DEVICE\_DEFAULTS Statement” on page 259) and the MAP\_RECORD\_LENGTH operand on the XLINK\_VOLUME\_INCLUDE statement (page “XLINK\_VOLUME\_INCLUDE Statement” on page 268).

Figure 22 is an example of link records for an ECKD-formatted CSE area.

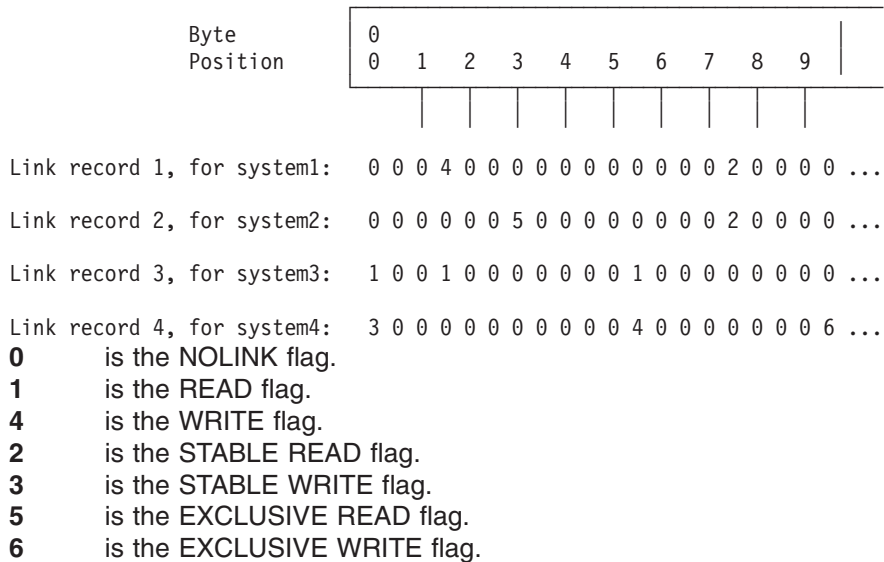


Figure 22. Sample Link Records for an ECKD-Formatted CSE Area for a Four-System Complex

These LINK records are on track 2 of the CSE area and show that:

- 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 a minidisk beginning at cylinder 11, and an exclusive write link to a minidisk beginning at cylinder 19.

**Default Values for the CSE Area:** The defaults in the CSE system tables are such that the cross system link facility utility formats each volume with the maximum number of map records that can be recorded on 1 track for CKD-formatted DASDs. The same defaults are used for ECKD capable CKD DASDs even though the maximum number allowed is higher. Table 13 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 13. 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
<b>Note:</b> These defaults support CKD and ECKD capable CKD DASDs. FBA is not supported. The length of each map record is the maximum number of cylinders on any currently existing model of that device type.			
<b>Note:</b> The NUMBER OF CYLINDERS NEEDED FOR CSE AREA is for ECKD-capable 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.			

Eight records are defined for *other* devices, but the actual number of records used is limited to the number that fit on a track of such *other* device.

## VM I/O Performance Measurement for Shared DASD

The *z/VM: Performance* book describes CP monitoring in detail. When you are using CSE, the information received when you request seek domain monitoring may be misleading. To measure use of a shared DASD accurately, you need to use hardware measuring methods.

CP monitor can function on only one system; it cannot determine the effect that requests for shared DASD made by each system in a CSE complex have on that device.

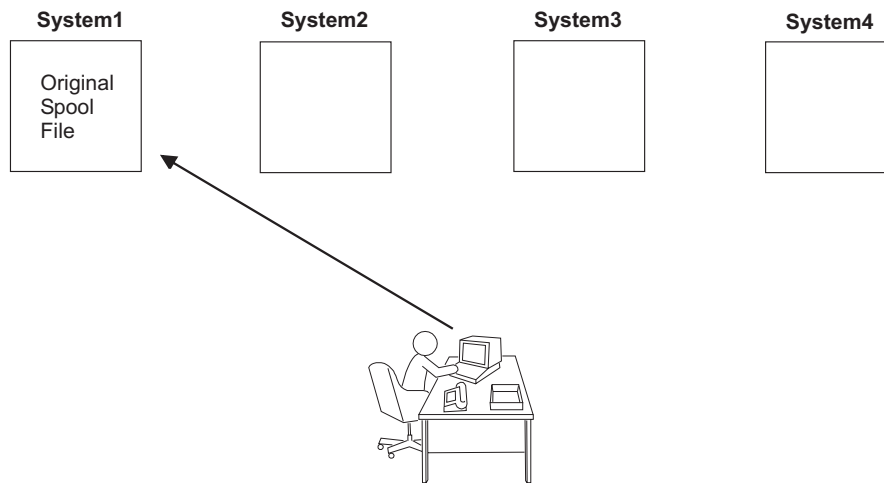
Except for DASD performance, measuring performance within a CSE complex is the same for the same systems operating independently.

## Administering CSE

Complexes with 3380-13 and 3380-23 cache controllers installed should consider sharing minidisks, even heavily used ones, by caching them. These cache controllers reduce mechanical I/O activity and therefore device contention that may be caused by either cross-system or single-system activity.

### What Is Cross System Spool?

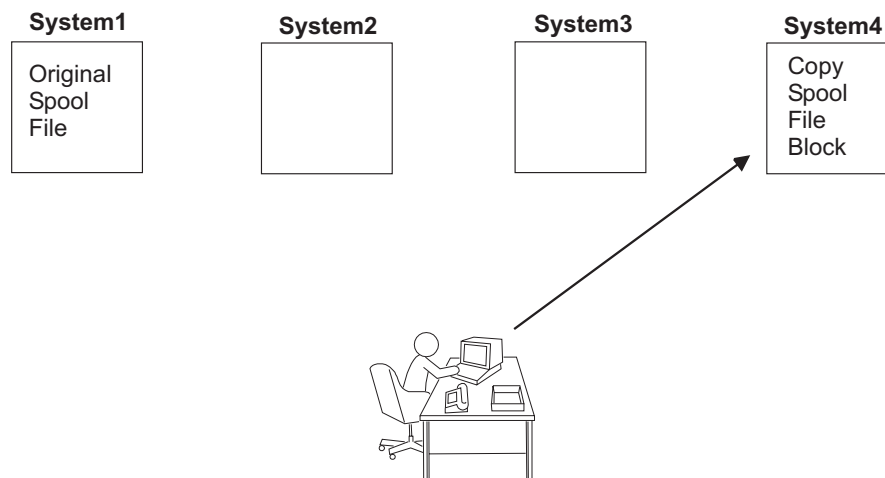
Figures 23 through 25 show how cross system spool works in a four-system complex.



The user:

1. Logs on to system1
2. Creates a reader spool file on system1.

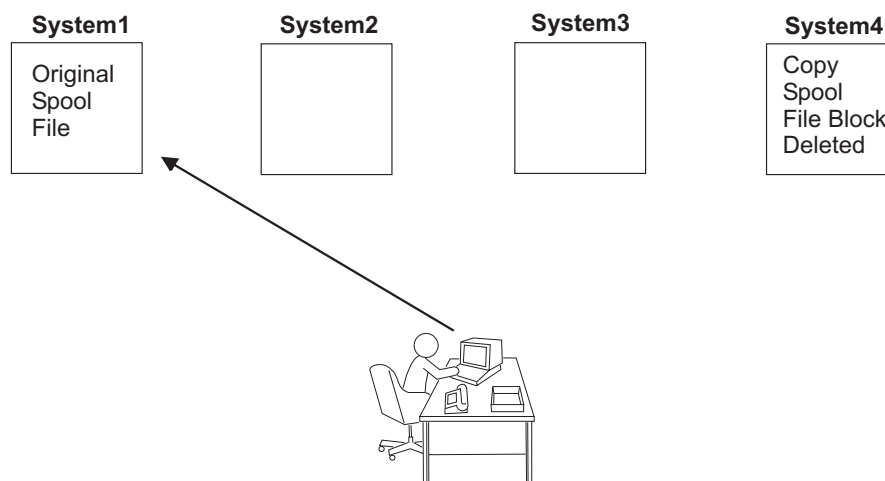
*Figure 23. Cross System Spool—Step 1*



The user:

1. Logs off from system1.
2. Logs on to system4. System1 sends system4 a copy of the spool file block from the original spool file.
3. Sees the spool file as being on system4. The user can process the spool file.

Figure 24. Cross System Spool—Step 2



The user:

1. Logs off from system4
2. Logs on to system1
3. Processes the reader spool file on system1 again.

The copy of the spool file block on system4 is deleted.

Figure 25. Cross System Spool—Step 3

## How CSE Extends Spooling

With cross system spool, the CP spooling system can include up to four z/VM systems connected in a CSE complex. The connection mechanism used requires shared DASD hardware and a VM/Pass-Through Facility network.

## Administering CSE

Within the CSE complex, users 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 these systems can access all of their spool files regardless of which system they have logged on to for that session. They can transfer these files to other users or to service virtual machines anywhere in the CSE complex. They can select printers and other spool output devices for processing spool files created by any user in the CSE complex. Thus, end-user view of spooling is nearly transparent. Although the CP commands that manipulate spool files operate differently internally, additional user options or parameters are not required.

Cross system spooling enables CP spooling functions on each system in the complex to manage and share spool files cooperatively. Because of this, when all the systems are operating normally, there is little awareness of CSE and no special measures are required either for users or for operators. For example, a printer pool that might formerly have resided on one system and have been reached by users on other systems over a network connection appears local to each user, regardless of which system in the CSE complex that user is on. The files to be printed appear to the spooling operator to be local to the system on which the printers are active. Operator intervention is required for proper handling of scheduled or unscheduled system outages.

Cross system spool is concerned only with closed spool files and not with the allocation of CP spool space. Each system has its own set of spooling disks to allocate from exclusively. All other CP systems within the CSE complex have read and write access to these spool disks because they are on shared DASDs. These other systems, however, cannot allocate or release space on these volumes.

The system on which a spool file is created owns the spool space. The spool descriptor on the system where the file was created is called an *original spool descriptor*. The identity of the originating system, the one that owns the spool space the file occupies, is kept in the spool descriptor. Another system may receive a copy of the original spool descriptor, which is known as a *copy spool descriptor*. Copy spool descriptors may be deleted by any of the sharing systems, but the actual returning of spool file resources is performed by the originating system. Each system has peer status. This means that each system has responsibility for its original spool descriptors and for keeping the other systems' copy spool descriptors up to date.

CSE 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 systems so that they can be handled by real printers or punches. Input spool files must be available only on the system where their owning user ID is active.

Because more users are involved, the number of spool file blocks in a CSE complex may easily exceed the number normally expected on a single system. Across the CSE complex, the total amount of main storage required for original and copy spool file blocks increases each time a system receives a copy spool file block.

### **z/VM Spooling Commands Extended by CSE**

Cross system spool extends the scope of the following CP commands to apply to spool files anywhere in the CSE complex:

- 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 by CSE.

CP spooling commands, such as PURGE and CHANGE, normally access only files that are not locked at all or that are locked by the system on which the command was entered. For example, suppose that the system on which a command has been issued is referred to as *systemA*. Suppose that another system involved in the execution of the command is referred to as *systemB*.

Most spooling commands directly process only original spool files, that is, spool files created on *systemA*. Commands, such as TRANSFER and CHANGE, that try to modify the status of *copy spool descriptors* are sent to the system that originated the spool file, *systemB*, for processing. (Copy spool descriptors are those created by another system in the CSE complex.) These commands are executed on *systemB* exactly as they were entered on *systemA*, as if the user had actually entered the command on *systemB*.

A copy of the spooling command is sent to another system whenever a candidate spool file is found that is a copy of an original spool file owned by that system. This concept reduces the amount of additional CSE processing necessary to process CP spooling commands across the CSE complex.

The TAG and PURGE commands are exceptions. TAG information is not kept in the spool descriptor but is on a DASD. Also, all systems actually have shared R/W access to that DASD. Thus, *systemA* performs the command directly, whether it is the originating system or not. PURGE does not modify the spool descriptor, but merely obtains the file's lock and deletes the file.

SPOOL commands, such as SPOOL PRINTER HOLD, that refer to virtual devices rather than to spool files are executed only on the system on which they are entered.

### Copy Input (Reader) Spool Files

When an original spool descriptor is closed and placed on the input queue and the owner is logged on to the same system, a copy spool descriptor is not sent to another system. There is just one original spool descriptor for this spool file. If the owner is logged on to another system but is not logged on to the originating system, a copy input spool descriptor is sent to the system that the user is currently logged on to. 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 system exists.

When a user logs on, all other systems in the CSE complex send a copy of any available input (reader) spool descriptors owned by the user to the system where the user is logged on. A copy input spool descriptor exists on a system as long as the owner remains logged on to that system, assuming nothing is done to change its place in the input queue. When a user logs off, all copy input spool descriptors on that system will be deleted. Only the original spool descriptors remain on the originating system.

**Note:** If a reader spool file is being dumped to tape by the SPXTAPE DUMP command on the originating system, and the owner of the file is logged on to another system in the CSE complex, 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 the other systems when an original spool file is added to the output queue. The spool descriptor remains on each system until it 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 system 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 system 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 system receives a copy spool descriptor, 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 system. A file's spool ID is always the same throughout the complex. Since any system can process an output spool file, whether the user is logged on or not, output spool files always exist on all the systems in the CSE complex.

### Spool File Numbering

See "Spool File Directory Statements" on page 341 for a description of how the maximum number of spool files is determined.

Spool files for any given user are assigned spool ID numbers starting from 0001 and proceeding to 9999, or to the MAXSPOOL limit if a SPOOLFILE statement was specified in that user's directory entry. In a z/VM system without CSE, these spool ID numbers start at 1 and increment by 1 for each new spool file owned by that user. When CSE is installed, the increment and the initial value of the spool ID numbers depend on the relative position of the system on which the spool file is created in the list of systems in the CSE complex. This is done so that no user has more than one spool file with any given spool ID number.

If only two systems are in the CSE complex, spool files created on one system are assigned all the odd numbers from 0001 to MAXSPOOL, and those created on the other system are assigned all the even numbers in the same range. If three systems are specified, spool ID numbers for spool files created on any system1 start at 0001, on system2 at 0002, and on system3 at 0003. If four systems are specified, spool ID numbers for spool files created on any system1 start at 0001, on system2 at 0002, on system3 at 0003, and on system4 at 0004.

The number by which the spool ID number is incremented is the same as the number of systems in the complex. For example, in a two-system complex, 2 is added to each spool ID number to create the next; in a three-system complex, 3 is added to that number, and so on.

### Access to Spool Files

Privileged users have access to all output spool files from any of the coupled systems. Privileged users can access the reader files of other users only on the system where the owner is logged on and on the system having the original spool descriptor.

However, since privileged users need access only to the spool files that the system can process or to the input files that exist on their system, this is not a limitation.

## VM/Pass-Through Facility Functions for Cross System Spool

The VM/Pass-Through Facility is used to communicate among systems within the CSE complex. The virtual machine that runs the VM/Pass-Through Facility for this purpose is called the *communication virtual machine (CVM)*. The VM/Pass-Through Facility supports either binary synchronous communication (BSC) lines or channel-to-channel adapters (CTCAs). Though CSE supports BSC for its operations, BSC lines are not recommended, because they are too slow.

Each CVM is a VM/Pass-Through Facility virtual machine. There is a control task, CSECOM, in each CVM for each system within the CSE complex. When four systems are connected and accessing each other's spool files, three CSECOM tasks are started and active on each system. A CSECOM task may be started by an authorized VM/Pass-Through Facility user or as part of the VM/Pass-Through Facility virtual machine initialization.

A CSECOM task:

- Synchronizes spool files
- Communicates with CP
- Sends data to, and receives data from, another CSECOM task.

CP communicates with the VM/Pass-Through Facility virtual machine using the Virtual Machine Communication Facility (VMCF).

Synchronization takes place:

- Each time the CSECOM task starts up
- When another system requests synchronization
- When a CP spool command detects a problem requiring synchronization.

CSE queues requests to be sent to the other systems, such as requests to lock, unlock, or delete spool files. The CSECOM task associated with a particular system within the CSE complex is posted when a request item is queued to be sent to that system. The CSECOM task reads the request and then sends it to the CSECOM task on the other system. There the requests are received by the CSECOM task, passed to CP, and serviced.

The VM/Pass-Through Facility virtual machine controls the CTCA line hardware that connects the systems. This minimizes the communication hardware necessary to couple systems and allows the sharing of any existing hardware connection with other VM/Pass-Through Facility tasks. The VM/Pass-Through Facility virtual machine handles the connection, reconnection, rerouting, and the other line problems associated with communication service.

### Spool Synchronization Request Tracing

CSE maintains a trace of requests to synchronize spool files among systems within the CSE complex. The trace entries record why the synchronization was requested and which module caused it. A synchronization request can be lost because of programming errors or hardware problems in the link between CVMs. The trace entries identify the source of the error.

## Multiple Concurrent Logon Sessions

CSE prevents a user from logging on with the same user ID concurrently on more than one system within the CSE complex. However, sometimes (primarily when operational and service virtual machines are involved) multiple concurrent logon sessions with the same user ID are useful. Excluding a user ID from cross system

## Administering CSE

spooling is done using the XSPPOOL\_XLIST\_INPUT statement (page 274) and the XSPPOOL\_XLIST\_OUTPUT statement (page 276).

The user IDs specified are excluded from cross system spooling but can log on to multiple systems.

**Note:** The lists of excluded user IDs must be the same for every system in the CSE complex.

For information about creating a single source directory for a CSE complex, refer to *z/VM: Directory Maintenance Facility Commands Reference*.

---

## Preparing to Deal with Outages

If the hardware of a system in the CSE complex should fail, both the users of that system and other users throughout the CSE complex may be affected. If the outage is a long one, the option to continue service should be available by operating the failed system's software in a virtual machine on a remaining system.

You should be aware that virtual operation makes spool files created on the down system available to all users of the CSE complex. For performance, during this backup mode of service, users should use and applications should run on, the other systems within the CSE complex whenever that is practical.

For each system in the CSE complex, an entry in the single source directory should be created with the following information:

- Full-pack minidisk definitions should be included for each volume in the list of CP-owned volumes.
- Full-pack minidisk definitions should be included for each volume containing minidisks required during the period of virtual operation. If the number of users and applications supported during this period is limited, not all minidisk volumes may be required.
- DEDICATE directory control statements should be included for the real channel-to-channel adapters (CTCAs) or IBM 3088 ports used to connect the down hardware with the other systems within the CSE complex. If the real hardware is not available when virtual operation is required, virtual CTCAs can be defined to connect the virtual system with its host. If more than two systems are in the complex, you may have to change PVM CONFIG files of some CVMs.
- Terminal definitions, as required, should be included if users are to be permitted to use the DIAL command to access this system from the host.

The *z/VM: System Operation* book describes how to respond to outages.

## Returning to Non-CSE Operation

If you disable CSE from your complex, you must do the following:

- Distribute spool files to their owners using RSCS or the SPXTAPE command.
- Assign users to one particular system.
- Duplicate the once-common source directory, and provide a copy for each of the now separate systems.
- If you are using DIRMAINT Release 5 to synchronize directories, define a service virtual machine using that program for each system.
- Delete MDISK statements defining minidisks that will not exist on a particular system from that system's user directory.

---

## Chapter 11. 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 the *z/VM: CP Exit Customization* book.

---

### 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 26 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.
*
*****
MSUEXEND SLR   R15,R15          Set return code of 0
          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 26. Module HCPMSU Overview

HCPMSU has the following requirements and restrictions:

1. HCPMSU is written in Assembler H coding language.

## Customizing Messages

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, shown in Figure 5 on page 312. 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."

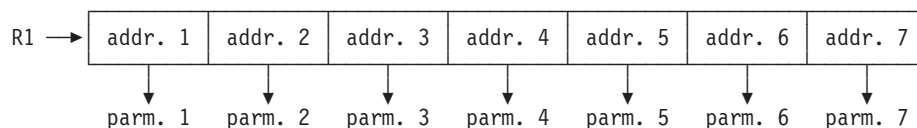
Upon return to the CP message function from entry point HCPMSUEX, general-purpose register 15 must contain one of the following return codes:

Value	Meaning
0	Normal processing continues.
<i>nnnn</i>	An error was detected during HCPMSUEX exit processing. The installation-defined return code <i>nnnn</i> 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 <i>nnnn</i> .

See *z/VM: System Messages and Codes - CP* 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.
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:

## Customizing Messages

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.
- 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.
  - 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.
  - 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 <i>userid</i> : <i>text</i>
MSGNOH	<i>text</i>
SMSG	<i>text</i>
WARNING	* WNG FROM <i>userid</i> : <i>text</i>

**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.

- 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.
- 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.

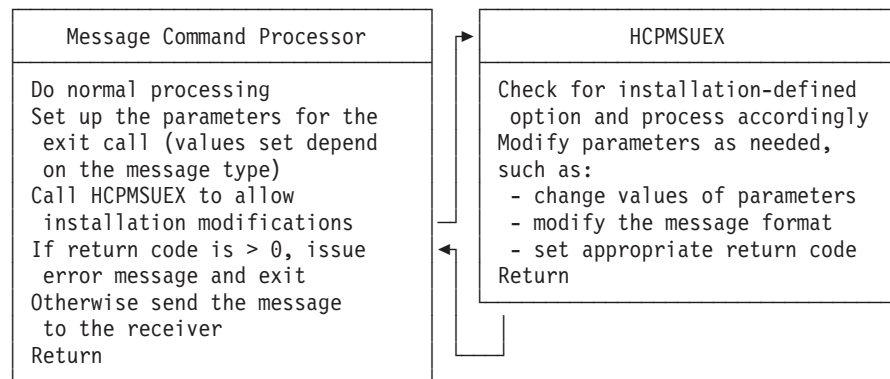


Figure 27. Flow of CP Message Function Customization

Figure 27 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

## Customizing Messages

processed. You can change these values, reformat the message, and do additional verification. See “Parameter Values” on page 374 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 12. 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 VM
- Directory Maintenance licensed program (DIRMAINT)
- Interactive System Productivity Facility (ISPF)

#### ***Resource Access Control Facility (RACF) for VM:***

- Helps your installation implement its security policy
- Identifies and authenticates each user
- Controls each user's access to sensitive data
- Logs and reports events that are relevant to the system's security.

For detailed information on RACF for VM, see the following manuals:

*RACF General Information*

*System Programming Library: RACF*

*RACF Command Language Reference*

*RACF Security Administrator's Guide*

*RACF Auditor's Guide*

*RACF Diagnosis Guide*

#### ***Directory Maintenance Licensed Program (DIRMAINT):***

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.

#### ***Interactive System Productivity Facility (ISPF):***

ISPF 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 DATAMOVR 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 AUTOLOG function to log on the DATAMOVR 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 DATAMOVR virtual machine automates this process. Make sure that the DATAMOVR virtual machine is logged on whenever z/VM is initialized. Place the CP AUTOLOG command for DATAMOVR 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 user ID 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 CP FORMAT 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 user's directory entry 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 directory entries 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 product, this is the only way to maintain the directory. Manually updating the z/VM user directory could compromise system security and integrity.

Do not let any user directly access the z/VM user directory source file. With DIRMAINT and RACF installed, the only safe, legitimate way for an administrator to modify this file is by using dual registration (see the next recommendation).

- Use dual registration, an interface between DIRMAINT and RACF, to add, modify, or delete a user entry in the z/VM user directory and in the RACF/VM database with one transaction.

When you delete a user from the system using dual registration, take special care. Dual registration deletes the user ID from the z/VM user directory and from the main RACF database. It does not, however, delete all occurrences of the individual's user ID or associations with RACF groups. You must deal with these separately, according to instructions in *System Programming Library: RACF*. Use the ICHUT100 utility program to identify and locate all occurrences of the user ID in the RACF database.

The XEDIT option must be in effect with dual registration.

- Limit DIRMAINT privileged users. Strictly limit the number of user IDs you list in the DIRMAINT DIRM\_STAFF installation option. The users you list here can enter DIRMAINT privileged commands.
- Limit access to the DIRMAINT LOG. The DIRMAINT LOG, 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 LOG, limit the number of user IDs you associate with the DIRM\_STAFF option in the DIRMAINT DATA file. Only a user listed on this option 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 the *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 only to system administrators. Their use should be one of the audited privileges of an administrator. However, if the minidisk is simply 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 whose user ID is NOLOG. This forces the administrator to use the LINK command—an auditable and controllable event—to access the minidisk.

You should also audit the 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 may request information about their own minidisks without RACF authorization.

- 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 384.)
- Audit the OVERRIDE utility. The OVERRIDE utility lets you restructure 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. When the audit flag is set for DIAGNOSE code X'C4', an audit record is made when a user tries to restructure privilege classes.
- 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 the *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, diagnoses, 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 by using an Integrated Cryptographic Facility (an optional hardware feature).

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 the *z/VM: CMS File Pool Planning, Administration, and Operation* book.

### Using an External Security Manager for Auditing and Protecting

An external security manager (ESM) is a service virtual machine used to maintain VM/ESA 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 the *z/VM: CP Programming Services* book.

### 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 the *z/VM: CP Programming Services* book.

### 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 or the SYSCLR operand of the SYSRES macroinstruction 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,  
or
- when a temporary disk is detached from a user



if the ENABLE CLEAR\_TDISK operand is specified on the FEATURES system configuration file statement. Or, see “SYSRES (Required)” on page 693 for additional information on the SYSCLR operand of the SYSRES macroinstruction.

Class B system resource operators can query the state of temporary disk clearing by using QUERY TDISKCLR. See *z/VM: CP Commands and Utilities Reference* for additional details on the purpose and syntax of the QUERY TDISKCLR command.

**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.

**Compatibility Note:** z/VM temporary disk clearing is not compatible with VM/SP HPO temporary disk clearing. VM/SP HPO clears temporary disks only during DEFINE TDISK and issues messages to the user during the clearing. z/VM clears temporary disks during IPL, attach to system, and DETACH TDISK. Messages are issued only when CP volumes that contain temporary disk space are attached to the system.

**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 user’s directory entry 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’s directory entry. For additional information on these operands of the OPTION statement, see “OPTION Directory Control Statement (General)” on page 533.

### 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

## Security and Integrity in z/VM

- 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” on page 299.

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 160. For instructions on how to code this macroinstruction, see “SYSJRL (Optional)” on page 682. 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 2C2. 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

With VM/ESA, 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 control 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 control 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 control 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 control statement, see "OPTION Directory Control Statement (General)" on page 533. See "LINK Directory Control Statement (Device)" on page 500 and "MDISK Directory Control Statement (Device)" on page 513 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 136 for information on how to use the FEATURES PASWORDS\_ON\_CMDS operand.

## Using an Integrated Cryptographic Facility

The Integrated Cryptographic Facility (ICRF) is an optional feature that provides cryptographic capabilities for z/VM. z/VM allows multiple virtual machines to access the ICRF. The ICRF is available only to ESA mode virtual machines and only if the real processor has a real ICRF installed. (z/VM does not provide a software simulation of this hardware.) A processor complex can contain one or two ICRFs and the ICRFs can be used by one or more guest virtual machines. A processor can access only one ICRF, although each ICRF can be accessed by more than one processor.

## Security and Integrity in z/VM

A real ICRF can be shared among virtual machines if the real processor with which it is associated is not dedicated to a virtual processor of a virtual machine. If a real processor, with its associated ICRF, is dedicated to a virtual processor of a virtual machine, then that ICRF is unavailable to other virtual machines. This is true even if the virtual machine to which the processor is dedicated does not have a virtual ICRF defined for the virtual processor.

Virtual machines can make use of the real ICRF by means of one or more virtual ICRFs. The number of users that can access a real ICRF is limited by the number of cryptographic domains installed. An ICRF can have up to 16 cryptographic domains installed. Therefore, up to 16 users can have access to a virtual ICRF at any given time. If a user that is authorized for a certain cryptographic domain logs on and that domain is in use, CP issues an informational message stating that the domain is already in use by another user.

Each cryptographic domain consists of one master-key register and one auxiliary-master-key register. The master key can only be manually installed but is done with interaction with a program which uses the new cryptographic instructions. Users are assigned use of one or more domains. It is the users' responsibility to enter the keys and do key management. z/VM will not do any key management.

In a processor configuration with two ICRFs installed, the keys in each domain of each ICRF are entered separately. Because the keys are set in the ICRF by interaction with a program, that program must know which ICRF a processor accesses. The program can then dispatch work on a processor that has access to the ICRF whose key it is trying to set. When keys are changed it is up to the user to set the keys of each domain in both ICRFs. When the keys are different, cryptography work must only be dispatched on processors accessing the ICRF whose key is the value the program expects (the ICRF that contains the key the program has used to encipher its data).

Programs running in virtual processors have the same requirement. That is, the virtual processors must only be dispatched on real processors which have access to the ICRF (Crypto) containing the key they expect. If there are two real Crypto units available and the keys are the same in both Crypto units, a virtual CPU may be dispatched on either Crypto processor.

Virtual machines should be defined with the same number of virtual CPUs with Crypto, as the number of real Crypto processors available. For example, if there are two real Crypto units available then the virtual machine should have two virtual CPUs with Crypto defined. If the virtual machines are not defined like this the user will not be able to interact with the program to set the keys in both real Crypto processors. Likewise, if there is one real Crypto unit available, there would only be a need for one virtual CPU with Crypto.

To define two virtual CPUs with Crypto, you need to define a minimum of two virtual CPUs, a virtual Crypto on both virtual CPUs and specify that the virtual Cryptos be assigned to real Cryptos 0 and 1. The virtual Crypto units are defined either by CPU directory control statements or the DEFINE CPU and DEFINE CRYPTO commands. The virtual Crypto units should be assigned to real Cryptos with the CSU \* operand of the CRYPTO directory control statement. See "CRYPTO Directory Control Statement (General)" on page 468 for more information.

An example of the directory control statements:

```
CRYPTO DOMAIN 1 CSU *  
CPU 0 NOVECTOR CRYPTO  
CPU 1 NOVECTOR CRYPTO
```

In a real configuration with two ICRFs installed, if virtual machines are not defined with more than one ICRF, the user will not be able to interact with the program to set the keys in both real ICRFs. While not the recommended mode of operation, if this virtual configuration is desired, the CSU 0 or CSU 1 operand of the CRYPTO directory control statement can be used to specify which real ICRF the virtual ICRF should be assigned to.

The ICRF, like the Vector Facility, is an optional feature which may be installed on only some of the processors in a real processor complex. When a vector or crypto instruction is issued by a guest on a virtual CPU, z/VM assigns that virtual CPU affinity to a real processor that has the corresponding facility installed. If both vector and crypto instructions are being executed on a virtual CPU, z/VM would continually be switching the virtual CPU from one real processor to another since no real processor exists that can execute both sets of instructions. Therefore, virtual configurations should not be defined with vector and crypto facilities on the same virtual processor.

---

## 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 main 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 System/390 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 System/390 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

## Security and Integrity in z/VM

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 the *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 nonprivileged 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 re-entrant 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” on page 398 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 system 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. Since z/VM's integrity 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 multi-user 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 non-shared 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 the *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 he releases it. Temporary disks (T-disks) are the exception. operand on the FEATURES system configuration file statement or the SYSCLR option of the SYSRES macroinstruction, 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” on page 392 and “Sensitive DIAGNOSE Functions to Restrict” on page 394.) 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.

## Security and Integrity in z/VM

**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 non-disruptive 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 override the privilege class structure that comes with your system. To learn the details of tailoring your system's privilege class structure, see Chapter 16, "Redefining Command Privilege Classes," on page 433.

**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**

Displays the contents of real storage at the user's terminal.

### **DUMP HOST**

Prints the contents of real 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.

## Security and Integrity in z/VM

### 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

Lets a user alter the contents of real 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 real 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 system directory.

### DIAGNOSE X'4C'

Allows a user with the ACCT option in his directory entry 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 system 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 System Directory:** The CP system directory is a table maintained on DASD, each entry of which describes the configuration of a particular virtual machine in your system. To put it another way, the CP system directory tells z/VM exactly which system resources will be available to each virtual machine, how each should be built, and how access to each shall be governed.

The items in each directory entry 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 control statement, letting the user invoke the DIAGNOSE X'98' function.
- LNKNOPAS and D84PAS options on the OPTION control statement.
- POSIXINFO statement for the UID and GIDIGNAME
- 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.

## Security and Integrity in z/VM

**Directory Update-in-Place:** DIAGNOSE X'84' gives a privilege class B user the ability to replace data in the CP system 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 main storage. Most dumps are created by highly privileged users to help them solve system problems.

There is no way to predict what main 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 CMS's point of view, the composite reader file is many files; from z/VM's 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 entry in the CP system 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 Main Storage:** z/VM provides both fetch and store protection for real, main storage. No user may store in or fetch from a segment of main storage unless his key matches the protection key of that storage.

z/VM protects real, main storage using the privilege class structure. As the system is delivered, only privilege class C users can alter CP storage. Of course, your organization may have chosen to change the privilege class structure that IBM has built into your system.

**Allocation of Main Storage:** Real storage is allocated to a user when he first refers to a virtual page for which a real page has not yet been allocated. Allocation continues, as needed, up to and including the maximum storage size indicated in the user's CP system directory entry.

Before z/VM allocates a new, real page, that page 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 he has no authorization.

## Security and Integrity in z/VM

**Paging:** In z/VM, many users share main storage simultaneously. Not every page of storage could possibly be active every second of the time. When a page of main storage is not being actively accessed by a user, the system transfers it to expanded storage or DASD. This is known as paging out. The space in main storage that it abandons then becomes available to another user who really needs it. When and if that same page is needed again, the system transfers it from expanded storage or DASD back to main storage. This is known as paging in.

z/VM maintains a set of page and segment data tables that contain precise information on the location and owner of every page and segment of main storage, whether paged in or paged out. The system follows them rigorously, helping to maintain the integrity of the system.

**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 Temporary Minidisks:** You can also tailor your system to automatically clear each temporary minidisk (T-disk) before it is reassigned. Simply add the ENABLE CLEAR\_TDISK operand on the FEATURES system configuration file statement.

**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*

## Security and Integrity in z/VM

```
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 389. 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 13. Using the Stand-Alone Dump Utility

CP can produce several types 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 the stand-alone dump utility and use it. For more information on the stand-alone dump utility, debugging various types of problems, the other dump types, and other problem diagnosis tools, see the *z/VM: Diagnosis Guide*.

z/VM includes a stand-alone dump utility that you tailor according to your installation's configuration, using CMS. After you generate z/VM, you should create the stand-alone dump utility and place it on tape or a DASD for emergency use. If, after a system failure, CP cannot create an abend dump, you can use the stand-alone dump utility to dump all of storage.

This chapter describes how to create the stand-alone dump utility and provides reference information about the:

- HCPSADMP EXEC
- HCPSDC ASSEMBLE file.

---

### Overview

With the stand-alone dump utility, you can dump all of real storage when z/VM cannot create a CP abend dump. This utility dumps all resident, nonzero pages, CP and non-CP. It cannot dump virtual machine storage and nonresident pages from the paging device.

To use the stand-alone dump utility to dump real storage, you must have access to IPL the real machine. (This is usually done by the System Administrator, not a user.) You can IPL the stand-alone dump utility from tape or a DASD and direct the output to a tape. You may need to reserve several tapes to hold all the information. Basic error recovery is available for DASD and tape devices used as IPL or output devices.

Typically, an installation can have several stand-alone dump utilities generated and ready to run. It would be useful to have the following configurations available for the stand-alone dump utility:

- IPL from tape with output directed to tapes
- IPL from DASD with output directed to tapes

These configurations let you take a stand-alone dump with any of the supported possible environments.

---

### Creating the Stand-Alone Dump Utility

Your installation can generate the stand-alone dump utility to customize the facility to your system configuration. This gives you control over the device used to IPL the stand-alone dump utility and the output device for the dump. Invoke the HCPSADMP EXEC in a CMS virtual machine to do the generation.

Do not call the HCPSADMP EXEC from within another exec. Also, do not queue up the answers ahead of time when running the HCPSADMP EXEC.

## Using the Stand-Alone Dump Utility

To use the HCPSADMP EXEC:

- You must have read/write access to file mode A.
- The following files must exist on an accessed disk:
  - HCPSADMP EXEC
  - HCPGPI MACLIB
  - HCPPSI MACLIB
  - HCPOM1 MACLIB
  - HCPOM2 MACLIB
  - HCPVM CNTRL (the control file)
  - HCPLDR MODULE
  - HCPSADLD EXEC (the load list)
  - LDT HCPSADWT
  - HCPSAD TEXT
  - HCPSAH TEXT.
- You need the H assembler or the high-level assembler.

You are asked to answer a series of questions that describe the environment where the stand-alone dump utility will run. The HCPSADMP EXEC checks all input for validity and returns messages if you enter invalid data.

1. If the system has generated more than one stand-alone configuration, use unique names for each configuration. The active configuration must always be named HCPSDC. If you answer Y to create a new HCPSDC module, (refer to “Example for Generating the Stand-Alone Dump Utility” on page 406), the original is erased.
2. If you respond with N to any of the questions (refer to “Example for Generating the Stand-Alone Dump Utility” on page 406), the exec goes directly to the next question without doing the indicated work.
3. When the exec asks you to enter the real output device number, you are limited to four digits for z/VM.

An example of the prompts and replies that appear on the virtual machine console during HCPSADMP EXEC execution is shown in “Example for Generating the Stand-Alone Dump Utility” on page 406.

Following the data that you provide, the HCPSADMP EXEC:

- Creates a file called HCPSDC ASSEMBLE and writes the file on your A disk
- Assembles the HCPSDC ASSEMBLE file to create the HCPSDC TEXT file and link-edits it with the stand-alone dump utility module
- Writes the stand-alone dump utility load module on a real device.

At the end of this procedure, the stand-alone dump utility resides on auxiliary storage (tape or disk) ready for loading at any time.

## Before You Begin

Before you create the stand-alone dump utility, note the following:

- If you use a DASD as the IPL device, the stand-alone dump utility takes up cylinder 0. The volume you use as the IPL device may be any CP-formatted DASD volume *except* the system residence (SYSRES) volume. The cylinder

must be formatted as PERM space; use ICKDSF. The remaining cylinders may be formatted and used for any CP function *other* than system residence functions.

- If the volume you are going to use as your IPL device is shared through cross system link, the data created for cross system link by the XLINK FORMAT command must be placed in a cylinder other than cylinder 0 (its default location). The CSELVOL macroinstruction may be used to define an alternative location for cross system link data for specified volumes.
- Any IPL device (tape or DASD) must be write-enabled.
- When you create the stand-alone dump utility, you can specify up to eight dump devices. With one exception, the stand-alone dump utility chooses the first ready device as a dump device. The exception is the IPL device. If a dump device is also the IPL device, the stand-alone dump utility chooses any other ready dump device first.
- Choosing a dump device:
  - You may wish to include tape drives on different tape control units. Otherwise, if the tape control unit for the stand-alone dump devices is inoperative at the time of failure, you will not be able to run the utility.
  - To increase the amount of dump data the stand-alone dump utility can write on one tape, choose tape drives that can write the most data per tape reel or cartridge.

**Note:** It is suggested that the 3590 be used for central storage sizes of 2Gig or more due to the 3590's speed and high data capacity.

**Attention:** The stand-alone dump utility does *not* support device sharing with other processor complexes. Therefore, when you are ready to use the stand-alone dump utility, you should make sure no other system on any other processor complex is using any of the possible dump devices.

## Devices You Can Use to IPL a Stand-Alone Dump

The following are the devices you can use to IPL the stand-alone dump utility:

<b>DASD</b>	3380 and 3390
<b>Tape</b>	3422, 3480, 3490 and 3590

### Notes:

1. If you select a DASD as the IPL device:
  - a. It cannot be a device that has the Stand-Alone Program Loader (SAPL) on it.
  - b. It cannot be the resident system device.
  - c. It must be CP-formatted by using either a CPFMTXA command or ICKDSF (recommended).
  - d. Cylinder 0 must be allocated as permanent space.
  - e. The stand-alone dump uses cylinder 0.
  - f. The remaining cylinders may be formatted and used for any CP function other than system residence functions.
2. The stand-alone dump IPL tape can be the same as the tape to which you direct the dump output. If you choose to use the same tape, the dump will NOT be written over the stand-alone dump utility. That is, the same tape can be IPLed to take a dump at a later time.

## Using the Stand-Alone Dump Utility

3. Do not try to IPL from a device that is not in the previous list. The stand-alone dump utility does not support FBA DASD.

## Devices to Which You Can Send Dump Output

The following are the devices to which the dump output can be directed:

**Tape** 3422, 3480, 3490 and 3590

### Notes:

1. You can specify a maximum of eight real device numbers for the dump output device. To increase the amount of dump data the stand-alone dump utility can write on a tape, choose tape drives that can write the most data per tape reel or cartridge. You can enter up to four digits for each real output device number. It is suggested that the 3590 be used for central storage sizes of 2Gig or more due to the 3590's speed and high data capacity.
2. When you configure the stand-alone dump utility, you can use any tape device type from the list of supported devices. You may wish to include tape drives on different control units. Otherwise, if the tape control unit is inoperative at the time of the failure, you will not be able to run the program.
3. All 3480 tape devices save tape data in non-compact format. All 3490 and 3590 tape devices save tape data in compact format.
4. Do not send the stand-alone dump output to a device that is not included in the previous list.

The stand-alone dump utility selects, as the dump output device, the first available device in the list, excluding the IPL device (if it is in the list). If you want the stand-alone dump output to go to the IPL tape, make all other devices that are in the list not ready. If no other device within the output address list is available and the IPL tape address is in the list, the IPL device receives the dump.

If you select a tape for the dump output device, other than the IPL tape, the stand-alone dump utility:

1. Rewinds the tape to ensure that the dump is at the beginning of the tape
2. Sets the density to the highest value for the tape device.

If the tape device selected is the one on which the stand-alone dump utility resides, the utility writes the dump at the same density as the stand-alone dump utility was written.

When using tape, reserve the complete tape for the stand-alone dump utility; do not put the stand-alone dump utility on a tape with the other stand-alone utilities.

Use a single-volume, nonlabeled tape for the stand-alone dump utility. Be sure that the tape is nonlabeled, because the utility does not check to ensure that it is a nonlabeled tape.

## Example for Generating the Stand-Alone Dump Utility

The following is an example of generating the stand-alone dump utility. In this example:

- You are placing the stand-alone dump utility on a 3390 DASD device with an address of 150.
- The control file in this example is HCPVM.
- The system sends the dump output to the first available 3490 tape drive whose address is 570, 571, 572, 573, 574, 575, 576, or 577.

## Using the Stand-Alone Dump Utility

- The HCPSDC ASSEMBLE file is created and contains the HCPSDCMP macroinstruction.
- It is assumed that the stand-alone dump utility residence volume is attached to your virtual machine and that you have the appropriate minidisk accesses.

**Note:** To ensure the proper access order, use the UTILITY EXEC (see the *z/VM: Guide for Automated Installation and Service*).

1. Use the CP ATTACH command to attach logically the stand-alone dump utility residence device to your virtual machine. For example, if you enter:  

```
attach rdev * vdev
```

the command attaches the device specified by *rdev* to the virtual machine.
2. Establish the appropriate minidisk accesses by entering:  

```
vmfsetup esa cp
```
3. Ensure that the HCPSADMP EXEC and HCPVM CNTRL files to be used are on a minidisk accessed earliest in the user's search order. HCPSADMP EXEC uses the HCPVM CNTRL file.

Under CMS, enter:

```
hcpsadmp
```

The exec replies:

THE HCPSADMP EXEC:

```
-OPTIONALLY CREATES A NEW HCPSDC ASSEMBLE FILE  
CONTAINING AN HCPSDCMP MACRO WITH SELECTED PARAMETERS  
ON YOUR A DISK. THE NEWLY CREATED MODULE WILL BE  
REASSEMBLED.  
-OPTIONALLY PLACES A LINK-EDITED HCPSAD MODULE (HCPSAD,  
HCPSAH AND HCPSDC) ONTO WHATEVER DEVICE IS DEFINED AS THE  
RESIDENCE DEVICE, WHERE THE RESIDENCE DEVICE IS THE  
TARGET DEVICE ON WHICH THE STAND-ALONE DUMP UTILITY  
WILL RESIDE.
```

```
PLEASE RESPOND YES, NO (OR END TO EXIT EXEC) WHEN IT  
PROMPTS YOU. IF YOU RESPOND NO, YOU MUST HAVE A HCPSDC  
TEXT FILE FROM A PRIOR GENERATION OF THE STAND-ALONE  
DUMP UTILITY.
```

```
DO YOU WANT TO CREATE A NEW HCPSDC MODULE?
```

Enter:

```
yes
```

```
PLEASE SELECT RESIDENCE DEVICE TYPE  
(3350, 3375, 3380, 3390, 9345, 3420, 3422,  
3424, 3430, 3480, 3490, 3590, 9348 or END):
```

```
9345
```

```
PLEASE ENTER RESIDENCE DEVICE NUMBER:
```

```
150
```

```
PLEASE SELECT OUTPUT DEVICE TYPE:  
(3420, 3422, 3424, 3430, 3480, 3490, 3590, 9348 or END).
```

```
3420
```

## Using the Stand-Alone Dump Utility

```
PLEASE SELECT OUTPUT DEVICE NUMBER OR LIST:  
A LIST MUST BE PLACED WITHIN  
PARENTHESES AND THE ADDRESSES  
SEPARATED BY COMMAS
```

```
(570,571,572,573,574,575,576,577)
```

```
ASSEMBLING HCPSDC
```

```
HCPSDC TEXT CREATED PRT FILE 4584 SENT FROM MAINT PRT AS 1234 RECS 0056  
COPY 001 HOLD NOKEEP
```

```
DO YOU WANT TO PLACE THE STAND  
ALONE DUMP ONTO 150?
```

```
yes
```

```
LOAD LIST: HCPSADLD EXEC A2 mm/dd/yy hh:mm (MNT190)
```

These messages inform you about the load list that the exec is using.

```
CPU STOPPED; DISABLED WAIT PSW 000A0000 00008200
```

When the HCPSADMP EXEC completes successfully, your virtual machine enters a disabled wait state, code 8200.

---

## Taking a Stand-Alone Dump

If you plan to dump 16 MB of storage on a 1/2-inch reel, use a tape density of 1600 or 6250 BPI. A 16 MB dump may not fit on a tape at 800 BPI.

To invoke the stand-alone dump:

1. For multiple processor systems, stop all tightly coupled processors. Do *not* clear storage.
2. For multiple-processor systems, select the processor with the I/O configuration that has access to the resident volume address and the output device address(es).
3. Display locations X'0' to X'18' at the console. The stand-alone dump IPL sequence overlays these bytes, so they cannot be recovered.
4. Do a STORE STATUS operation on the CPU where you will IPL the stand-alone dump utility. If you do not do the STORE STATUS, the following are not saved in low storage:
  - CPU timer
  - Clock comparator
  - Current PSW
  - Prefix
  - Model-dependent features
  - Control registers
  - Floating-point registers 0, 2, 4, and 6
  - Access registers
  - General-purpose registers.

If the prefix value is not saved in low storage, the information from the prefix page is not available for the formatted section of the dump.

5. If the stand-alone dump utility resides on a tape, then mount and ready the volume, making sure that the tape is write enabled.
6. Ready the output device. If you want the system to place the stand-alone dump on the IPL tape, make all other tapes listed as output devices (at generation



time) *not* ready. If you do not want the stand-alone dump on a device that is listed as a possible output device, the device must not be ready at the time you IPL the stand-alone dump utility.

7. IPL the stand-alone dump utility.

**Note:** Do *not* use the LOAD CLEAR command to load the stand-alone dump utility. This command destroys the storage contents you are trying to dump.

Initially, the stand-alone dump utility writes the first seventeen pages of storage to the IPL device. This provides an area to load the stand-alone dump utility and work space. (See “Tape Format” on page 777 and “DASD Format” on page 779 for information about the tape and DASD format.) This step causes the system to place the dump on the output tape.

8. After the CPU has gone into a wait state, display the PSW to determine the status. A wait state of 8200 indicates that the stand-alone dump has successfully completed. If the stand-alone dump utility is unsuccessful because of some error that you can fix (for example, an I/O error on the output tape):
  - a. Correct the error.
  - b. Invoke a hardware RESTART to restart the stand-alone dump.

If you re-IPL the stand-alone dump utility again, part or all of the first six pages of storage will be invalid. After the initial IPL, you cannot change the IPL address or IPL volume. See the *z/VM: System Operation* book for a complete description on the different wait state codes.

---

## Processing the Stand-Alone Dump Data on Tape

Re-IPL z/VM. Use the CP DUMpload utility to create a CMS file from the dump on the tape, and then use the VM Dump Tool to process the dump. For details on using DUMpload see the *z/VM: CP Commands and Utilities Reference*. For details on using the VM Dump Tool, see the *z/VM: VM Dump Tool* book.

---

## The HCPSADMP EXEC

The HCPSADMP EXEC creates a load module of the stand-alone dump utility and writes the module on a real device. To invoke this exec, enter:

```
hcpsadmp
```

The exec command line has no operands. It prompts you for the information it needs. See “Creating the Stand-Alone Dump Utility” on page 403.

## Usage Notes

1. The IPL device, the device from which the stand-alone dump utility is loaded, cannot be the CP system residence device.
2. If the IPL device is a tape, the stand-alone dump utility supports that same tape as a dump device, the device to which dumps are sent.
3. This utility does not support FBA DASD devices. For device support information, refer to “Devices You Can Use to IPL a Stand-Alone Dump” on page 405 and “Devices to Which You Can Send Dump Output” on page 406.
4. When the HCPSADMP EXEC completes successfully, your virtual machine enters a disabled wait state, code 8200.

## Using the Stand-Alone Dump Utility

---

## Chapter 14. 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 XTB1xxxx 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 al
```

```
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.)

## Creating and Modifying Image Libraries

*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 the *z/VM: CP Commands and Utilities Reference* book. 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 tapes or CD-ROM, 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 the *z/VM: General Information* book.

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," "Forms Control Buffers Supplied by IBM" on page 418, and "Naming Conventions for UCS Buffer Images and FCBs" on page 416.
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:

<b>Name</b>	<b>Meaning</b>
<b>AN</b>	Normal alphanumeric notation character set arrangement
<b>HN</b>	Normal hexadecimal notation character set arrangement
<b>PCAN</b>	Preferred alphanumeric notation character set arrangement
<b>PCHN</b>	Preferred hexadecimal notation character set arrangement
<b>PN</b>	PL/I—60 graphics
<b>QN</b>	PL/I—60 graphics
<b>QNC</b>	PL/I—60 graphics
<b>RN</b>	FORTRAN, COBOL commercial
<b>SN</b>	Text printing 84 graphics
<b>TN</b>	Text printing 120 graphics
<b>YN</b>	High-speed alphanumeric

IBM also supplies the following UCS buffer images for devices that emulate the 3211 printer.

<b>Name</b>	<b>Meaning</b>
<b>A11</b>	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
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

## Creating and Modifying Image Libraries

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 422.

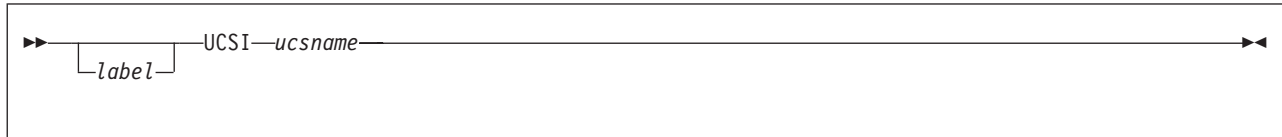
**Note:** Only one UCS may be defined in an ASSEMBLE file.

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:

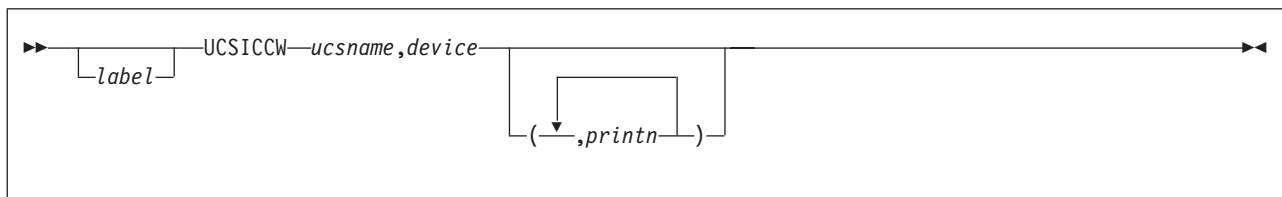


*uciname*

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 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:



*uciname*

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.

## Creating and Modifying Image Libraries

**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 422.

### **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 HCPCW0EQ
          COPY HCPEQUAT
          SPACE
EX01      CSECT ,
          UCSI      EX01
          DC        5C'1234567890A...Z1234567890*/'
          DC        X'0010101010101010101010004000404000' 240-255
          DC        X'401010101010101010101010004040400000' 256-271
          DC        X'404010101010101010101010004000000000' 272-287
          DC        X'101010101010101010101010000000404000' 288-303
          UCSICW    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 HCPCW0EQ
        COPY HCPEQUAT
        SPACE
EX02    CSECT ,
        UCSI    NUM1
        DC      24C'1234567890'
        DC      X'00101010101010101010100004000404000'    240-255
        DC      X'4010101010101010101010100004040400000'    256-271
        DC      X'4040101010101010101010100004000000000'    272-287
        DC      X'1010101010101010101010100000000404000'    288-303
        UCSICW 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 HCPCW0EQ
        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'000000000000000000000000000000101010'    436-450
        DC      X'101010101010100040404240004010'    451-465
        DC      X'1010101010101000404041000040'    466-480
        DC      X'4010101010101010100040400000000'    481-495
        DC      X'101010101010101010100040404448'    496-510
        DC      X'0000'    511-512
        UCSICW ALL,3211,(48,48,48,48,48,48,48,48,48)
        END      ALL
    
```

Note that the DC specification contains 49 characters and the UCSICW macroinstruction specifies 48 characters. The ampersand (&) must be coded twice to be accepted by the assembler. The single quote (') 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:

## Creating and Modifying Image Libraries

```
P48      CSECT
        SPACE
        COPY HCPCW0EQ
        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
UCSICW 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.

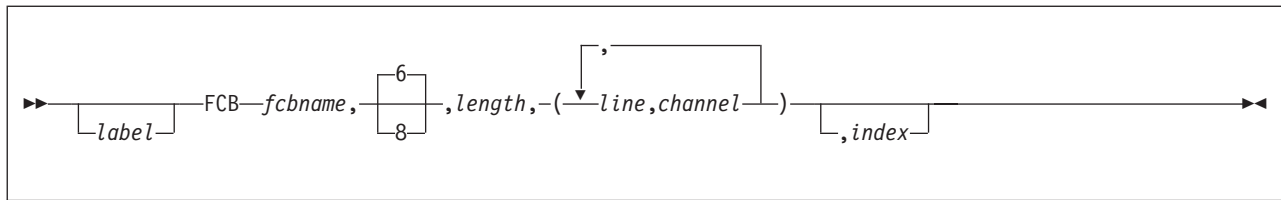
For example, the FCB1 buffer image associated with the 4248 printer is located in 4248FCB1 ASSEMBLE.

### 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 416. 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 422.

### 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 HPCPSB241E will be issued when you try to load any FCB other than 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

## Creating and Modifying Image Libraries

- 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.

```
fcbl spec,8,60,(1,1,40,12,60,9),10
```

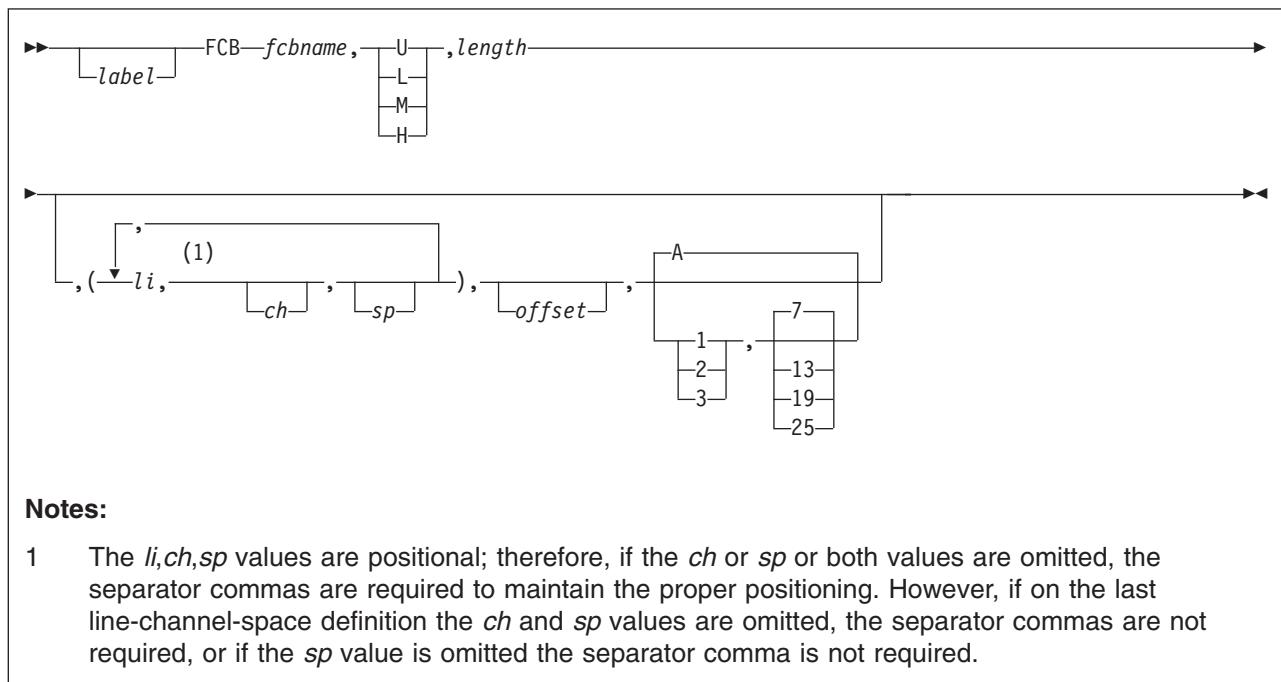
If you want another forms control buffer, called LONG, to be exactly the same as SPEC with the exception that only six lines are printed per inch, code either of the following:

```
fcbl long,6,60,(1,1,40,12,60,9),10
```

```
fcbl 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:



#### *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
<b>U</b>	Unchanged
<b>L</b>	Low
<b>M</b>	Medium
<b>H</b>	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:

<b>Value</b>	<b>Meaning</b>
<b>A</b>	Automatic stacker level control, which is the default
<b>1</b>	Paper tray lowered 1 inch below automatic position
<b>2</b>	Paper tray lowered 2 inches below automatic position
<b>3</b>	Paper tray lowered 3 inches below automatic position

### *rate*

specifies the paper stacker drop rate. Acceptable values are:

<b>Value</b>	<b>Meaning</b>
<b>7</b>	Drop the paper tray after seven sheets, which is the default
<b>13</b>	Drop the paper tray after 13 sheets
<b>19</b>	Drop the paper tray after 19 sheets
<b>25</b>	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 HPCSB241E will be issued when you try to load any FCB other than 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*

## Creating and Modifying Image Libraries

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:

```
fcbl 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:

```
fcbl 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* book. For information on image libraries, see "Installing Your Own Image Library" on page 423.

---

## 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 tapes or CD-ROM, 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” below describes how to do this.

For additional information on the IMAGELIB utility, refer to the *z/VM: CP Commands and Utilities Reference* book.

### 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 the *z/VM: CP Commands and Utilities Reference* book.

## Creating and Modifying Image Libraries

### 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 the *z/VM: CP Commands and Utilities Reference* book.

#### 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 411.

#### 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 the *z/VM: CP Commands and Utilities Reference* book.

### 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 the *z/VM: System Operation* or *z/VM: CP Commands and Utilities Reference* books.

---

### 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 the *z/VM: CP Commands and Utilities Reference* book.

For information on operating a 3800 printer as a real spooling device, see the *z/VM: System Operation* book.

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*

## Creating and Modifying Image Libraries

- *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 15. 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 14 shows the table module's file name and the macroinstruction name associated with each device class.

*Table 14. Table and Macroinstruction Names for Device Classes*

<b>Table</b>	<b>Macroinstruction</b>	<b>Device Class</b>
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 14 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.

## CCW Translation

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 429 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.

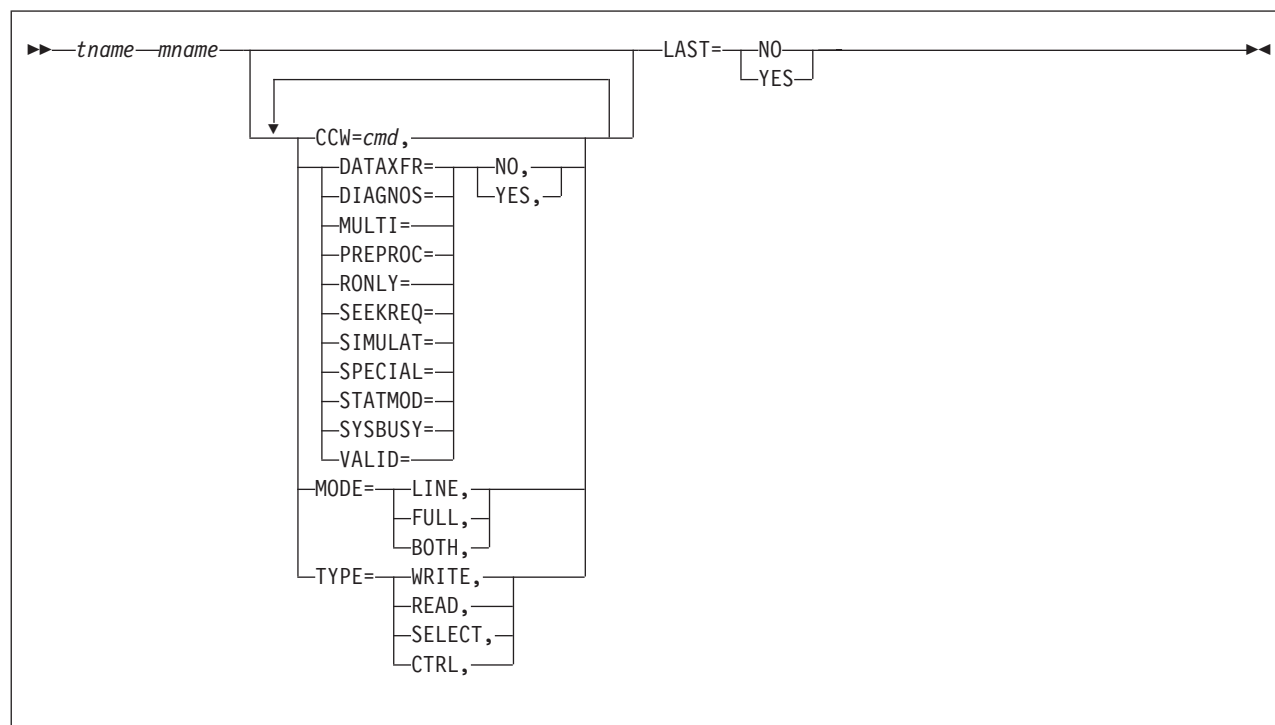
---

## 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 14 on page 427. 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'.

### **DATAFER=NO**

### **DATAFER=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.

## CCW Translation

**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.

---

## Part 3. User Planning and Administration

<b>Chapter 16. Redefining Command Privilege Classes</b>	433
IBM-Defined Privilege Classes	433
Planning a New User Class Structure	434
Defining Users' Needs	434
Assigning Commands to Types of Users	435
Associating Privilege Classes with Users and Commands	436
Further Considerations	437
Preparing and Activating the Override File	438
Creating a Class Override File	438
Class Override File Example	439
Verifying and Activating the Override File	441
Querying the User Class Restructure File	442
Changing Back to the IBM-Defined User Classes	442
Another Way of Changing the Privilege Class of Certain CP Functions	443
Changing the Directory	443
Defining Privilege Classes for a Virtual Machine	444
Changing the Setting for a Logged-On Virtual Machine	444
Changing the User Directory	444
How Users Can Find Which Commands They Can Enter	446
<b>Chapter 17. Creating and Updating a User Directory</b>	447
Overview	447
Creating the User Directory	451
Continued Directory Control Statements	452
Quoted String Operands	452
Running the User Directory Program	453
Changing the Directory	454
Checking a Directory for Errors	454
Specifying a Directory That Contains VM/SP Control Statements	455
Creating Directory Profiles	455
Determining How Much Space the Directory Needs	455
ACCOUNT Directory Control Statement (General)	456
ACIGROUP Directory Control Statement (General)	458
APPCPASS Directory Control Statement (General)	459
AUTOLOG Directory Control Statement (General)	461
CLASS Directory Control Statement (General)	462
CONSOLE Directory Control Statement (Device)	463
CPU Directory Control Statement (General)	465
CRYPTO Directory Control Statement (General)	468
DASDOPT Directory Control Statement (Device)	472
DATEFORMAT Directory Control Statement (General)	475
DEDICATE Directory Control Statement (Device)	477
DIRECTORY Directory Control Statement (Control)	480
D8ONECMD Directory Control Statement (General)	483
GLOBALDEFS Directory Control Statement (Control)	485
GLOBALOPTS Directory Control Statement (Control)	486
INCLUDE Directory Control Statement (General)	487
IOPRIORITY Directory Control Statement (General)	488
IPL Directory Control Statement (General)	490
IUCV Directory Control Statement (General)	493
LINK Directory Control Statement (Device)	500
LOAD Directory Control Statement (Control)	504
LOADDEV Directory Control Statement (General)	507

LOGONBY Directory Control Statement (General)	509
MACHINE Directory Control Statement (General)	510
MAXSTORAGE Directory Control Statement (General)	512
MDISK Directory Control Statement (Device)	513
MINIOPT Directory Control Statement (Device)	524
NAMESAVE Directory Control Statement (General)	527
NICDEF Directory Control Statement (Device)	529
NOPDATA Directory Control Statement (General)	532
OPTION Directory Control Statement (General)	533
POOL Directory Control Statement (General)	540
POSIXGLIST Directory Control Statement (General)	541
POSIXGROUP Directory Control Statement (Control)	543
POSIXINFO Directory Control Statement (General)	545
POSIXOPT Directory Control Statement (General)	548
PROFILE Directory Control Statement (Control)	551
SCREEN Directory Control Statement (General)	553
SHARE Directory Control Statement (General)	555
SPECIAL Directory Control Statement (Device)	557
SPOOL Directory Control Statement (Device)	562
SPOOLFILE Directory Control Statement (General)	565
STDEVOPT Directory Control Statement (General)	566
STORAGE Directory Control Statement (General)	568
SYSAFFIN Directory Control Statement (Control)	569
USER Directory Control Statement (Control)	572
XAUTOLOG Directory Control Statement (General)	578
XCONFIG Directory Control Statement (General)	580
XCONFIG ACCESSLIST Operand	581
XCONFIG ADDRSPACE Operand	583
XSTORE Directory Control Statement (General)	585



---

## Chapter 16. Redefining Command Privilege Classes

The user class restructure feature allows an installation to 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.

**Note:** An installation can place override requests into a system configuration file by using MODIFY statements. However, you should use either the OVERRIDE utility or the MODIFY statements. If any MODIFY statement is detected at IPL time, the UCR (User Class Restructure) override changes will be bypassed. IBM recommends using the new MODIFY statements. For more information on this method of extending the privilege-class structure of a system, see the *z/VM: CP Exit Customization* book.

This chapter tells you how to:

- Use the user class restructure feature
- Plan a new user class structure
- Prepare and activate an override file
- Display information about the current user class structure
- Change back to the default IBM-defined class structure
- Change the user directories.

To redefine your user class structure, do the following:

1. Plan your new structure by identifying each user group and the functions that they need to perform.
2. Prepare an override file. This is a CMS file that contains a list of all commands, DIAGNOSE codes, and system functions for which you wish to define new classes.
3. Update the directory to assign the new classes to each user's virtual machine.

These steps are described in more detail in the sections that follow.

---

### 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 system directory assigns each user one or more privilege classes as part of the user's entry 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 15 on page 434. 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

## Redefining Command Privilege Classes

controlled by directory control statements; see “USER Directory Control Statement (Control)” on page 572 for more information about NOLOG users.

Table 15. IBM-Defined Privilege Classes

Class	User and Function
A	<i>System Operator:</i> 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. <b>Note:</b> The class A user who is automatically logged on during CP initialization is designated as the primary system operator.
B	<i>System Resource Operator:</i> 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	<i>System Programmer:</i> The class C user updates or changes system-wide parameters of the z/VM system.
D	<i>Spooling Operator:</i> The class D user controls spool files and the system's real reader, printer, and punch equipment allocated to spooling use.
E	<i>System Analyst:</i> The class E user examines and saves system operation data in specified z/VM storage areas.
F	<i>Service Representative:</i> The class F user obtains, and examines in detail, data about input and output devices connected to the z/VM system. This class is reserved for IBM use.
G	<i>General User:</i> The class G user controls functions associate 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.
H	Reserved for IBM use.
I - Z 1 - 6	Classes I through Z are reserved for redefinition through user class restructure as needed by each installation.

---

## 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 16 on page 435.

Table 16. 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 17 on page 436.

## Redefining Command Privilege Classes

Table 17. 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		*	*	*	*					
SYSDFLT	G									*	
SYSOPR	A		*	*							
SYSSERV	F					*					

### 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 the *z/VM: CP Commands and Utilities Reference* book. 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 control statement that does the same function. For example, if there is a link control statement in a user's directory, that control 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 18 on page 437.

Table 18. 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					
SYSDFLT	G	P								P	
SYSOPR	A	IJ	I	J							
SYSSERV	F	L				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 restructure 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 the *z/VM: CMS User's Guide* for information on tailoring the HELP Facility. **Documentation Considerations**

## Redefining Command Privilege Classes

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.

---

## Preparing and Activating the Override File

To assign new privilege classes to commands, DIAGNOSE codes, and system functions, you must:

1. Create a class override file
2. Verify the class override file syntax
3. Activate the class override file immediately or at the next IPL.

## Creating a Class Override File

To override the IBM-defined privilege classes for commands, DIAGNOSE codes, and system functions, first create a class override file. (The CP `OVERRIDE` utility uses the default file name and file type of `CLASS OVERRIDE`. The `OVERRIDE` utility is described in the *z/VM: CP Commands and Utilities Reference*). This is a standard CMS file (fixed format, 80-characters wide) that contains control statements that override privilege class assignments.

The privilege classes for the system functions are defined by the `PRIV_CLASSES` system configuration file statement. For more information, see “`PRIV_CLASSES Statement`” on page 184 or “`SYSFCN (Optional)`” on page 673.

Statements in the override file have this format:

► `name`—NEWCLASS=`class` (1) IBMCLASS=`class`—►

### Notes:

- 1 NEWCLASS and IBMCLASS can be specified in any order. IBMCLASS is necessary only for commands that have more than one IBM-defined class. It is not required for DIAGNOSE codes and system functions.

The first field of each `OVERRIDE` control statement is the name of the command, DIAGNOSE code (using the form `DIAGxx`) or system function (using the form `SYSxxxx`—either `SYSOPER`, `SYSOPRD`, `SYSOPWT`, `SYSOSERV`, or `SYSDFLT`). These system functions are described in Appendix B, “Defining Your System (HCP) Macroinstructions,” on page 643. Make sure that this field appears first on the control statement line, with no other characters preceding it.

**Note:** Commands defined by the system as class `ANY` are not valid on the `OVERRIDE` control statement.

The next two fields may appear in either order:

- The `NEWCLASS=` parameter, which specifies the new privilege class to be assigned. You can specify more than one new class with this parameter. For example, the statement `NEWCLASS=HIJKL` defines classes H, I, J, K, and L. If a command, diagnose code, or system function is assigned to more than one user class, you must specify all the classes in the `NEWCLASS` parameter.

Example:

1. **Adding user classes to a DIAGNOSE code:** To add classes A and B to DIAGNOSE code X'04', code the control statement as follows:

```
DIAG04 NEWCLASS=ABCE
```

The IBM defined class is C and E; therefore, by specifying NEWCLASS=ABCE, you are adding classes A and B to the already existing classes C and E.

- The IBMCLASS= parameter, which specifies the original IBM-defined class for that command. Use this parameter only if the command has more than one IBM-defined class. It is not required for commands that have only one IBM-defined privilege class, for DIAGNOSE codes, or for system functions. You can specify only one class with this parameter, as the examples below illustrate.

Examples:

1. **Changing one class to another:** To change the PURGE command from classes A and G to A and U, code the control statement as follows:

```
PURGE NEWCLASS=U IBMCLASS=G
```

This would simply replace the old class G with the new class U. The class A command would remain unchanged.

2. **Adding a class:** To change the PURGE command from classes A and G to A, G, and Y, code the control statement as follows:

```
PURGE NEWCLASS=AY IBMCLASS=A
```

or

```
PURGE NEWCLASS=GY IBMCLASS=G
```

Your choice between these two possibilities would depend on which version of the PURGE command you wish to allow the class Y user to use, either the powerful IBM class A command or the general class G command.

3. **Removing a class:** To change the PURGE command from classes A and G to just A, choose a dummy class (we use Z here) and code the control statement as follows:

```
PURGE NEWCLASS=Z IBMCLASS=G
```

If you never give any user access to class Z commands, this control statement effectively removes the class G PURGE command from your class structure.

### Class Override File Example

Using our example of the insurance company from the previous section (Table 18 on page 437), the override file would contain the entries shown in Figure 28 on page 440.

## Redefining Command Privilege Classes

```
*
* CP COMMAND OVERRIDES FOR 'OUR INSURANCE COMPANY'
*
* USER CLASSES REPRESENT:
*
*     I = SYSTEM ADMINISTRATOR (SAD)
*     J = SENIOR SYSTEM PROGRAMMER (SSP)
*     K = JUNIOR SYSTEM PROGRAMMER (JSP)
*     L = SYSTEM ANALYST (SA)
*     M = SYSTEM OPERATOR (SO)
*     N = DATABASE ADMINISTRATOR (DBA)
*     O = SERVICE ENGINEER (SE)
*     P = COMPLEX USERS (U1)
*     Q = GENERAL USERS (U2)
*
ACNT      NEWCLASS=L
ATTN      NEWCLASS=PQ
XAUTOLOG  NEWCLASS=IJ      IBMCLASS=A
XAUTOLOG  NEWCLASS=KLMN   IBMCLASS=B
XAUTOLOG  NEWCLASS=P       IBMCLASS=G
CHANGE    NEWCLASS=MN     IBMCLASS=D
CHANGE    NEWCLASS=PQ     IBMCLASS=G
DEDICATE  NEWCLASS=J
DEFINE    NEWCLASS=M       IBMCLASS=A
DEFINE    NEWCLASS=PQ     IBMCLASS=G
IPL       NEWCLASS=PQ
MESSAGE   NEWCLASS=JK     IBMCLASS=A
MESSAGE   NEWCLASS=M       IBMCLASS=B
QUERY TIME NEWCLASS=I
SAVESYS   NEWCLASS=IJ
SET       NEWCLASS=J       IBMCLASS=G
SPOOL    NEWCLASS=PQ
DIAG04   NEWCLASS=IJKM
DIAG3C   NEWCLASS=IJKL
SYSDFLT  NEWCLASS=P
SYSOPR   NEWCLASS=IJ
SYSSERV  NEWCLASS=L
```

Figure 28. Class Override File Example

The privilege classes of the individual operands of the QUERY and SET commands can be changed independently of the other operands for those commands. To do this, specify the command and operand of any QUERY or SET command to be restructured. In the insurance company example, the QUERY TIME command will be changed to a new class I command. To change the privilege class for an entire set of QUERY or SET commands with the same IBM-defined privilege class, use the IBMCLASS= operand as in the above example. Here, all IBM-defined class G SET commands are changed to new class J.

You must code the IBMCLASS= operand for XAUTOLOG, CHANGE, DEFINE, and MESSAGE, as they are associated with more than one class. To assign more than one class to a command or command type, all new classes are placed on the same override control statement.

**Note:** Restricting the user class on a console command does not restrict the function of the directory control statement that does the same function. For example, if there is a LINK control statement in a user's directory, that control statement takes effect at logon, even though you have restricted the LINK command.



## Verifying and Activating the Override File

To verify that the control statements in the class override file have the correct syntax (without actually activating the new class structure), enter `OVERRIDE` (IBM-defined privilege classes A, B, or C) with the `VALIDATE` option:

```
override fn ft fm (validate
```

where the *fn ft fm* identifies the class override file. The `OVERRIDE` utility is described in the *z/VM: CP Commands and Utilities Reference*; the default is `CLASS OVERRIDE *`. If an error is detected, the statement in error is displayed, and a message informs you what the error is.

After you create the class override file and verify the syntax of the control statements in it, enter `OVERRIDE` with either the `IMMEDIATE` option or the `DEFER` option, as follows:

```
override fn ft fm (immediate
```

or

```
override fn ft fm (defer
```

where the file name (*fn*), file type (*ft*), and file mode (*fm*) identify the class override file. (Note that the file mode can represent an accessed minidisk or an accessed shared file system directory, and that the default is `CLASS OVERRIDE *`.)

If you enter the `OVERRIDE` with no options, the `DEFER` option (the default) is assumed.

The `OVERRIDE` utility's processing checks for errors using a two-pass process, as follows:

1. On the first pass, syntax errors (such as incorrectly placed options or improperly specified new classes) are flagged with a message informing you of the error. No matter how you entered `OVERRIDE` (with `DEFER`, `IMMEDIATE`, or `VALIDATE`), processing continues in `VALIDATE` mode.
2. On the second pass, other errors (such as incorrectly specified IBM-defined classes or misspelled command names) are flagged with a message identifying the incorrect line. Processing continues in whichever mode (`DEFER`, `IMMEDIATE`, or `VALIDATE`) you specified.

Whereas the class override file is the file you create to specify your class structure changes, it is the user class restructure (UCR) file that actually activates those changes. The UCR file is a system data file (SDF). It is acted on through the options on the `OVERRIDE` utility, as Table 19 shows.

Table 19. *OVERRIDE* Utility Options

Option	Action
DEFER (the default)	Information from the class override file is written to a user class restructure (UCR) file. This is a class D file, meaning that the new class structure will take effect at the next system IPL. If an old UCR file already exists, it remains in effect until the next system IPL.
IMMEDIATE	Information from the class override file is written to a UCR file. This is a class I file, meaning that the new class structure takes effect immediately, along with any changes that were previously in effect. However, the UCR file that is created will replace any existing UCR file.

## Redefining Command Privilege Classes

Table 19. *OVERRIDE* Utility Options (continued)

Option	Action
VALIDATE	The class override file is checked for syntax errors, but no UCR file is created.
CLEAR	The existing UCR file is removed at the next system IPL. You are therefore reverting to the IBM-defined user classes.

## Querying the User Class Restructure File

Use the QUERY UCR command to display information about the user class restructure file. Enter the command as follows:

```
query ucr all                (ALL is the default)
```

or

```
query ucr name filename    (to query by file name)
```

or

```
query ucr spoolid          (to query by spool ID)
```

The response will be displayed in the following format (it is assumed here that the override file was named CLASS OVERRIDE):

```
OWNERID  FILE TYPE CL RECS  DATE   TIME      FILENAME FILETYPE ORIGINID
*UCR     nnnn UCR  n nnnn mm/dd hh:mm:ss CLASS  OVERRIDE n
```

**Note:** If you enter QUERY UCR and see that a class D user restructure file exists, this tells you only that the file has been created using the DEFER option on the OVERRIDE utility. It does not tell you whether this file has been activated yet. (In other words, it does not tell you whether the new class structure described in that UCR file has been activated yet by a new system IPL.)

As an alternative, you can enter the QUERY UCR COUNT command to display the number of user class restructure files:

```
query ucr count
```

The response line will simply tell you how many user class restructure files there are. For example:

```
FILES: 0001 UCR
```

## Changing Back to the IBM-Defined User Classes

If you want to cause the commands to be assigned their IBM-defined privilege classes again, enter OVERRIDE with the CLEAR option, as follows:

```
override (clear
```

and then IPL the system.

Alternatively, you can use the PURGE UCR command, as follows:

```
purge ucr name filename    (to purge a file by file name)
```

or

```
purge ucr spoolid          (to purge a file by spool ID)
```

and then IPL the system.

If, after reverting to the IBM-defined classes, you wish to return to the classes you defined in the override file, activate the class override file as described on “Verifying and Activating the Override File” on page 441.

**Note:** If your changes are extensive, you may need to install a different directory each time you switch between your own class structure and the IBM-defined class structure.

### Another Way of Changing the Privilege Class of Certain CP Functions

Privilege classes for the following CP functions can be overridden without using the class override file:

- 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 184.

In our insurance company example, the classes of three system functions:

- 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 directory entries

were changed. Instead of including them in the class override file, the system administrator could have entered either the `PRIV_CLASSES` statement as:

```
PRIV_CLASSES OPERATOR IJ HW_SERVICE L USER_DEFAULT P
```

or, the `SYSFCN` macroinstruction as follows:

```
SYSFCN OPER=IJ, SERV=L, DFLT=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 444. For a detailed description of the `SET PRIVCLASS` and `QUERY PRIVCLASS` commands, see the *z/VM: CP Commands and Utilities Reference* book.

---

## Changing the Directory

After preparing and activating the class override file, you must update the directory, as it contains the class or classes of commands that each user can successfully execute. The control statements in the directory that define the command privilege classes for a virtual machine are the `USER` and `CLASS` statements. (For more information about the directory and how to generate it, see Chapter 17, “Creating and Updating a User Directory,” on page 447.) Use the privilege class operand of the `USER` statement to define the new privilege classes for the virtual machine.

## Redefining Command Privilege Classes

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 608 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 136. 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 control 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 control statement.

If you can fit all the new classes onto the USER control 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 either the USER\_DEFAULT operand of the PRIV\_CLASSES statement in the system configuration file or the DFLT operand of the SYSFCN macroinstruction).

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 control 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 control statement:

- a. Change the privilege class operand to an asterisk (\*).

- b. Following the USER control statement, insert a CLASS control statement. The format of the CLASS control statement is described under “CLASS Directory Control Statement (General)” on page 462.

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 control statement and the CLASS control 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 453.) 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 directory entry 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.

**Directory Entry Example:** For our example of the insurance company, the directory might include the USER and CLASS control statements shown in Figure 29 on page 446.

## Redefining Command Privilege Classes

```
DIRECTORY 250 3350 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 NO MATTER WHAT
*     OVERRIDE FILE IS APPLIED. THE PASSWORD SHOULD BE
*     KNOWN BY ONLY A ***VERY*** FEW KEY PEOPLE.
*
USER ZZZMAINT SECRET 2M 8M *
  CLASS ABCDEFGHIJKLMNOPQRSTUVWXYZ123456
  (other control statements)
*
USER SAD1 XXXXXXXX 2M 8M IP
  (other control statements)
*
USER SSP1 XXXXXXXX 1M 2M JP
  (other control statements)
*
USER JSP1 XXXXXXXX 1M 2M KP
  (other control statements)
*
USER JSP2 XXXXXXXX 1M 2M KP
  (other control statements)
*
USER SA1 XXXXXXXX 1M 2M LP
  (other control statements)
*
USER OPERATOR XXXXXXXX 1M 2M MP
  (other control statements)
*
  :
  (and so on)
```

*Figure 29. Directory Entry 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 17. Creating and Updating a User Directory

This chapter 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 control statements
- Use directory profiles
- Use each of the directory control statements summarized in Table 21 on page 449.

---

### Overview

The z/VM user directory describes to CP the configuration and operating characteristics of each virtual machine that can be created by z/VM. A z/VM user directory exists in two forms: a source form that consists of one or more CMS files, and an object form, compiled from the source, on a CP-formatted disk. Multiple object directories can be accessed by z/VM, but only one object directory may be active (online) at any given instant.

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 (page 73) in your system configuration file, CP checks the volumes based on the slot number, in increasing order. If you defined the CP-owned list using the SYSCPVOL macro (page 664) in HCPSYS, CP checks the volumes in the order that they appear on the SYSCPVOL macro.

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 directory. In this case, you can bring a directory online by:

- Entering the DIRECTXA utility to install a directory on an appropriately formatted CP-owned volume
- Attaching a valid directory volume whose volume ID appears in the CP-owned volume list but which was not brought online during initialization.

After CP has brought a directory online, the volume on which it resides remains the directory volume for the duration of the IPL. Rewriting a directory on a different volume, even one higher in the search order, has no effect. (However, a directory written higher in the CP-owned volume search order becomes effective when the next hardware or software IPL occurs.)

The source form of the z/VM user directory consists of directory control statements. These statements define on which CP-formatted volume the object directory will be created, and the configuration and operating characteristics of the virtual machines that will be known to the target VM operating system. A single source directory may define an object directory for one or more VM operating systems.

## Creating and Updating a User Directory

The source directory consists of virtual machine definitions. The source directory can exist in two formats: the monolithic and the cluster. The monolithic format consists of a single CMS sequential file that contains all of the virtual machine definitions. The cluster form consists of an index file that points to one or more definition files that contain all of the virtual machine definitions. A definition file may be, as a group, either a separate part file, a file that contains a single USER definition or a single PROFILE definition, or a cluster file, a file that contains multiple USER or multiple PROFILE definitions. The index file consists of only LOAD directory control statements that point to other definition files to be included during the compile operation.

Regardless of the format used for the source directory, the directory definitions must be processed in the following order:

### **DIRECTORY Definition —**

Which consists of one or more DIRECTORY directory control statements that define the output object directories.

### **Global Definition —**

This section must begin with the GLOBALDEFS directory control statement. It also includes directory control statements which define global settings to be used across all user definitions.

### **PROFILE Definitions —**

Each of which begins with a PROFILE directory control statement and consolidates other directory control statements that are commonly used across multiple users.

### **USER Definitions —**

(Also called *virtual machine definitions*) each of which begins with a USER directory control statement and defines an individual virtual machine.

The Global definition section may appear only after all DIRECTORY statements and before any PROFILE or USER statements. The PROFILE and USER definitions consist of different types of control statements: control, general, and device. Not all directory control statements can be used in PROFILE and USER definitions. To see which ones can be used in which type of definition, refer to the individual statement descriptions described in this chapter. Table 20 lists control statement categories and their placement requirements.

Table 20. Directory Control Statement Categories

Category	Statements	Placement Requirements
General	ACCOUNT, ACIGROUP, APPCPASS, AUTOLOG, CLASS, CPU, CRYPTO, DATEFORMAT, D8ONECMD, INCLUDE, IOPRIORITY, IPL, IUCV, LOGONBY, MACHINE, NAMESAVE, NOPDATA, OPTION, POOL, POSIXGLIST, POSIXINFO, POSIXOPT, SCREEN, SHARE, SPOOLFILE, STDEVOPT, XAUTOLOG, XCONFIG, XSTORE	These statements may appear only before any of the device statements within either a USER definition or a PROFILE definition.
Global	GLOBALDEFS, GLOBALOPTS, POSIXGROUP	These statements may only appear after the DIRECTORY control statement(s) and before all PROFILE and USER statements. The first Global definition statement must be a GLOBALDEFS statement.



Table 20. Directory Control Statement Categories (continued)

Category	Statements	Placement Requirements
Profile	PROFILE	This statement may appear only after the last DIRECTORY statement or the global definition section and before any USER statements.
Device (type 1)	CONSOLE, DASDOPT, DEDICATE, LINK, SPECIAL, SPOOL	These statements may appear only after all general statements within either a USER definition or a PROFILE definition. You may define up to 24,576 virtual devices per virtual machine at logon.
Device (type 2)	MDISK, MINIOPT	These statements may appear only after all general statements within a USER definition.
System affinity	SYSAFFIN	The Prefix form may appear only <i>before</i> a USER statement. The Internal form may appear anywhere <i>within</i> a Global, Profile, or User definition.
Load	LOAD	This statement may appear only in a special form of the directory cluster format file.

Blank lines and lines beginning with an asterisk (\*) may appear. The information on each control statement must appear in columns 1 through 71.

Some directory control statements support mixed-case operands, so care must be taken when editing the directory source file so as 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.

Table 21 describes the directory control statements. It shows the page on which the statement is described, the statement category, and a brief summary of the statement's function. Although the directory control statements are listed in alphabetic order, this is not the order in which they would be coded in the directory.

Table 21. Summary of User Directory Control Statements

Statement	Category	Function	Page
ACCOUNT	General	Defines the account numbers to which a virtual machine may charge its costs. It also defines a distribution code.	456
ACIGROUP	General	Specifies the name of a group to which an individual user ID is assigned.	458
APPCPASS	General	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.	459
AUTOLOG	General	A synonym for XAUTOLOG. See XAUTOLOG on page 578.	461
CLASS	General	Specifies the privilege class (or classes) assigned to an individual user.	462
CONSOLE	Device	Defines a virtual machine operator's console.	463
CPU	General	Specifies a virtual processor that is to be defined automatically when the user logs on to the system.	465
CRYPTO	General	Authorizes a virtual machine to define virtual crypto facilities and specifies the crypto domain or domains the virtual machine is allowed to use. It also specifies whether the virtual machine is authorized to be enabled to enter keys, and whether the special security mode may be used.	468
DASDOPT	Device	An extension to the DEDICATE, LINK, and MDISK statements.	472
DATEFORMAT	General	Specifies a user's default date format for commands that provide multiple date formats.	475

## Creating and Updating a User Directory

Table 21. Summary of User Directory Control Statements (continued)

Statement	Category	Function	Page
DEDICATE	Device	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.	477
DIRECTORY	Control	Defines the device on which the directory resides and must be the first statement in the directory.	480
D8ONECMD	General	Defines whether a virtual machine can issue multiple CP commands with DIAGNOSE code X'08'.	483
GLOBALDEFS	Control	Signifies the beginning of the global definition section.	485
GLOBALOPTS	Control	Used to define global settings to be used while processing user definitions.	486
INCLUDE	General	Specifies the name of a PROFILE entry to be invoked as part of the USER entry.	487
IOPRIORITY	General	Defines a virtual machine's I/O priority queueing range.	488
IPL	General	Designates a device number or named saved system that CP automatically loads (IPLs) when the user logs on to the system.	490
IUCV	General	Authorizes a virtual machine to create an IUCV communication path with another virtual machine.	493
LINK	Device	Accesses another virtual machine's minidisk.	500
LOAD	Control	Specifies where the directory definition entry is to be found. The LOAD statement is valid only in a cluster file format directory.	504
LOADDEV	General	Identifies the location of a program to be loaded as a result of an FCP List-Directed IPL. Parameters to be passed to the program may also be defined.	507
LOGONBY	General	Designates up to eight user IDs that can use their own passwords to logon to and use a virtual machine.	509
MACHINE	General	Defines the virtual machine mode (ESA, XA, or XC) and the number of virtual processors a virtual machine can define.	510
MAXSTORAGE	General	Specifies the virtual storage size for a user.	512
MDISK	Device	Defines virtual disks (minidisks): permanent minidisks, temporary minidisks, and virtual disks in storage.	513
MINIOPT	Device	An extension to MDISK used when defining non-full-pack minidisks.	524
NAMESAVE	General	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.	527
NICDEF	Device	Defines virtual network adapters that are fully simulated by CP.	529
NOPDATA	General	Authorizes a virtual machine to use NOP CCWs to transfer data to CP spool files.	532
OPTION	General	Defines certain options available to the virtual machine.	533
POOL	General	Allows a set of virtual machines to be defined with the same configuration or characteristics.	540
POSIXGLIST	General	Specifies all POSIX groups of which the user is a member.	541
POSIXGROUP	Control	Defines a POSIX group.	543
POSIXINFO	General	Specifies a user's POSIX information.	545
POSIXOPT	General	Specifies option settings related to a user's POSIX capabilities.	548
PROFILE	Control	Defines the start of a PROFILE entry.	551
SCREEN	General	Defines the extended color and extended highlighting options for the virtual machine console.	553

Table 21. Summary of User Directory Control Statements (continued)

Statement	Category	Function	Page
SHARE	General	Specifies a virtual machine's scheduler share.	555
SPECIAL	Device	Defines virtual displays, communication lines, and channel-to-channel adapters that may or may not be connected to real devices when the user logs on.	557
SPOOL	Device	Defines virtual unit record devices (spooling devices).	562
SPOOLFILE	General	Describes virtual machine spool file characteristics.	565
STDEVOPT	General	Specifies the optional storage device management functions available to the virtual machine.	566
STORAGE	General	Specifies the virtual storage size for a user.	568
SYSAFFIN	Control	Defines how, and to which systems of a multiple-system complex, the subsequent statements apply.	569
USER	Control	<ol style="list-style-type: none"> <li>1. Starts a user's directory entry.</li> <li>2. Defines the virtual machine user's logon identification (user ID) and password.</li> <li>3. Defines the virtual machine's storage size.</li> <li>4. Defines the virtual machine user's CP command privilege classes.</li> </ol>	572
XAUTOLOG	General	Designates user IDs that can enter an XAUTOLOG command for the virtual machine.	578
XCONFIG	General	Specifies control parameters for the extended-configuration facilities provided in the ESA/XC virtual machine architecture: access lists and data spaces.	580
XSTORE	General	Authorizes a virtual machine for sole use of the Expanded Storage facility.	585
Blank Line		May 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.	

## Creating the User Directory

To create a user directory, you must:

1. Create a CMS file (or edit an existing CMS file) that contains an entry for each virtual machine that will run on z/VM.
2. Use the DIRECTXA utility to run the user directory creation program.

**Note:** If you are migrating from VM/XA SP™, you cannot use the z/VM DIRECTXA utility to install an object directory in z/VM format on the VM/XA SP system residence pack while your system is running on VM/XA SP. Instead, you should use a second physical pack for the z/VM object directory.

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 control statements in it.

## Creating and Updating a User Directory

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).

---

## Continued Directory Control Statements

Certain directory control 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 non-blank character in columns 1-71 is a comma (.). The statement is continued beginning in column 1 of the next non-comment, non-blank 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 non-comment, non-blank record, with one intervening blank (see "Quoted String Operands" for the syntax rules for very long strings). This process is repeated until a non-continued, non-comment, non-blank record is encountered. The statement is then parsed according to its normal rules. Statement names and keywords may not be split across multiple records.

Some statements that support continuation may 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 may 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 non-blank character count towards the limit.

The following are some examples of directory control 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 control 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 across multiple records of the directory file, a quoted string operand, with or without imbedded blanks, may continue across 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 may 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 quotes.
3. To include a single quote inside a quoted string surrounded by single quotes two consecutive single quotes are required inside the string. Likewise, to include a double quote inside a quoted string surrounded by double quotes, two consecutive double quotes are required inside the string.
4. To include a single quote inside a quoted string surrounded by double quotes, the single quote need not be duplicated. To include a double quote inside a quoted string surrounded by single quotes, the double quote need not be duplicated.
5. The final result string of a quoted string is formed as follows: any duplicated single or double quotes are reduced to one single or double quote, respectively, then the surrounding quotes 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 may not 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, may continue across multiple records by splitting it into quoted strings and splitting the statement between quoted strings.
8. A quoted string may not be imbedded in or concatenated with another quoted string or a non-quoted string. For example, 'AB"C' and ABC'DEF' are not valid quoted strings.

The following are some examples of directory control 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"""
```

---

## 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*

*fn* is the file name of your directory. The default is USER.

*ft* is the file type of your directory. The default is DIRECT.

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:

## Creating and Updating a User Directory

```
directxa special
```

To bring online the new version of a directory named SECRET DRCT, enter:

```
directxa secret drct
```

### Notes:

1. 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).
2. The system does not check for overlapping extents in the directory. Therefore, while changing the directory source file, you should check your new directory entries 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.
3. If you are migrating from VM/XA SP, you cannot use the DIRECTXA utility to install an object directory in z/VM format on the VM/XA SP system residence pack while your system is running on VM/XA SP. Instead, you should use a second physical pack for the z/VM object directory.

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 directory entries 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. It cannot, however, delete existing entries nor alter USER entries in the source directory. It only replaces 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
```

---

### Specifying a Directory That Contains VM/SP Control Statements

If you want to use a directory from a VM/SP system or VM/SP HPO system and some of the statements or options are not supported by z/VM, specify the MIXED option of the DIRECTXA utility. For example, assume you have a VM/SP directory in a file named SPDIRECT DIRECT. To bring this directory online, enter:

```
directxa spdirect (mixed
```

In response, DIRECTXA ignores most VM/SP or VM/SP HPO source statements that it doesn't support, rather than flagging them as errors.

**Note:** DIRECTXA sends you a warning when it encounters a VM/SP or VM/SP HPO statement but will continue to create the updated directory.

---

### Creating Directory Profiles

If certain directory control statements are repeated for several users, you can make use of directory profiles to save space in the directory. Using directory profiles is similar to creating a user entry in the directory. To create directory profile entries:

1. Determine which control statements are commonly repeated for a designated set of user entries.
2. Create a directory profile that contains those control statements on the source directory using the PROFILE statement (page 551).
3. Insert an INCLUDE control statement (page 487) immediately after the USER control statement for those entries that will reference the profile.
4. Remove the existing directory control statements in the user entries that reference a profile except for those control statements that are unique to those user entries.
5. Run DIRECTXA in EDIT mode to ensure that there are no errors. Any error conditions will result in appropriate messages to your console.
6. Correct any errors and run DIRECTXA again to install the newly created directory.

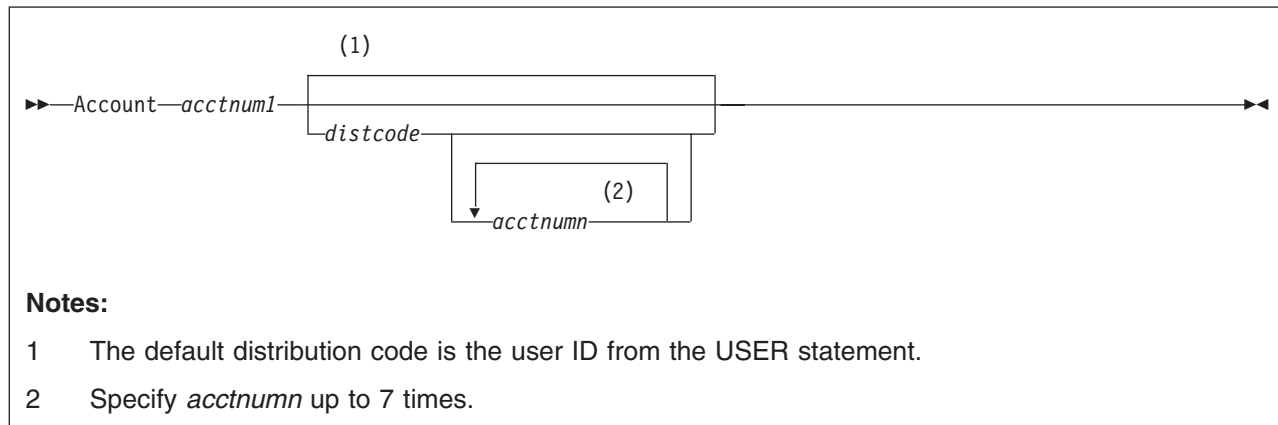
---

### Determining How Much Space the Directory Needs

The directory normally requires about two cylinders of CKD/ECKD space or 50 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 608.

## ACCOUNT

### ACCOUNT Directory Control Statement (General)



### 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

If specified, the ACCOUNT statement must go before the first device statement you code in a user's directory entry. (For a list of device statements, see Table 20 on page 448.) Multiple ACCOUNT statements are allowed within a profile, but are only used if no ACCOUNT statements are in the user directory 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 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 user's 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 user's directory entry, 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 about the SET ACCOUNT command, see the *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 directory entry:

```
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 directory entry:

```
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 directory entry:

```
Account job101 bin-99z job102 job103 play999
```

---

## ACIGROUP Directory Control Statement (General)

```
▶—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 directory entries 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

If you use the ACIGROUP statement, it must precede any device statements you specify in a user's directory entry. (The device statements are: CONSOLE, DASDOPT, DEDICATE, LINK, SPECIAL, and SPOOL.) One ACIGROUP statement is allowed within a profile but is only used if no ACIGROUP statement is in the user directory 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 USER directory entry.

### Examples

To specify that user ID JONES belongs to group MUNDANE, use the following directory control statements:

```
User jones ww11qq 8m 16m *
  ⋮
ACIgroup mundane
```

## APPCPASS Directory Control Statement (General)

```
▶—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 directory entry 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

If you use the APPCPASS statement, it must precede any device statements you code in the user's directory entry. (For a list of device statements, see Table 20 on page 448.) Multiple APPCPASS statements are allowed within a profile entry. Any APPCPASS statements found within a profile entry are added to those found in the including user entry with no duplicate checking performed. A statement that appears within a user definition is processed before a statement that appears within an included profile definition.

### 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

## APPCPASS

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<sup>®</sup> (SAA<sup>®</sup>) Common Programming Interface for Communications (CPIC).

## Examples

The following APPCPASS statements in the directory 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 Control Statement (General)

### Purpose

The AUTOLOG statement is a synonym for XAUTOLOG that is available for compatibility reasons. Refer to “XAUTOLOG Directory Control Statement (General)” on page 578 for the format and for information on coding the XAUTOLOG statement.

### How to Specify

Multiple AUTOLOG statements are allowed within a profile but are only used if no AUTOLOG or XAUTOLOG statements are in the user directory entry.

---

## CLASS Directory Control Statement (General)

►—Class—*classes*—◄

### Purpose

The CLASS statement specifies the privilege class or classes assigned to an individual user.

### How to Specify

The CLASS statement, if used, must be after the USER statement and must precede any device statements. (For a list of device statements, see Table 20 on page 448.) One CLASS statement is allowed within a profile if the USER statement CLASS field of each of the including virtual machines is blank or contains only an asterisk (\*). A CLASS statement within a user directory entry overrides a CLASS statement in a profile.

### 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. See Chapter 16, “Redefining Command Privilege Classes,” on page 433 for information on how to restructure your command privilege classes if you want them to be different from the IBM-defined default classes A, B, C, D, E, F, and G.

### Usage Notes

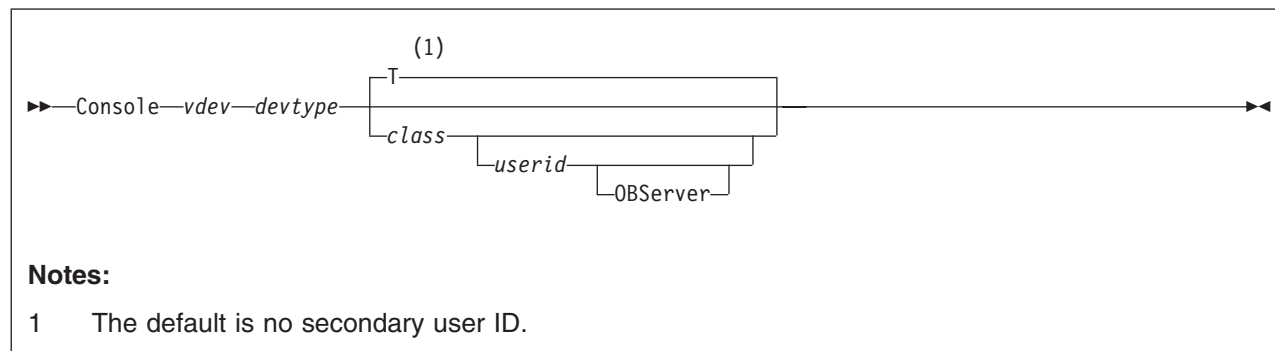
1. If you do not specify a CLASS statement and you specify the class field of the USER statement as either a blank or an asterisk (\*), CP uses the default class or classes specified on the PRIV\_CLASSES statement in your system configuration file.
2. If the CLASS field of the USER statement is not blank 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, because some CMS commands invoke DIAGNOSE codes (for example, DUMPVIEW, GENIMAGE, and OVERRIDE).

### 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 control statements:

```
User Bones ww11qq 8m 16m *
Class z16axcb25g
```

## CONSOLE Directory Control Statement (Device)



### Purpose

The CONSOLE statement defines the virtual machine console for any virtual machine whose user directory entry contains no CONSOLE statement but does contain an INCLUDE statement referring to that profile.

### How to Specify

One CONSOLE statement is allowed within a profile entry. If you define a CONSOLE statement in both a user directory entry and its included profile entry, at logon time two requests to define a console are presented to the system; the user directory entry is presented first.

If you code a CONSOLE statement, it must follow any general statements you code in a user's directory entry. (For a list of general statements, see Table 20 on page 448.)

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.

## CONSOLE

### 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	<ul style="list-style-type: none"><li>• Changes the console address</li><li>• Creates a console if one does not already exist.</li></ul>
DETACH	Deletes the console
SPOOL	Changes the spooling class
TERMINAL CONMODE	Changes the device type

The user ID field cannot be changed or deleted by the user.

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.

## 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's directory entry:

```
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's directory entry:

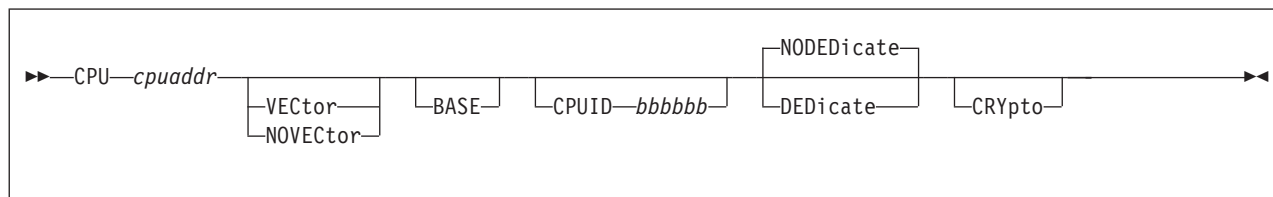
```
Console 0a9 3270 y
```

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's directory entry:

```
Console 01f 3215 t aljones
```



## CPU Directory Control Statement (General)



### 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 maximum number of CPU statements allowed in the user's directory entry is controlled by the *mcpu* operand of the user's MACHINE statement, if specified. If you specify the CPU statement in a user's directory entry, it must appear after the USER statement and before any device statements. (For a list of device statements, see Table 20 on page 448.)

Multiple CPU statements are allowed within a profile entry. If you specify CPU statements in a user directory entry and a profile entry with the same processor address (*cpuaddr*), the user directory entry overrides the profile entry. Otherwise, the user and profile CPU entries are additive.

If you do not specify a CPU statement in a user's directory entry, 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'.

#### **VEctor**

#### **NOVEctor**

specifies whether a virtual vector facility should be defined automatically for the virtual CPU at LOGON time. For base virtual processors with at least one real vector facility installed, the default is VECTOR; otherwise, the default is NOVECTOR. For nonbase processors, the default is always NOVECTOR. If you define the virtual processor as VECTOR and no real vector facility exists on the real processor, CP issues a message that indicates that no real vector facility exists to service the virtual vector facility.

#### **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. 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

## CPU

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 the CPUID operand on the OPTION statement, see page 533.

### **DEDicate**

#### **NODEDicate**

specifies whether this virtual processor is to be dedicated at LOGON time to a real processor.

If NODEDICATE is specified, this virtual processor is not to be dedicated to a real processor. It has the same effect as entering the UNDEDICATE command for this virtual CPU. NODEDICATE is the default.

### **CRYPto**

specifies whether a virtual Integrated Cryptographic Facility (ICRF) should be defined automatically for the virtual CPU at LOGON time. If you define the virtual CPU as CRYPTO and no real ICRF is installed on the real processor at LOGON time, CP issues a message that indicates that no real crypto facility is installed.

## Usage Notes

1. If the virtual processor is to be dedicated, CP tries to dedicate a real processor that has the same facilities as the virtual processor. For example, CP tries to match a real processor with a vector facility to a virtual CPU with a defined virtual vector facility, and a real processor without a vector facility to a virtual CPU without a defined virtual vector facility.
2. The order of virtual machine CPU dedication is the base CPU first, with any other nonbase processors following in the same order as the CPU entries in the user's directory entry.
3. If DEDICATE is specified on the CPU statement and dedication occurs as a result of this, and if the processor becomes no longer dedicated later for any reason, rededication does not occur.
4. The DEDICATE and UNDEDICATE commands override this statement.
5. The directory entry contains conflicting statements if multiple processors are defined on CPU statements and OPTION CONCEAL is specified. An information message is issued, the CONCEAL option is rejected, and directory processing continues.
6. 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.
7. Use of the ICRF must be authorized by the CRYPTO statement if the CRYPTO operand is to be specified. The default is to not define an ICRF.
8. A maximum of two virtual crypto facilities may be defined for a virtual machine. If a virtual machine user has more than two CPU statements with CRYPTO specified, CP defines virtual crypto facilities for only the first two CPU statements specifying crypto. CP issues an error message to indicate that the maximum number of cryptos have already been defined.

## Examples

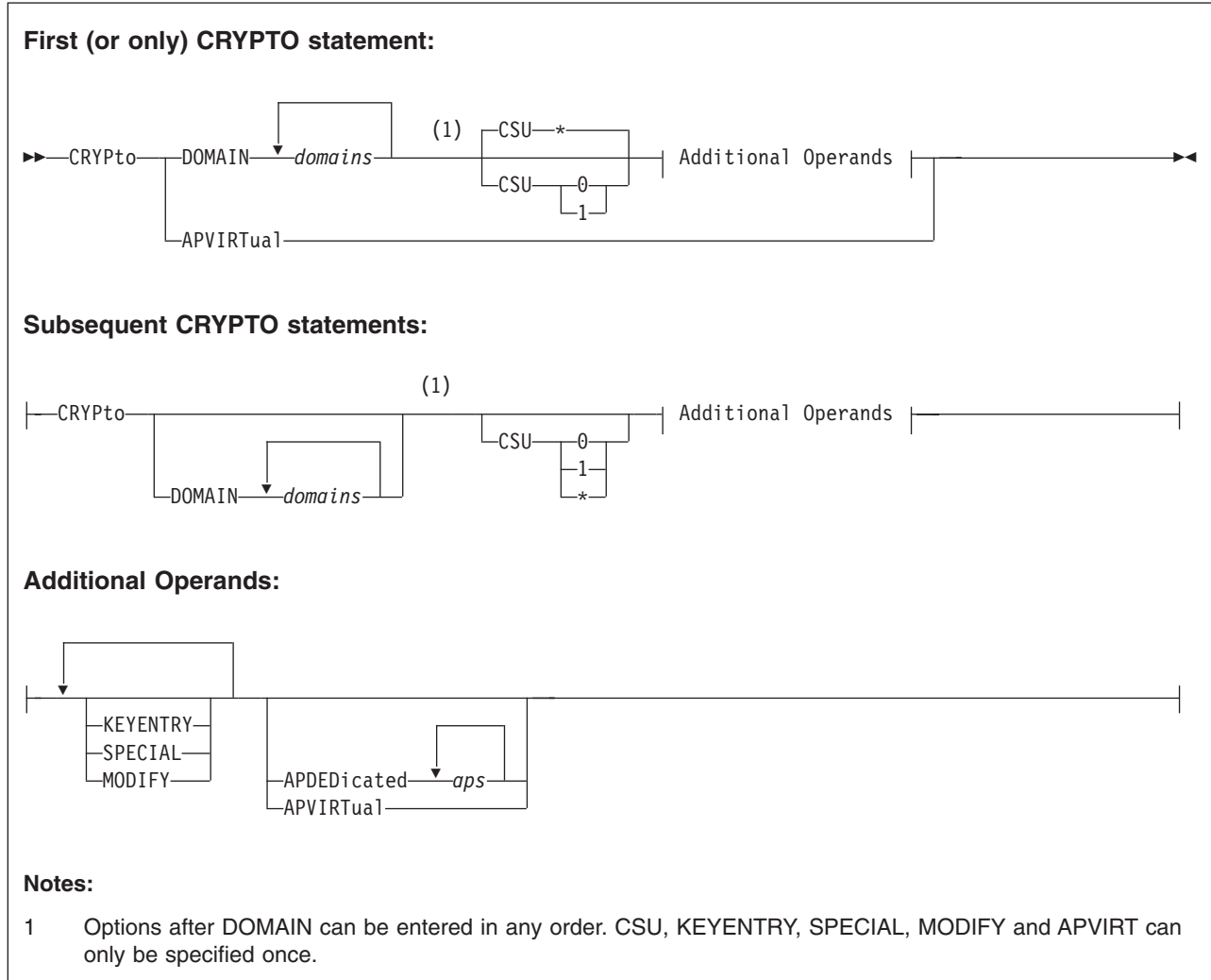
To define a vector-enabled base CPU at address X'3F', with a CPU ID of X'00003F', use the following CPU statement:

```
CPU 3f vector base CPUid 00003f
```

To define a crypto-enabled base CPU at address X'DE', with a CPU ID of X'0000DE', use the following CPU statement:

```
CPU de novector base CPUid 0000de Crypto
```

## CRYPTO Directory Control Statement (General)



### Purpose

The CRYPTO statement authorizes the user to define virtual cryptographic facilities and specifies which domains and CAM queues the virtual machine is allowed to use. It also specifies whether the virtual machine is authorized to be enabled to enter keys or issue PKSC Modify instructions, and whether the special security mode can be used.

### How to Specify

The CRYPTO statement, if included in the directory entry, must appear after the USER statement and before any device statements. (For a list of device statements, see Table 20 on page 448.)

The CRYPTO statement cannot be used in a profile unless it is used with APVIRT only.

You can specify as many CRYPTO statements as you need to assign domains or APs to the virtual machine. Once the DOMAIN operand is specified, additional CRYPTO statements may be specified with the other operands. Also, the DOMAIN

or APDED operands may be specified again on additional CRYPTO statements followed by the additional domains or APs to be assigned. Note, however, that the CSU, APVIRT, and other operands may only be specified once.

## Operands

### **DOMAIN** *domains*

specifies up to 16 domains the virtual machine may use. Domains are 0 through 15, specified in decimal. They may appear in any order, without duplication. If CAM queues are available, the virtual machine may also use the CAM queues corresponding to the domains specified. If AP queues are available, and the APDED operand is specified, the virtual machine may use the AP queues corresponding to the domains specified. The DOMAIN operand may be specified more than once on additional CRYPTO statements, but the DOMAIN operand must be specified before any additional operands other than APVIRT are specified.

### **CSU \***

tells CP to assign this virtual machine's virtual crypto facilities to real crypto units. This operand also tells CP to allow this virtual machine to use both Crypto Asynchronous Processors (CAPs) 1 and 0. If you do not specify CSU, the default is CSU \*.

### **CSU 0**

tells CP to assign this virtual machine's virtual crypto facility to crypto unit 0. This operand also tells CP to allow this virtual machine to use CAP 0.

### **CSU 1**

tells CP to assign this virtual machine's virtual crypto facility to crypto unit 1. This operand also tells CP to allow this virtual machine to use CAP 1.

### **KEYENTRY**

tells CP that this virtual machine is authorized to issue certain PCCF functions that operate on cryptographic keys.

### **SPECIAL**

tells CP that this virtual machine can use the special security mode.

### **MODIFY**

tells CP that this virtual machine can issue the PKSC Modify instructions.

### **APVIRTual**

tells CP that this virtual machine may share access with other virtual machines to the clear-key functions of the Adjunct Processor (AP) cryptographic facility. The APVIRT operand may be specified once in any order. The APVIRT operand may not be specified if the APDED operand is specified. It is not necessary to specify the DOMAIN or CSU operands when specifying the APVIRT operand.

### **APDEDicated** *aps*

specifies up to 64 APs the virtual machine may use for dedicated access to the Adjunct Processor (AP) cryptographic facility. APs are 0 through 63, specified in decimal. They may appear in any order, without duplication. The DOMAIN operand also must be specified to indicate the AP queues to access. The AP queues equivalent to the domains specified by all of the DOMAIN operands will be assigned to the guest for each AP specified on the APDED operand on all CRYPTO statements. The APDED operand may not be specified if the APVIRT operand is specified. The APDED operand may be specified more than once on additional CRYPTO statements.

## Usage Notes

1. Only one virtual machine may use a domain at a time. If more than one virtual machine has a CRYPTO statement for a given domain, only the first virtual machine that logs on receives use of the domain.
2. The domains specified must be installed on the real processor. If a specified domain is not installed on the real processor at LOGON time, a message is issued that the domain is not available.
3. If there is more than one crypto unit in the real cryptographic facility, the CSU option should be specified. CSU 0 or CSU 1 should be specified for virtual machines that will have a virtual crypto facility on only one virtual CPU. CSU \* should be specified for configurations that will have a virtual crypto facility defined on more than one virtual CPU.
4. For CMOS cryptographic facilities, the use of CSU \* is recommended. This allows the virtual machine to use both Crypto Asynchronous Processor (CAP) 0 and CAP 1.
5. The KEYENTRY parameter was necessary on older processors along with the SET CRYPTO KEYENTRY command to enable key-entry for one user at a time. Today, the KEYENTRY parameter is still necessary to allow use of the PCCF key functions. However, the SET CRYPTO KEYENTRY command is not needed to enable individual key-entry because more than one user may issue these functions at a time on current processors.
6. It is recommended that multiple CRYPTO statements for a user be specified contiguously in the user directory entry.
7. The APs specified must be installed on the real processor. If a specified AP is not installed on the real processor at LOGON time, a message is issued that the AP is not available.
8. On real processors which have only the Adjunct Processor (AP) cryptographic facility installed, it is not necessary to specify the CSU, KEYENTRY, SPECIAL, or MODIFY operands. If any of these operands are specified, they will be ignored without generating any errors.
9. If major changes are made to the configuration of the PCI crypto queues by updating the APDED and APVIRT operands on the CRYPTO statements for users without a subsequent system IPL, it is possible that no queues of the type selected by z/VM will remain for queue-sharing, even though they may appear to be available when the QUERY CRYPTO APQS command is issued. If it is necessary to change the type of queues available for queue-sharing, a system IPL must be performed after the directory changes are made.

## Examples

1. To specify that a virtual machine can use domains 1 and 2, use the following CRYPTO statement in the virtual machine's directory entry:

```
Crypto Domain 1 2 CSU * Special Modify Keyentry
```

CP will assign the virtual crypto facilities to real crypto units. If the Crypto Asynchronous Processor (CAP) is provided in the processor configuration, the virtual machine also will be able to use CAP 0 and CAP 1 and CAM queues 1 and 2.

2. To specify that the virtual machine may use the clear-key functions of the Adjunct Processor (AP) cryptographic facility, use the following CRYPTO statement in the virtual machine's directory entry:

```
Crypto Apvirt
```

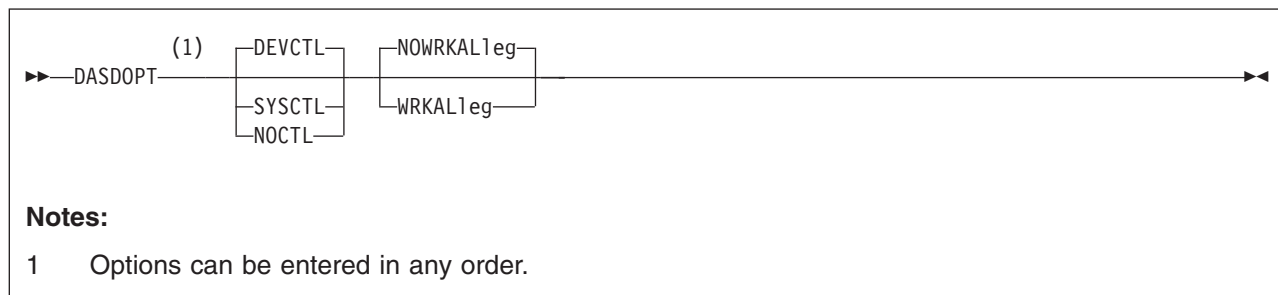
3. To specify that a virtual machine can use domains 3 and 7 and AP queues 3 and 7 on APs 0, 2, and 3, use the following CRYPTO statement in the virtual machine's directory entry:

```
Crypto Domain 3 7 CSU * Special Modify Keyentry Apded 0 2 3
```

4. To specify that a virtual machine can use AP queues 2, 10, 11, 12, and 13 on AP 4 use the following CRYPTO statement in the virtual machine's directory entry:

```
Crypto Domain 2 10 11 12 13 Apded 4
```

## DASDOPT Directory Control Statement (Device)



### Purpose

The DASDOPT statement is an extension to the MDISK, LINK, and DEDICATE statements, and immediately follows them.

### How to Specify

Multiple DASDOPT statements are allowed within a profile entry, but each must directly follow a DEDICATE statement. A DASDOPT statement can follow a LINK or MDISK statement only if that statement defines a full-pack minidisk. DIRECTXA cannot detect every case when the MDISK preceding the DASDOPT is not a full-pack minidisk nor can it 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. Unexpected results may occur.

When processing the DASDOPT statement, DIRECTXA checks for two tokens: one of the control tokens (DEVCTL, NOCTL, or SYSCTL) and WRKALLEG. 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

#### 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 473 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 473 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

#### 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.



## 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.

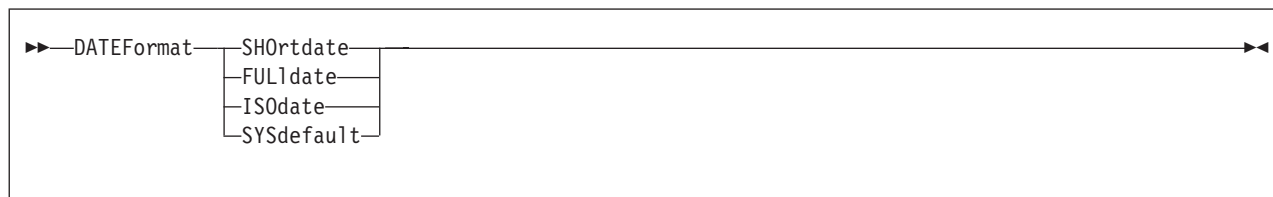
Table 22 on page 474 shows you the additional control CCWs that CP accepts for a level of control. The DEVCTL column shows the CCWs that CP accepts in addition to those it accepts for the NOCTL level of control. The SYSCTL column shows the CCWs that CP accepts in addition to those it accepts for the DEVCTL and NOCTL levels of control.

## DASDOPT

Table 22. DASD Control Levels

Control Unit Type	Additional Control CCWs Accepted for DEVCTL	Additional Control CCWs Accepted for SYSCTL
3990-3,6	<ul style="list-style-type: none"><li>• Set subsystem mode<ul style="list-style-type: none"><li>– Activate caching for device</li><li>– Deactivate caching for device</li><li>– Activate DASD fast write</li><li>– Deactivate DASD fast write</li><li>– Force deactivate DASD fast write</li></ul></li><li>• Perform Subsystem Function<ul style="list-style-type: none"><li>– Establish Duplex Pair</li><li>– Terminate Duplex Pair</li><li>– Suspend Duplex Pair</li><li>– Direct I/O to One Device of the Duplex Pair</li><li>– Set Interface ID</li></ul></li></ul>	<ul style="list-style-type: none"><li>• Set subsystem mode<ul style="list-style-type: none"><li>– Make cache available</li><li>– Make cache unavailable</li><li>– Force cache unavailable</li><li>– Make NVS available</li><li>– Make NVS unavailable</li><li>– Activate cache fast write</li><li>– Deactivate cache fast write</li></ul></li><li>• Perform Subsystem Function<ul style="list-style-type: none"><li>– Destage Modified Tracks</li><li>– Set Cache Allocation Parameters</li><li>– Suspend/Resume Function</li></ul></li></ul>

## DATEFORMAT Directory Control Statement (General)



### 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 user or profile entry. A DATEFORMAT statement in a user entry overrides a DATEFORMAT specification in a profile entry.

If you specify the DATEFORMAT statement, it must precede any device statements that you specify in a profile or user entry. (For a list of device statements, see Table 20 on page 448.)

### 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. Refer to "SYSTEM\_DATEFORMAT Statement" on page 225 for a description of the SYSTEM\_DATEFORMAT configuration statement, and the *z/VM: CP Commands and Utilities Reference* for a description of the SET DATEFORMAT command.

### 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 user's directory entry, 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.

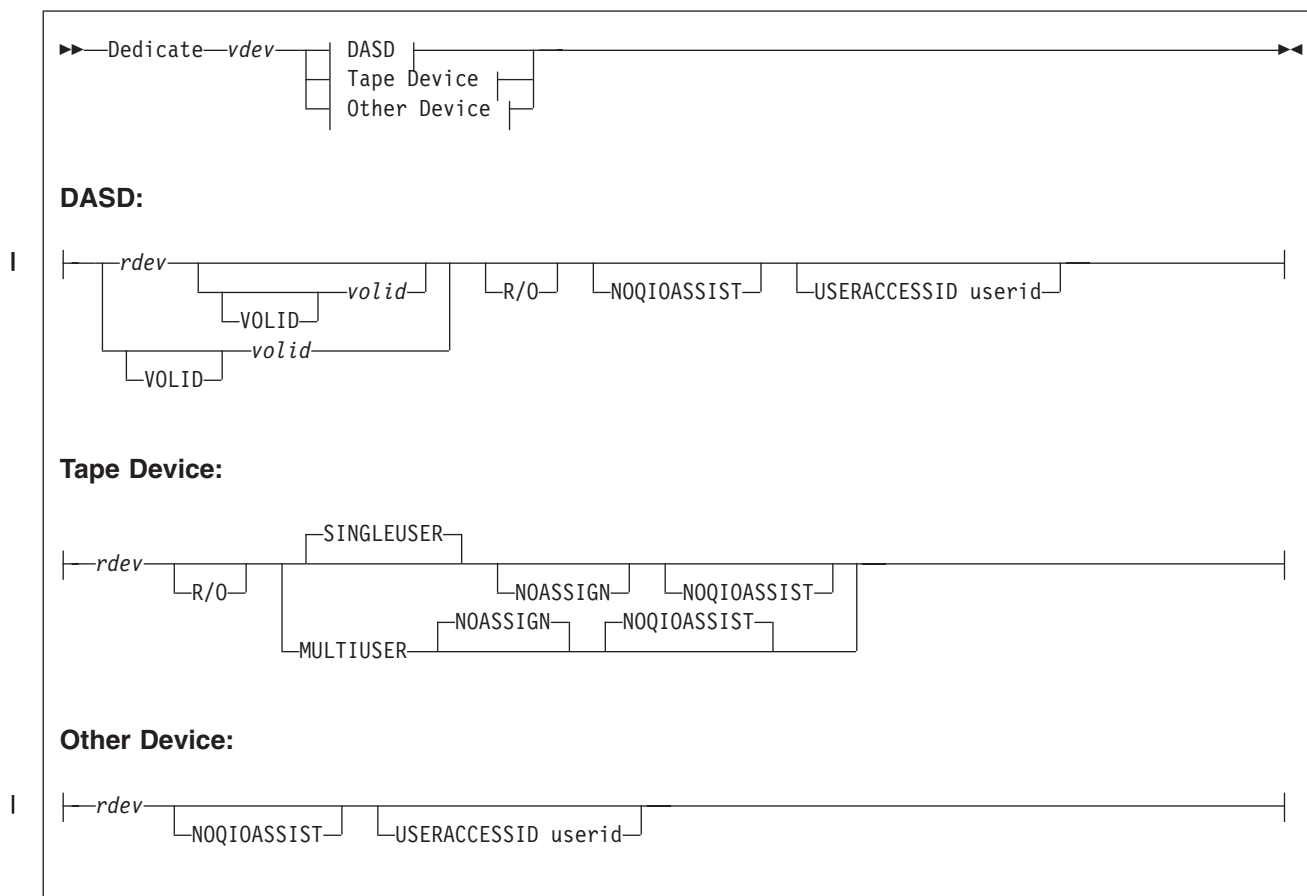
## DATEFORMAT

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 user's directory entry:  
DATEFORMAT SHORTDATE
2. To specify the user's default date format as FULLDATE, use the following DATEFORMAT statement in the user's directory entry:  
DATEFORMAT FULLDATE
3. To specify the user's default date format as ISODATE, use the following DATEFORMAT statement in the user's directory entry:  
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 user's directory entry:  
DATEFORMAT SYSDEFAULT

## DEDICATE Directory Control Statement (Device)



### 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 about the DASDOPT statement, see “DASDOPT Directory Control Statement (Device)” on page 472.

### How to Specify

If you specify DEDICATE statements, they must follow any general statements in a user’s directory entry. (For a list of general statements, see Table 20 on page 448.)

IBM recommends that you 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.

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 entry is resolved at logon time; the DEDICATE request from the user entry is presented to the system first.

## DEDICATE

### Operands

*vdev*

is the virtual device number.

*rdev*

*rdev* is the real device number.

**VOLID** *valid*

is the volume serial number of a disk pack mounted on a real disk storage device. The variable *valid* must be a 1- to 6-character string. If *valid* 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.

**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 about the ATTACH command, see the *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 directory entry 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 3424, 3480, 3490, and 3590 tape devices.

## 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's directory entry:

```
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's directory entry:

```
Dedicate 220 200 valid 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's directory entry:

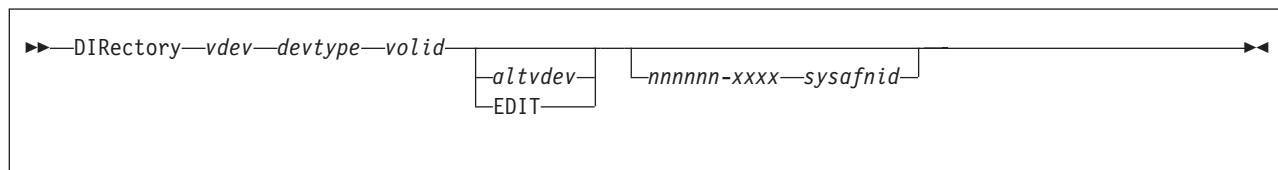
```
Dedicate 330 1300 valid xb9999 r/o
```

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's directory entry:

```
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 Control Statement (Control)



### Purpose

The DIRECTORY statement defines to CP the device on which you have allocated space for the directory. (This device must also be a CP-owned volume.) If your system is part of a multisystem complex, a DIRECTORY statement is required for each logical system.

### How to Specify

The DIRECTORY statements must be the first statements in your source directory or control DIRMPART file (cluster format directory). DIRECTORY statements are not allowed within a profile entry.

If you specify the EDIT option 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 option in the source directory.

If any of these conditions is not met, an error message is issued.

If you are sharing a single source directory among several systems, all DIRECTORY statements must be listed at the beginning of the source directory and before any other control statements.

When processing the DIRECTORY statement, DIRECTXA checks for a maximum of six tokens: virtual device number, device type, volume serial number, alternate device number or the EDIT operand, the processor ID, and a system ID, in that order. If you specify more than six 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.

*devtype*

is the device type. Valid device types are:

3380                      3390                      9336                      FB-512

FB-512 is a generic value which may 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.

*valid*

is the volume serial number of the directory volume. The variable *valid* 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.

**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 option. You can only specify the EDIT option on one DIRECTORY statement, and it must be the last of the set of DIRECTORY statements. DIRECTXA validates the syntax of this statement but ignores its contents.

*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 processor identification number, and *xxxx* is the model number of the real machine. For more information about the QUERY CPUID command, see the *z/VM: CP Commands and Utilities Reference*.

If your system is an *n*-way processor operating in single-image mode, you can specify the processor ID as *\*nnnnn-xxxx*. This allows you to use one DIRECTORY statement to define all CPUIDs. See “Usage Notes” for more information about defining processor IDs with one DIRECTORY statement.

*sysafnid*

is a 1- to 8-character alphanumeric string that identifies the system whose object directory is affected by the SYSAFFIN statements (page 569).

**Usage Notes**

1. CP dynamically updates the active z/VM directory 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 or using the SYSCPVOL macroinstruction in HCPSYS). For more information about the CP\_OWNED statement, see page 73. For more information about the SYSCPVOL macroinstruction, see page 664.
2. An asterisk (\*) in the first position of the processor ID, *\*nnnnn-xxxx*, lets you use one DIRECTORY statement to define all of the processor IDs of an *n*-way processor operating in single-image mode. Nevertheless, if a four-way processor is partitioned into two logical two-way processors, you must create a DIRECTORY statement for each processor that specifies all 11 digits of the processor ID (*nnnnnn-xxxx*) for each logical processor. This is because the processor ID of the two logical systems would differ only in the first character of the processor ID.
3. 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 16 unique system affinity IDs can be specified on multiple DIRECTORY statements.

**Examples**

The following examples show the various ways of coding the DIRECTORY control statement:

## DIRECTORY

1. 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 0123code the following DIRECTORY statement:

```
Directory 0123 fb-512 xa0001 0223
```

2. To specify:
  - A multiple system definition where a 2064 two-way processor is partitioned into two uniprocessors, SYSTEMA (serial 012345) and SYSTEMB (serial 112345)
  - A 3380 EDIT work volume with a virtual device number of 0243 and a volume serial number of DRM19F

code the following DIRECTORY statements:

```
Directory 0123 3380 xa0001 0223 012345-2064 systema
Directory 0223 3380 xa0002 0233 112345-2064 systemb
Directory 0243 3380 drm19f edit
```

3. To specify a 2064 four-way processor configured as a single-image processor so that the system can be IPLed on any of the four processors, use the following DIRECTORY statements:

```
Directory 0b64 3380 vmaipl 015211-2064 vma
Directory 0b64 3380 vmaipl 115211-2064 vma
Directory 0b64 3380 vmaipl 215211-2064 vma
Directory 0b64 3380 vmaipl 315211-2064 vma
Directory 0243 3380 drm19f edit
```

Or you can use:

```
Directory 0b64 3380 vmaipl *15211-2064 vma
Directory 0243 3380 drm19f edit
```

4. To specify a 2064 partitioned into two 2-way images, you must use two pairs of DIRECTORY statements, one pair for each image. Each statement in a pair differs only in the first character of the processor ID:

```
Directory 0b64 3390 vmaipl 020893-2064 vma
Directory 0b64 3390 vmaipl 220893-2064 vma
Directory 0784 3380 vmbipl 120893-2064 vmb
Directory 0784 3380 vmbipl 320893-2064 vmb
Directory 0243 3390 drm19f edit
```

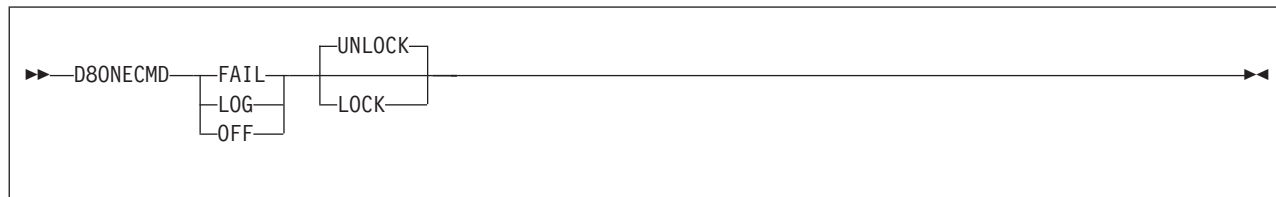
5. To specify an *n*-way processor, such as a 2064, that is to be treated as a single image within a complex, use a DIRECTORY statement with an asterisk (\*) in the first character of the processor ID. This ensures that the DIRECTXA command can be entered on any processor in the complex.

```
Directory 0b64 3390 vmaipl *20893-2064 vma
```

6. To specify a multiple system environment where four 2064 processors share a single directory, use the following DIRECTORY statements:

```
Directory 0123 9336 fba001 0223 012345-2064 fbasysa
Directory 0223 9336 fba002 0223 012355-2064 fbasysb
Directory 0323 9336 fba003 0223 012445-2064 fbasysc
Directory 0423 9336 fba004 0223 013345-2064 fbasysd
```

## D8ONECMD Directory Control Statement (General)



### 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

If you specify a D8ONECMD statement, it must appear after the USER statement or in a profile for a user. You can only have one D8ONECMD statement for each user entry. If you specify more than one, you will receive an error message. If you specify a D8ONECMD statement in a user entry and you also specify a D8ONECMD statement within the profile entry for that user, CP uses the D8ONECMD statement in the user entry and ignores the D8ONECMD statement in the profile entry.

If you do not specify a D8ONECMD statement, CP will not prevent users from issuing multiple commands with DIAGNOSE code X'08' and will allow 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 user entry. This prevents users from changing the D8ONECMD setting for their virtual machine.

#### UNLOCK

tells CP not to lock the D8ONECMD setting for a specific user entry. This allows user to change the D8ONECMD setting for their virtual machine.

## D8ONECMD

### Usage Notes

1. If you omit the D8ONECMD for any user entry, 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 a back 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 a back 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 settinguse the following D8ONECMD statement:

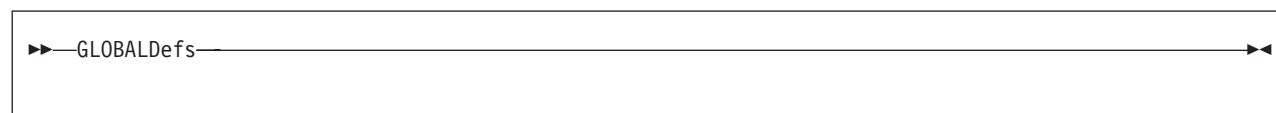
```
D8OneCmd Fail Lock
```

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 settinguse the following D8ONECMD statement:

```
D8OneCmd Log UnLock
```

---

## GLOBALDEFS Directory Control Statement (Control)



### Purpose

The GLOBALDEFS statement denotes the start of the global definition section. If no global definitions are specified, this statement is optional.

### How to Specify

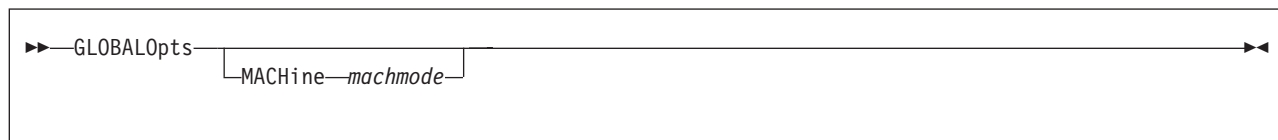
The GLOBALDEFS statement, if specified, must directly follow the DIRECTORY control statement(s) and precede all PROFILE and USER definitions. There can be only one GLOBALDEFS directory control statement specified.

### Examples

To see an example of the use of this statement, see “GLOBALOPTS Directory Control Statement (Control)” on page 486.

---

## GLOBALOPTS Directory Control Statement (Control)



### 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 section of the source directory (after the GLOBALDEFS statement and before any USER or PROFILE definitions).

### Operands

#### **MACHINE** *machmode*

specifies the virtual machine mode for any user definition that does not contain a MACHINE directory control statement. The valid values that may be used for the virtual machine mode are ESA, XA, and XC.

### Examples

In the examples that follow, assume this multisystem complex definition:

```
Directory 07a4 3380 vmaip1 *01234-2064 vma
Directory 0b64 3380 vmbip1 104567-2064 vmb
Directory 0cF4 3380 vmcip1 *15211-2064 vmc
Directory 0664 3380 vmdip1 *00012-2064 vmd
```

The following global definition section 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
```

---

## INCLUDE Directory Control Statement (General)

```
▶—INclude—profilename—◀
```

### Purpose

The INCLUDE statement specifies the name of a profile entry to be invoked as part of the USER statement (page 572).

### How to Specify

INCLUDE statements are not allowed within a profile entry. If you specify an INCLUDE statement, it must directly follow the USER statement. Normally, you can only specify one INCLUDE statement in a user directory entry. However, in a CSE system you could have an INCLUDE statement 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 user directory entry definition and analyzed with respect to the rest of the user's definition data.
2. In a system using cross system extensions, 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

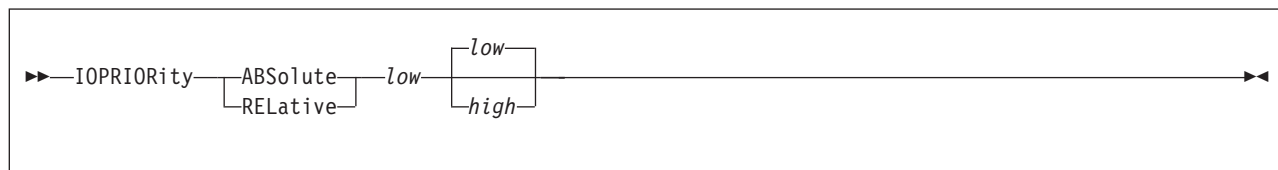
1. To specify PROFILE1 as a profile entry to be invoked as part of the USER statement, use the following statements:

```
User example ...
Include profile1
⋮
```

2. To tailor a user's profile selection by system in a CSE complex, use the following statements:

```
User example ...
Sysaffin vma
Include d123vma
Sysaffin vmb
Include d123vmb
⋮
```

## IOPRIORITY Directory Control Statement (General)



### 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 or profile entry. An IOPRIORITY statement in a user 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 or user entry. (For a list of device statements, see Table 20 on page 448.)

### Operands

#### ABSolute

indicates the type of I/O priority range. If LPAR I/O Priority Facility is enabled, the I/O priority range must fit onto the range 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.



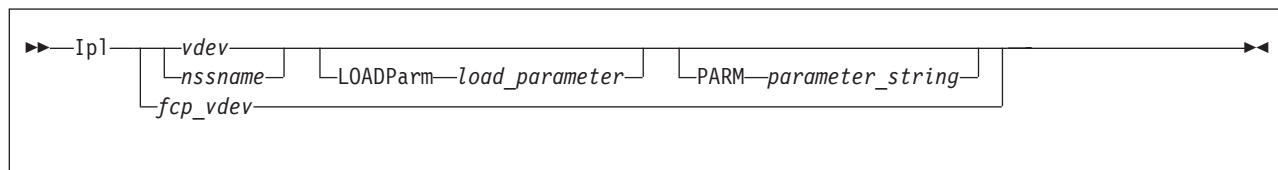
2. For more information on IOPRIORITY ranges, refer to the CP SET IOPRIORITY command in the *z/VM: CP Commands and Utilities Reference*.

## Examples

1. To assign a RELATIVE range of 8 values, use the following IOPRIORITY statement in a virtual machine's directory entry:  

```
ioprior rel 0 7
```

## IPL Directory Control Statement (General)



### Purpose

The IPL statement designates a named saved system or the device number of a device that CP automatically loads (IPLs) when the user logs on the virtual machine.

### How to Specify

If you specify the IPL statement, it must precede any device statements in a user's directory entry. (For a list of device statements, see Table 20 on page 448.)

Multiple IPL statements are allowed within a profile but are used only if no IPL statements are in the user directory entry. If multiple IPL statements are in an included profile entry, only the last one is used.

### Operands

#### *vdev*

is the virtual device number of the device CP is to automatically IPL when the user logs on. The variable *vdev* is any hexadecimal number from X'0000' to X'FFFF'.

#### *nssname*

is the 1- to 8-character alphanumeric name of a named saved system.

#### *fcp\_vdev*

If the specified virtual device is an FCP-attached device, a guest IPL from SCSI disk will be initiated. This feature requires that parameters have previously been defined with the LOADDEV directory statement.

#### **PARM** *parameter\_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. Although the IPL command allows for 64 bytes of parameters, 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 differs from the IPL command, where trailing blanks are included.)

If you specify an IPL statement using 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 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 on the CP DEFSYS command, see the *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 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\_parameter*

specifies a 1 to 8 character load parameter that is used by the IPL'd system. It may be necessary to enclose the load parameter in single quotes. The load parameter may be retrieved by the guest operating system during its IPL sequence. It is preserved until a system-reset-clear operation is performed on the virtual machine; this operation resets it 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 known by the NSS name. Members of a virtual machine group can connect to the signal system service to provide signalling within the group, including 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 may authorize trace data recording to a system data file for the group.
4. See the *z/VM: CP Commands and Utilities Reference* for detailed information on the IPL command (including information about the LOADPARM and PARM options).
5. The IPL action may not be completed if the user does not have certain privilege classes. For example, if IPL CMS is coded in a nonclass G user's directory entry, the IPL fails because CMS issues class G CP commands using DIAGNOSE code X'08'.
6. Specifying the load parameter in single quotes gives you leading blanks, embedded blanks, or single quotes. Remember that any single quote that is a part of the load parameter must be doubled. For example, use 'BETSY'S' to specify BETSY'S as a load parameter.
7. You may 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.

## IPL

### 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's directory entry:

```
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's directory entry:

```
Ipl cms
```

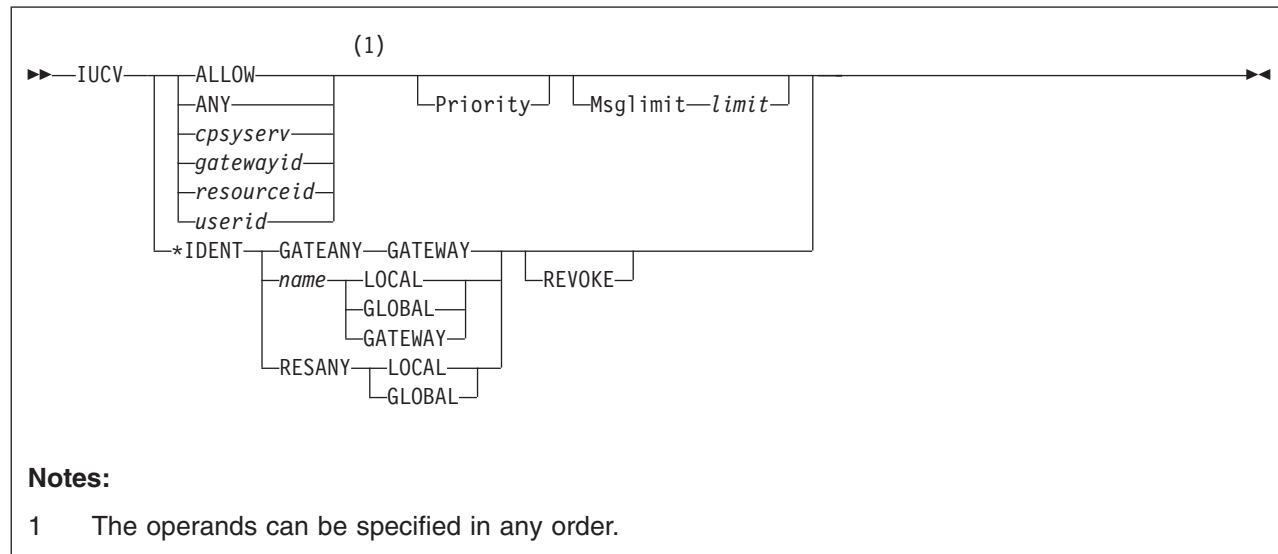
3. To specify that CP is to automatically IPL virtual device number 490 for a virtual machine and pass parameters XYZ to that virtual machine, use the following IPL statement in the virtual machine's directory entry:

```
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's directory entry:

```
Ipl 490 LOADParm 'betsy''s' Parm xyz
```

## IUCV Directory Control Statement (General)



### Purpose

The IUCV statement authorizes a virtual machine to create an inter-user communications vehicle (IUCV) communication path with another virtual machine or a 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 directory entry.)

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 the *z/VM: Connectivity* book.

### How to Specify

If you specify the IUCV statement, it must precede any device statements in a user's directory entry. (For a list of device statements, see Table 20 on page 448.)

You can specify multiple IUCV statements for each virtual machine.

You can specify multiple IUCV statements within a profile 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.

*cpsyserv*

specifies the 1- to 8-character identification of the CP system service to which an IUCV communication path is authorized. The CP system service name must begin with an asterisk (\*). The following CP system services are available:

**\*ACCOUNT**

Accounting system service

**\*BLOCKIO**

DASD block I/O system service

**\*CRM**

Collection resource management system service

**\*LOGREC**

Error recording system service

**\*MONITOR**

\*MONITOR system service

**\*MSG**

Message system service

**\*MSGALL**

Message all system service

**\*RPI**

Access verification system service

**\*SIGNAL**

Signal system service

**\*SPL**

Spool system service

**\*SYMPTOM**

Symptom system service

**\*VSWITCH**

Virtual switch system service

Also, if you want to authorize the VSM to use IUCV to communicate with SNA/CCS, specify *cpsyserv* as \*CCS. For more information on VSM and SNA/CCS, see Chapter 9, "Planning for SNA Console Communication Services (SNA/CCS)," on page 329.

If the virtual machine is running the Enterprise Systems Connection Manager (ESCM) licensed program, specify *cpsyserv* as \*CONFIG.

*gatewayid*

is a 1- 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 nonalphanumeric 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- 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 1- to 8-character user ID of the virtual machine to which an IUCV communication path is authorized.

**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 control statement for the initial invoker of the IUCV CONNECT.

**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.

**Note:** This directory specification does not apply to IUCV CONNECTs involving system services.

**\*IDENT**

allows the virtual machine to connect to the Identify System Service to identify a resource or gateway LU.

**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 directory entries 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 entry
  - c. The target's IUCV statements for an ALLOW entry.
 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 the OPTION statement on page 533.
3. For information on how to code IUCV programs, see *z/VM: CP Programming Services*.
4. Communications with the following system services *require* authorization using an IUCV statement with the system service name as the *cpsyserv* name:

System Service	<i>cpsyserv</i> Name
Accounting	*ACCOUNT
Collection resource management	*CRM
Error recording	*LOGREC
Access verification	*RPI <b>Note:</b> Before communications can be established with *RPI, an external security manager must be installed on the system.
Spool	*SPL
Symptom	*SYMPTOM
Virtual switch	*VSWITCH

5. Communications with SNA/CCS requires authorization using an IUCV statement with \*CCS as the *cpsyserv* name.
6. A virtual machine running the Enterprise Systems Connection Manager (ESCM) licensed program requires authorization using an IUCV statement with \*CONFIG as the *cpsyserv* name.
7. Communications with the following system services *do not require* authorization by the virtual machine. If an IUCV statement is specified, the following *cpsyserv* names should be used:

System Service	<i>cpsyserv</i> Name
Message	*MSG
Message all	*MSGALL <b>Note:</b> Although the PRIORITY and MSGLIMIT operand may be specified on the IUCV statement for *MSG and *MSGALL, the specifications are never used, because the virtual machine will only be receiving messages.
Signal system service	*SIGNAL
DASD block I/O	*BLOCKIO

8. \*MONITOR is the *cpsyserv* name required on the IUCV statement to authorize 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.

**Note:** More than one virtual machine can have an IUCV \*MONITOR statement. This means that every virtual machine with an IUCV statement for \*MONITOR can connect concurrently to \*MONITOR.

9. When the directory entry 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.
10. If a virtual machine must identify more than one resource, you can specify more than one IUCV statement for \*IDENT in the virtual machine's directory entry. 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's directory entry 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's directory entry:
 

```
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's directory entry:
 

```
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's directory entry:

```
Iucv target Priority MsgLimit 100
Iucv Any
```

4. To indicate that any virtual machine can establish a path to the virtual machine whose directory entry 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
```

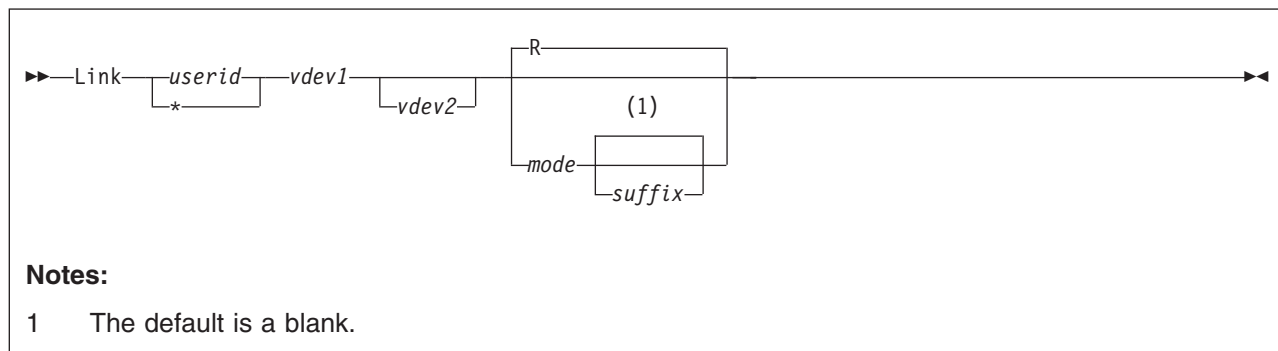
RESMGR3 can identify the resources, RESIDX and RESIDY, as local and global resources. RESMGR3 is not authorized to revoke any global resources. Because of the last IUCV statement, RESMGR3 can identify any resource as a local resource. Also, RESMGR3 can revoke any resource known on the local system as 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 Control Statement (Device)



### Purpose

The LINK statement gets access to another user's minidisk. The target minidisk must be defined by an MDISK statement (page 513) in the target user's directory entry. The DASDOPT statement is an extension to the LINK statement. The Link Directory statement Defaults to NOCTL (works like the Link command) if no DASDOPT statement follows it. For more information on the DASDOPT statement, see "DASDOPT Directory Control Statement (Device)" on page 472.

### How to Specify

If you specify the LINK statement, it must follow any general statements you specify in a user's directory entry. (For a list of general statements, see Table 20 on page 448.)

Multiple LINK statements are allowed within a profile entry. Any LINK statements within a profile entry are added to those in the including user entry with no user-to-profile duplicate virtual device number or address checking performed. LINK statements with duplicate virtual device numbers or addresses within a profile entry are errors.

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.

*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 about stable and exclusive access modes, see the *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.

## LINK

- 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, using the OPTION statement values, LNKSTABL and LNKEXCLU (page 533)
  - Specifically, using the mode suffix letters (S or E) on the LINK or MDISK (page 513) statements.
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 user's directory entry 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.

## 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's directory entry:

```
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's directory entry:

```
Link linkman 291 292 wr
```

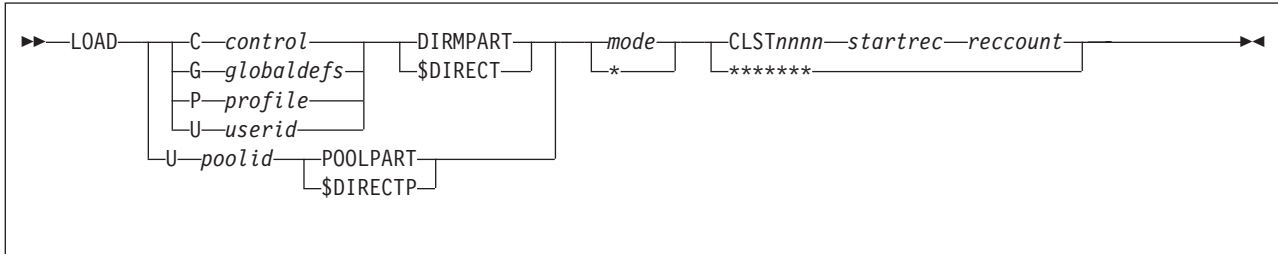
WR indicates that write access is given only if no one else is linked to the disk; if someone else is linked to the disk, read access is given.

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's directory entry:

```
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 Control Statement (Control)



### 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.

### Operands

**C**  
**G**  
**P**

**U** specifies the type of definition to be loaded; C for CONTROL, G for Global definitions, P for PROFILE, and U for USER.

#### *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.

#### *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 a zero or more type P definitions, followed by a zero or more type U definitions.
  - 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 only contain global definition statements, each type P should only contain the named PROFILE definition, and each type U should only contain the named USER definition.
  - The first control statement of each definition should match the definition type. Thus, the first control statement in a type C definition should be a DIRECTORY statement, the first control statement in a type G definition should be a GLOBALDEFS statement, the first control statement in a type P definition should be a PROFILE statement, and the first control statement in a type U definition should be a USER 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.

## LOAD

### Examples

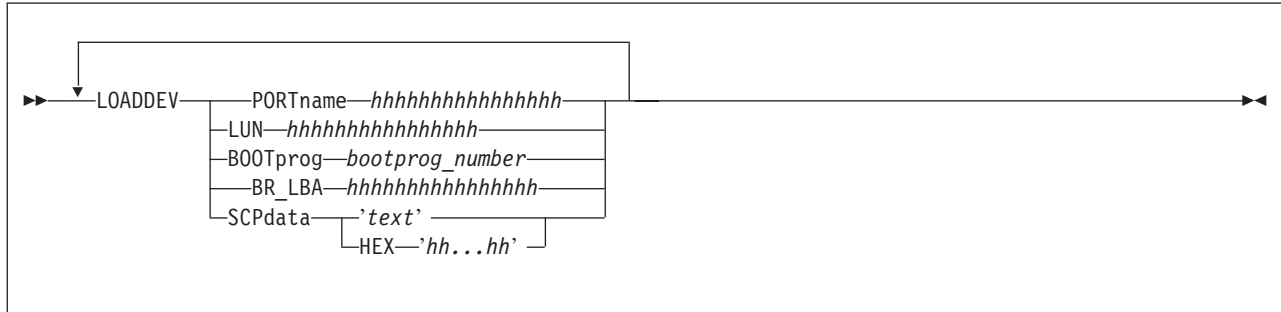
To define a directory in cluster file format with:

- 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
- 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 $dirct1$  dirmpart e *****
Load G glbdefs   dirmpart e *****
Load P dept0000  dirmpart e *****
Load P dept0001  dirmpart e clst0000 1 20
Load U user0001  dirmpart e clst0000 21 5
Load U user0002  dirmpart e clst0000 26 5
Load U user0003  dirmpart e *****
Load U usr       poolpart e *****
Load U usr       poolpart e clst0000 31 2
```

## LOADDEV Directory Control Statement (General)



**Note:** The maximum SCPDATA that can be input is 4096 whether as a text string in quotes, or as hex characters.

### Purpose

Use the LOADDEV directory statement to identify the location of a program to be loaded as a result of a guest IPL from SCSI disk. Additionally, parameters to be passed to the program may also be defined.

### How to Specify

If you specify the LOADDEV statement, it must precede any device statements you specify in a profile or user entry. Multiple LOADDEV statements are allowed within a user or profile entry. Each operand may only be specified once within a user or profile entry. LOADDEV values within a user entry override those in a profile entry. Each LOADDEV statement may only specify one parameter.

### Operands

#### PORTname

The hexadecimal digits designating the one- to eight-byte fibre channel port name of the FCP-I/O device. It must be a value from X'0-FFFFFFFFFFFFFFFF'. There is no default.

#### LUN

The hexadecimal digits representing the one- to eight-byte logical unit number of the FCP-I/O device. It must be a value from X'0-FFFFFFFFFFFFFFFF'. The default is 0.

#### BOOTprog

The decimal value between 0 and 30 representing the script to IPL from.

#### BR\_LBA

The hexadecimal digits designating the logical-block address of the boot record of the FCP-I/O device. It must be a value from X'0-FFFFFFFFFFFFFFFF'. The default is 0.

#### SCPDATA

Designates information to be passed to the program which was loaded during guest IPL. If the program does not require any information, then this parameter is optional. Up to 4096 (4K) characters of data (text or hex) may be entered for SCPDATA. The actual number of input characters could be less 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

## LOADDEV

defined using the HEX option is 2048 (1/2 of the 4K character input limit). The hex string must be an even number of digits. There is no default.

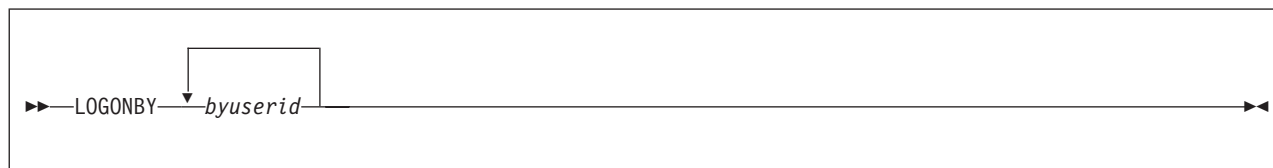
### HEX

Indicates that the parameter value that follows is UTF-8 encoded hex (characters 0-F). If not specified, the value is assumed to be EBCDIC text (codepage 924).

### 'text'

The text string in quotation marks is assumed to be in codepage 924.

## LOGONBY Directory Control Statement (General)



### 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 must precede any device statements in a user's directory entry. (For a list of device statements, see Table 20 on page 448.) A directory entry can contain several LOGONBY statements, specifying up to eight user IDs.

You can use several LOGONBY statements within a profile, but only if no LOGONBY statements are in the including user directory entry.

### 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.

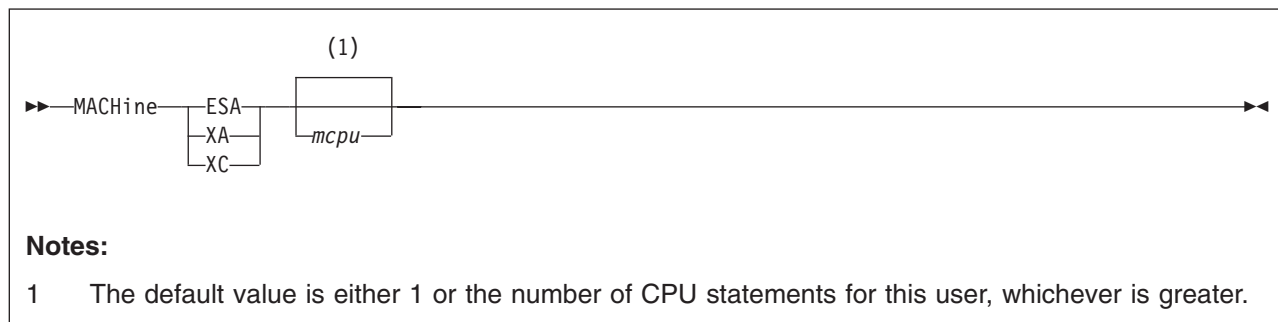
### Usage Notes

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. See the *z/VM: CP Commands and Utilities Reference* for information about the BY operand of the CP LOGON command.
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's directory entry:  
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's directory entry: LOGONBY oper1 oper2 oper3 jones

## MACHINE Directory Control Statement (General)



### Purpose

The MACHINE statement specifies the virtual machine architecture.

### How to Specify

One MACHINE statement is allowed within a profile, but is used only if no MACHINE statement is in the user directory entry. If you specify the MACHINE statement, it must precede any device statements you specify in a user's directory entry. (For a list of device statements, see Table 20 on page 448.)

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 may have the capability to issue an instruction to switch the virtual machine to z/Architecture mode.

#### 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.

#### *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 user's directory entry and no CPU statements are in the user's entry, the user's virtual machine mode is defined by the mode specified by the GLOBALOPTS directory control statement, and the virtual machine has no virtual multiprocessor capabilities. However, if CPU statements are in the user's entry, the number of these statements determines the allowable number of virtual processors.

2. For information on globally changing the default architecture, see “GLOBALOPTS Directory Control Statement (Control)” on page 486.
3. For information on defining multiple virtual processors for a virtual machine, see *z/VM: Running Guest Operating Systems*.
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.

## Examples

1. To specify that a virtual machine can simulate ESA/390 architecture, use the following MACHINE statement in the virtual machine's directory entry:  

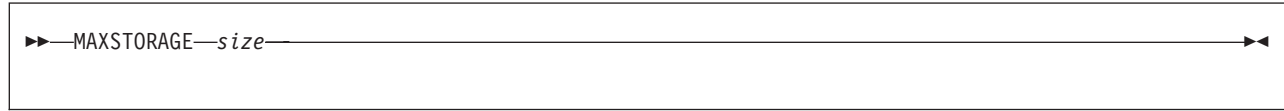
```
Machine esa
```
2. To specify that a virtual machine can simulate ESA/390 architecture and define a maximum of 12 virtual processors, use the following MACHINE statement in the virtual machine's directory entry:  

```
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's directory entry:  

```
Machine xc
```

---

## MAXSTORAGE Directory Control Statement (General)



### Purpose

The MAXSTORAGE statement specifies the maximum virtual storage size for a user.

### How to Specify

The MAXSTORAGE statement, if used, must follow a USER statement and must precede any device statements. A MAXSTORAGE statement is allowed in a profile if the USER statement maximum storage size field of each of the including virtual machines is either omitted or specified as an asterisk (\*). A MAXSTORAGE statement in a user directory entry overrides one in a profile.

### Operands

*size*

is the maximum storage size. Specify it as *nu*, where *n* is a decimal number and *u* is the unit of measure:

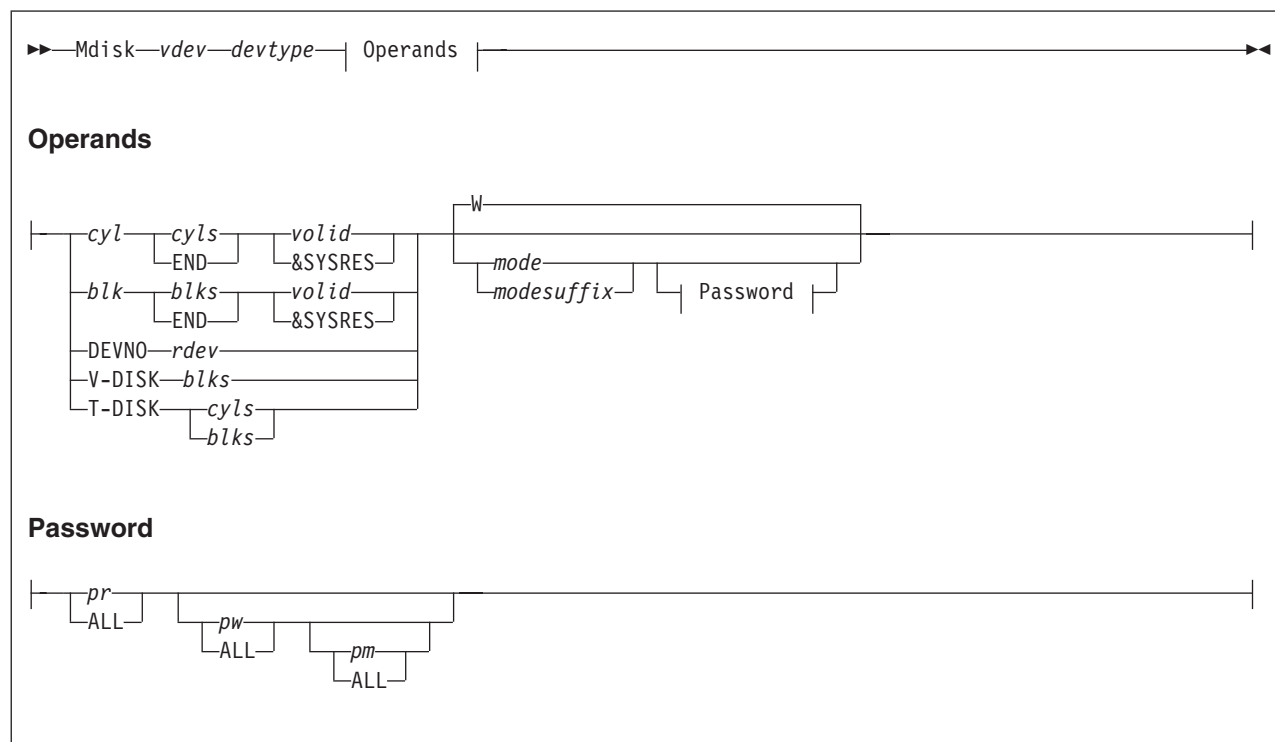
- K** kilobytes -  $2^{10}$
- M** megabytes -  $2^{20}$
- G** gigabytes -  $2^{30}$
- T** terabytes -  $2^{40}$
- P** petabytes -  $2^{50}$
- E** exabytes -  $2^{60}$

### Usage Notes

1. A maximum storage setting on a USER statement overrides a MAXSTORAGE statement in a profile.
2. A MAXSTORAGE statement in a user's directory entry overrides one in a profile.



## MDISK Directory Control Statement (Device)



### Purpose

The MDISK statement defines minidisks (virtual disks) for virtual machines: permanent minidisks, temporary minidisks, and virtual disks in storage. The DASDOPT statement is an extension to the MDISK statement. The MDISK statement defaults to DEVCTL if no DASDOPT statement follows. For more information about the DASDOPT statement, see “DASDOPT Directory Control Statement (Device)” on page 472. You can use one DASD to provide several minidisks or you can use one DASD to provide one minidisk.

**Note:** 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 not allowed within a profile entry. If you specify the MDISK statement, it must follow any general statements you specify in a user’s directory entry. (For a list of general statements, see Table 20 on page 448.)

### 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 is the device type of the real device on which the minidisk resides. Valid device types are:

## MDISK

3380

3390

9336

FB-512

FB-512 is a generic specification for FBA DASD and may 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 is the device type of the device that is simulated in system storage. A virtual disk in storage simulates an FBA minidisk; therefore, the device type must be FB-512.

*cyl*

*blk*

is the starting cylinder (CKD/ECKD) or block number (FBA) on the real DASD you specify on the *valid* operand.

### 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.

### T-DISK

indicates that the minidisk is temporary, that is, it is created from a preselected pool of temporary disk space when the virtual machine logs on and is returned to the pool when the virtual machine logs off or the virtual device is detached. Temporary disks cannot have passwords.

*cyls*

*blks*

is the number of cylinders (CKD/ECKD) or blocks (FBA) allocated to the minidisk. Table 23 on page 517 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 should be defined with the remaining available cylinders or blocks. This is useful when defining a full-pack minidisk. See usage note 4 on page 518 for more information.

*rdev*

is the real device number.

*valid*

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 be used to represent the volum serial number of the system residence volume for minidisks defined using the &SYSRES operand on the MDISK statement.

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.

*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. In the case of a previous write or stable (SR, SW, SM) access, read-only access is established. In the case of an exclusive mode access existing, 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 about stable and exclusive access modes, see the *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 using CP's virtual reserve/release.

**S** Authorizes the virtual machine in whose directory definition this statement

## MDISK

appears the ability to use the LINK command stable access modes (SR, SW, SM) against the specified user's minidisk. For more information, see the *z/VM: CP Commands and Utilities Reference*.

- E** Authorizes the virtual machine in whose directory 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 the *z/VM: CP Commands and Utilities Reference*.
- 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.

*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 23. Maximum Minidisk Sizes

Device Type	Models	CMS/VSAM, CMS 512-Byte, 1 KB, 2 KB, or 4 KB Format
3380	A04, AA4, B04, AD4, BD4, AJ4, BJ4, CJ2	885 cylinders
3380	AE4, BE4	1770 cylinders
3380	AK4, BK4	2655 cylinders
3390 <sup>1</sup>	A14, A18, B14, B18, B1C	1113 cylinders
3390 <sup>1</sup>	A24, A28, B24, B28, B2C	2226 cylinders
3390 <sup>1</sup>	A34, A38, B34, B38, B3C	3339 cylinders
3390 <sup>2</sup>	A98, B9C	10017 cylinders <sup>3, 4, 6</sup>
9336	020	2147483640 blocks <sup>5, 6</sup>
FB-512 (virtual disk in storage)	(not applicable)	4194296 blocks

**Notes:**

<sup>1</sup> These capacities apply to 3390 mode and 3380 track compatibility mode.

<sup>2</sup> 3380 track compatibility mode is not supported for these 3390 models.

<sup>3</sup> Minidisks 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.

<sup>4</sup> Value can be up to the maximum number of cylinders on the DASD (CMS and GCS are limited to 32767 cylinders).

<sup>5</sup> z/VM supports a SCSI disk, which is emulated as a 9336-020, up to a capacity of 2147483640 512-byte blocks (1 terabyte minus 1 page). Note, however, that directory, paging, and spooling allocations must reside within the first 16777216 blocks (blocks 0 to 16777215) of a CP-formatted disk.

<sup>6</sup>Due to CMS file system limitations for 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 is 32767 cylinders for a 3390 disk on an ESS device, or about 22 GB of data. IBM suggests that customers defining FBA SCSI disks for use by CMS 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 the *z/VM: CMS Commands and Utilities Reference*. The maximum size for FBA SCSI disks supported for use by CMS or GCS is 381 GB.

**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 as described in the *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. If a user has written sensitive data on temporary disk space, he should reformat the disk before detaching it from his 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. This action removes any risk of security exposure.

## MDISK

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. To do this, specify '0' to 'END' on the MDISK statement. Note that:
  - A minidisk beginning with cylinder 1 or block 1 is not a full-pack minidisk.
  - You should use the keyword 'END' so your MDISK statement is not dependant on the size and type of DASD.

You can optionally 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 should not define any other type of overlap. CP does the overlap checking to ensure the integrity of link access modes. This is not a feature to exploit.

If you do overlap other minidisks, you may be:

- Prevented from linking to those minidisks and receive messages HCP104E, HCP105E, and HCP106E, or
  - Linked read only to those minidisks and receive messages HCP101E, HCP102E, and HCP103E.
6. The 3990s used for CP volumes will be 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. The ability to change modes is restricted to guests with the OPTION statement value MAINTCCW or DEVMAINT. The use of ICKDSF is the recommended method to change modes.
  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, either globally using the OPTION statement values, LNKSTABL and LNKEXCLU, or specifically, 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 defining full-pack minidisks where the guest will control the duplexing of the device.
  10. The DEVNO operand should be used when defining multiple minidisks with identical volume serial numbers.
  11. Refer to the appropriate storage control manuals for additional dual copy information.
  12. For information on the use of the MDISK statement for the CMS shared file system, refer to the *z/VM: CMS File Pool Planning, Administration, and Operation* book.
  13. If an external security manager (ESM) is installed, you may not be authorized to use the MDISK statement for all minidisks and all access modes. The ESM may downgrade certain requests for write access to read access. For additional information, contact your security administrator.
  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 may 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 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, but any new links are made using the new definition of the minidisk. Unpredictable results may occur.
17. If you change the definition of a virtual disk in storage 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.
18. Some programs may not support the use of &SYSRES, and require the use of a synonym. The synonym is the value 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 may be used to define a minidisk on the system residence volume.

## Examples

1. To define a minidisk that:
  - Has the virtual device number 191
  - Takes up 5 cylinders beginning at cylinder 100 of the 3380 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's directory entry:

```
Mdisk 191 3380 100 5 mddasd rr
```
2. To define a minidisk that:
  - Has the virtual device number 291
  - Takes up 10 cylinders beginning at cylinder 105 of the 3380 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's directory entry:

```
Mdisk 291 3380 105 10 mddasd mr all
```
3. To define a minidisk that:
  - Has the virtual device number 198
  - Takes up 6000 blocks beginning at block 12,000 of a supported FBA DASD with volume serial FBDASD

## MDISK

- Is accessible only in multiple-write (MW) modes
- Can be shared using CP's virtual reserve/release
- Has a read password of 12WE45

use the following MDISK statement in the virtual machine's directory entry:

```
Mdisk 198 9336 12000 6000 fbdasd mrv 12we45
```

4. To define a minidisk that:

- Has the virtual device number 198
- Takes up 6000 blocks beginning at block 12,000 of the 9336 DASD with volume serial FBDASD
- Will have the default access mode of Write when the virtual machine is logged on, but will specifically authorize the user to access it with a stable access mode using the LINK command at a later time.
- Can be shared using CP's virtual reserve/release
- Has a read password of 12WE45

use the following MDISK statement in the virtual machine's directory entry:

```
Mdisk 198 fb-512 12000 6000 fbdasd mvs 12we45
```

5. To define 5 cylinders of temporary 3380 DASD space at virtual device number 391, use the following MDISK statement in the virtual machine's directory entry:

```
Mdisk 391 3380 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's directory entry:

```
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's directory entry:

```
Mdisk 199 3380 000 885 mddasd mr  
Mdisk 199 3380 000 end mddasd mr
```

8. To define a pair of full-pack minidisks that are candidates for duplexing and:

- 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's directory entry:

```
mdisk 198 3390 devno 200 mr all  
mdisk 199 3390 devno 201 mr all
```

9. To define a virtual disk in storage that:

- Has virtual device number 401
- Consists of 8000 blocks
- Is accessible in read or write mode
- Can be shared using CP's virtual reserve/release

use the following MDISK statement in the virtual machine's directory entry:

```
mdisk 401 fb-512 v-disk 8000 mrv
```

10. To define a minidisk using 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 may not be dynamically modified during the I/O operation in these cases.

**Note:** For the Read Record Zero case, the above 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. There is no restriction for dedicated disks.
3. If the user's channel program for a count-key-data minidisk does not perform a seek operation, CP inserts a positioning seek operation into the user's channel program to prevent accidental accessing. Thus, certain channel programs may 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 initiated through an SIOF or SSCH may not indicate deferred CC 1.
4. A DASD channel program may give results on dedicated DASD that differ from results on minidisks having nonzero relocation factors if the channel program includes multiple-track operations and 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, may terminate prematurely if a real record 0 is encountered. Minidisks with nonzero relocation factors are not usable under OS and OS/VS systems. This 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) may 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, but the user must be aware that the track descriptor record may 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 issuing a Write Record Zero or a Write Home Address CCW on a nondedicated DASD is *not* defined as a full-pack minidisk.
7. When performing DASD I/O, if the record field of a Search ID Argument is zero when a virtual SIO, SIOF, or SSCH is issued, but the Search ID Argument is dynamically read by the channel program before the Search ID CCW is

## MDISK

executed, the real Search ID uses the relocated search argument instead of the argument that was read dynamically. To avoid this problem, the record field of a Search ID Argument should not be set to binary zero if the search argument is to be dynamically read or if the search ID on record 0 is not intended.

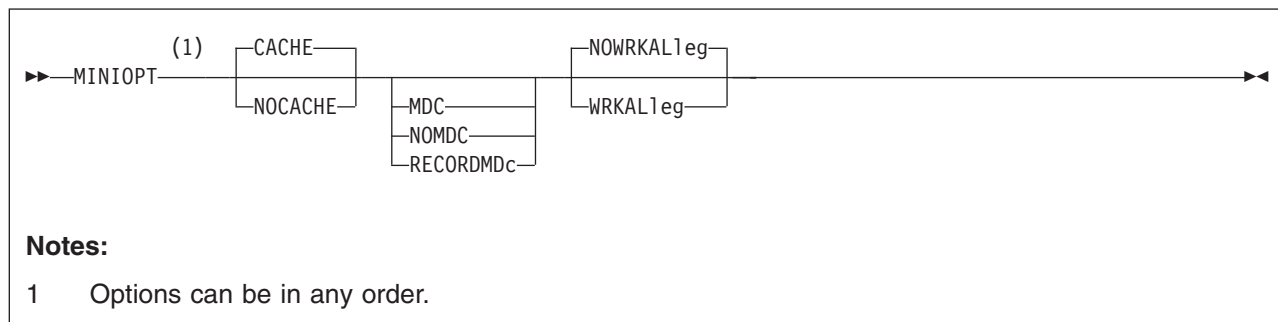
8. The following CCWs are restricted to virtual machines with MAINTCCW authority. You specify this authority by coding the MAINTCCW operand on the OPTION statement in the virtual machine's user directory entry.
  - Diagnostic Write Home Address
  - Diagnostic Read Home Address
  - Write Record Zero
  - Diagnostic Load
  - Diagnostic Sense/Reset Allegiance: CP treats the command as a reset allegiance CCW and accepts it as long as 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 3375, 3380, 3390, and 9345 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 running in a virtual machine.
11. The 3390s used for 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. 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 should never 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 for these units. For more information, see "DASDOPT Directory Control Statement (Device)" on page 472.
14. Minidisks defined on an FBA volume which are not page-aligned will not be eligible for minidisk cache and will therefore not benefit from its performance improvement. Aligning minidisks on page boundaries is highly recommended. Minidisks as well as full-pack minidisk volumes that are not page-aligned may result in residual blocks that are not formatted and utilized. Additional restrictions apply for SFS Filepool minidisks which are not page-aligned. For more information see the *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 the above usage note, residual blocks on FBA devices that have not been ICKDSF formatted may still be defined and used as MDISK space.

15. Minidisks defined on an FBA volume which are not page aligned cannot be mapped to an address space using minidisk mapping.

## MINIOPT Directory Control Statement (Device)



### 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.

**CAUTION:**

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.

## 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.

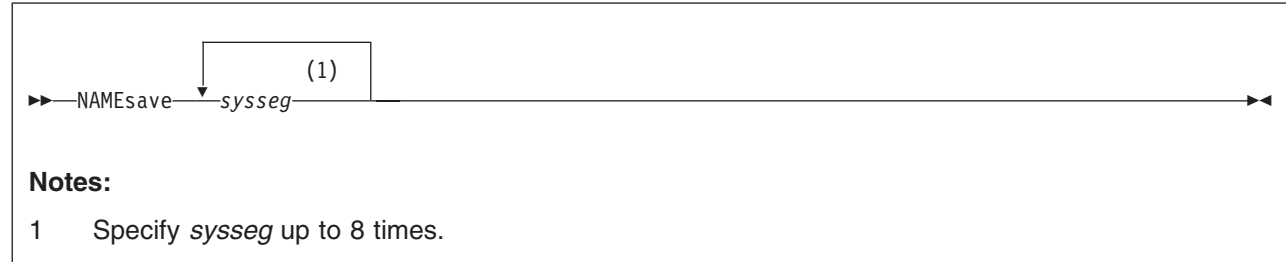
## MINIOPT

- 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 DASDOPT statement (page 472).
- 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 as well as main storage and expanded storage minidisk cache are all being overwhelmed by excessive unreferenced records being read by the full track cache support.

## 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`

## NAMESAVE Directory Control Statement (General)



### 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.)

### How to Specify

You can specify multiple NAMESAVE statements for each virtual machine. If you specify the NAMESAVE statement, it must precede any device statements you specify in a user's directory entry. (For a list of device statements, see Table 20 on page 448.)

Multiple NAMESAVE statements are allowed within a profile entry. NAMESAVE statements within a profile entry are added to those in the including user 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 the *z/VM: Virtual Machine Operation* book.
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.

## NAMESAVE

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's directory entry:

```
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's directory entry:

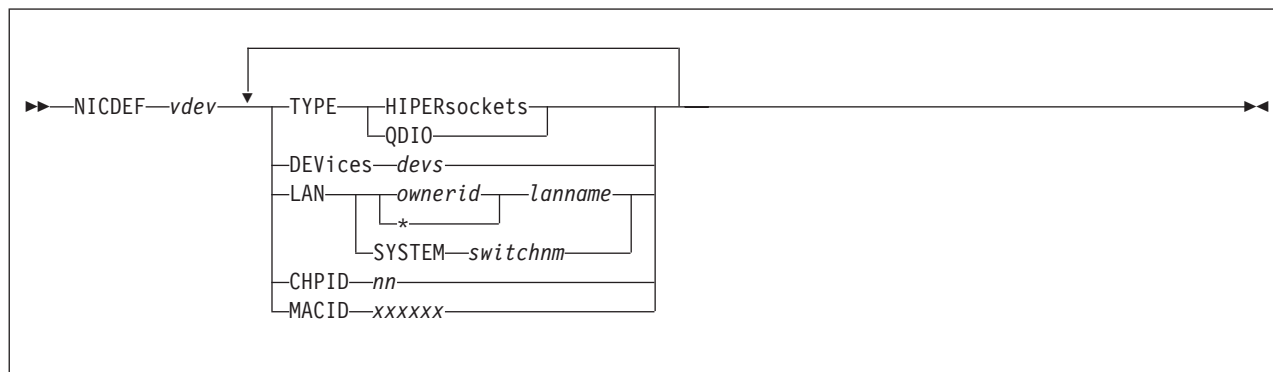
```
NameSave mvsxa1 mvsxa2 xaseg
```

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's directory entry:

```
NameSave xaseg1
```



## NICDEF Directory Control Statement (Device)



### Purpose

The NICDEF statement defines virtual network adapters that are fully simulated by CP. Use NICDEF to create a virtual HiperSockets or QDIO adapter in the virtual machine, and (optionally) connect it to a VM LAN or Virtual Switch.

### How to Specify

If you specify the NICDEF statement, it must follow any general statements you specify in a user's directory entry. (For a list of general statements, see Table 20 on page 448.)

The SPECIAL statement may 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.

NICDEF statements are allowed within a profile entry if no conflicting statements for the virtual device numbers are in the profile. When a NICDEF statement in the user entry matches the virtual device number in the profile, the USER entry replaces the profile entry.

### Operands

*vdev*

is the base (or first) device address in a series of virtual I/O devices that belong to the same unit.

#### **TYPE HIPERsockets | QDIO**

indicates the type of adapter to be created. This is a required keyword, and it must be the first keyword specified.

Use HIPERsockets for a simulation of the zSeries HiperSockets feature or QDIO for a simulation of the OSA Express feature in QDIO mode. If a LAN is identified in this statement or another with the same *vdev*, the NIC is automatically coupled to the specified *ownerid lanname*.

#### **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.

## NICDEF

Table 24. 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.

**Note:** If you must define an adapter type, ensure that it is compatible with the intended Guest LAN or virtual switch.

### CHPID *xx*

is a two 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 six hexadecimal digits) used as part of the adapter MAC Address. During LOGON, your MACID (three bytes) is appended to the system MACPREFIX (three 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. 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.

## Usage Notes

1. When a simulated NIC is defined (HIPERsockets or QDIO device types), 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 VM LAN or Virtual Switch by adding the optional LAN parameter. However, if the designated VM 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 VM LAN or Virtual Switch during system initialization.

## 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:

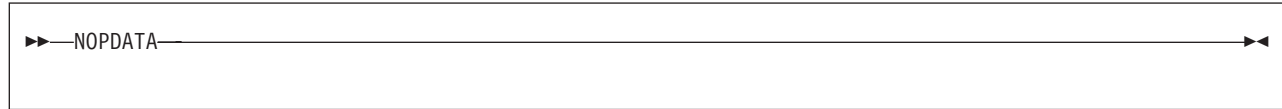
```
NICDEF FD20 TYPE HIPERS
```

```
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.

---

## NOPDATA Directory Control Statement (General)



### 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

If you specify the NOPDATA statement, it must precede any device statements you specify in a user's directory entry. (For a list of device statements, see Table 20 on page 448.)

Multiple NOPDATA statements are allowed within a profile entry.

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

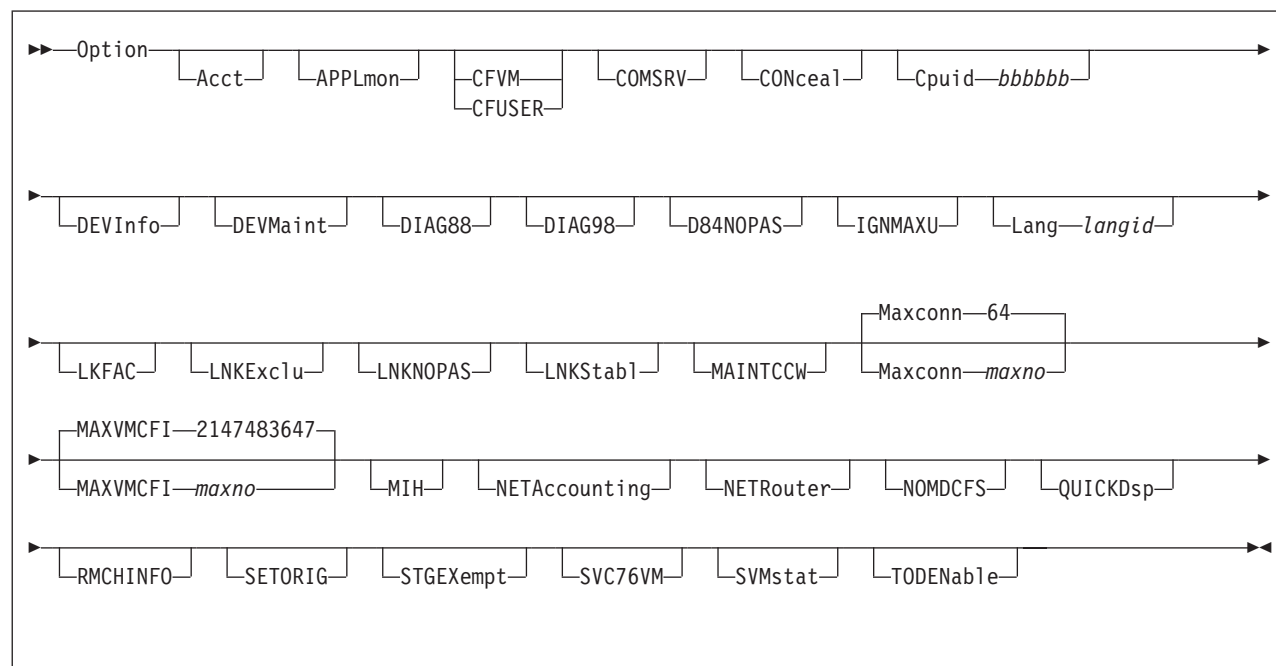
1. The NOPDATA statement should appear in the directory entry of the RSCS virtual machine. For more information, see the *RSCS Planning and Installation* book, SH24-5219.
2. Do not use data chaining with NOP CCWs.

### 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's directory entry:

```
NoPData
```

## OPTION Directory Control Statement (General)



### Purpose

The OPTION statement specifies special characteristics available to the virtual machine.

### How to Specify

If you specify the OPTION statement, it must precede any device statements you specify in a user's directory entry. (For a list of device statements, see Table 20 on page 448.)

Any number of OPTION statements are allowed within a profile, and are added to any OPTION statements in the user directory entry.

### Operands

#### Acct

specifies that the virtual machine can issue a DIAGNOSE code X'4C' to generate accounting records. For more information about DIAGNOSE code X'4C', see the *z/VM: CP Programming Services* book.

#### APPLmon

authorizes the virtual machine to issue DIAGNOSE code X'DC' by which an application may 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 about DIAGNOSE code X'DC', see the *z/VM: CP Programming Services* book.

#### 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 may not be specified with CFUSER or RMCHINFO.

## OPTION

### **CFUSER**

indicates that the user is allowed to define and couple Message Devices to a Coupling Facility Service Machine. CFUSER may not be specified with CFVM.

### **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.

### **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 directory entry specifies OPTION CONCEAL has the break key (for 3270 terminals) disabled at logon. If the user subsequently deactivates the protected application environment 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.

### **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 owned by this or another user ID. For more information about DIAGNOSE code X'E4', see the *z/VM: CP Programming Services* book.

### **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 about DIAGNOSE code X'E4', see the *z/VM: CP Programming Services* book.

**DIAG88**

specifies that the virtual machine is authorized to use DIAGNOSE code X'88' to validate user authorizations and link minidisks. A user whose directory entry contains this operand can run programs that access other users' minidisks without requiring passwords. For information about DIAGNOSE code X'88', see the *z/VM: CP Programming Services* book.

**DIAG98**

specifies that the virtual machine is authorized to use the DIAGNOSE code X'98' real I/O facility. A user whose directory entry 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 information about DIAGNOSE code X'98', see the *z/VM: CP Programming Services* book.

**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 only takes effect if there is no access control mechanism, such as RACF, in place. For more information about DIAGNOSE code X'84', see the *z/VM: CP Programming Services* book.

**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 three ways:

1. Using the FEATURES statement with the MAXUSERS operand in your system configuration file. For more information, see page 136.
2. Using the CP SET MAXUSERS command. For more information, see the *z/VM: CP Commands and Utilities Reference*.
3. Using the SYSMAXU macroinstruction in HCPSYS. For more information, see page 686.

**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 simulated for TPF guests.
- Real Multi-Path Lock Facility (MPLF) support. MPLF is a lock facility available on the D/T3990 model 6 control unit.

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 used by the TPF system.

**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 allowing 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 Control Statement (Device)" on page 500 and "MDISK Directory Control Statement (Device)" on page 513. The exclusive access modes negate

## OPTION

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.

### **LNKNOPAS**

specifies that this virtual machine can link to any other virtual machine's DASD without password authorization. When LNKNOPAS is specified, password authorization may still be required when an external security manager (ESM) is installed. For more information, refer to the documentation provided by your ESM.

### **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 access modes, see "LINK Directory Control Statement (Device)" on page 500 and "MDISK Directory Control Statement (Device)" on page 513. These modes negate the ability of other users to acquire write access the specified minidisk. This ensures that the data can not be changed underneath the user.

### **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.

### **Maxconn** *maxno*

specifies the maximum number of IUCV and APPC/VM connections 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. See the *z/VM: CMS File Pool Planning, Administration, and Operation* for more information.

### **MAXVMCFI** *maxno*

specifies the maximum number of VMCF inbound send messages, including those initiated by SMSG, plus IDENTIFY final response interrupts that may 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.

### **NETAccounting**

specifies that Network Data Transmission account records (type 0C) should be generated for this user. For more information on Network Data Transmission account records, see "Accounting Records Network Data Transmissions (Record Type C)" on page 307.



**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 designated as a router (over a VM LAN), will be reported in account records separate from transmissions with non-routers. For more information on Network Data Transmission account records, see “Accounting Records Network Data Transmissions (Record Type C)” on page 307.

**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 very 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.

**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 may require real machine configuration information) is running under CMS. RMCHINFO should not be specified for guest operating systems.

**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 about DIAGNOSE code X'F8', see the *z/VM: CP Programming Services* book.

**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.

**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.

**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.

**TODENable**

specifies that the user may change the virtual machine's Time-of-Day (TOD) clock with the SCK instruction or the SET VTOD CP command.

## Usage Notes

1. To run the device support facilities (ICKDSF) program in a virtual machine, you must specify the MAINTCCW operand.
2. The CPU statement overrides the specification on the OPTION statement.
3. 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.

## OPTION

4. You may not specify `OPTION CONCEAL` if the directory entry for this user defines a multiprocessor configuration. The protected application facility supports only virtual uniprocessor systems.
5. 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 and thus be protected from further CP interaction.
6. Though the break key is initially disabled at logon when the `OPTION CONCEAL` statement is used, its setting may 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` may 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 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 which should never be forced off.

## 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 it detects
  - The virtual machine is authorized to create up to 100 IUCV pathsuse the following `OPTION` statement in the virtual machine's directory entry:

```
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 it detects
  - The virtual machine is authorized to create up to 100 IUCV pathsuse the following `OPTION` statement in the virtual machine's directory entry:

```
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 user's directory entry:

- 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 App1Mon
- 7. To give the virtual machine the authority to enter a CP LINK *userid* 291 392 SR command, use:
  - Option LnkStab1
- 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 Control Statement (General)

```
►—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

If you specify the POOL statement, it must be the only control statement other than a USER statement that you code in a user's directory entry. POOL statements are not allowed within a profile 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 statement.
2. The user ID specified on the preceding USER statement must be from 1 to 3 characters in length.
3. A directory entry that contains a POOL statement must comprise only a USER statement and a POOL statement. Any other directory statements specified are invalid.

### Examples

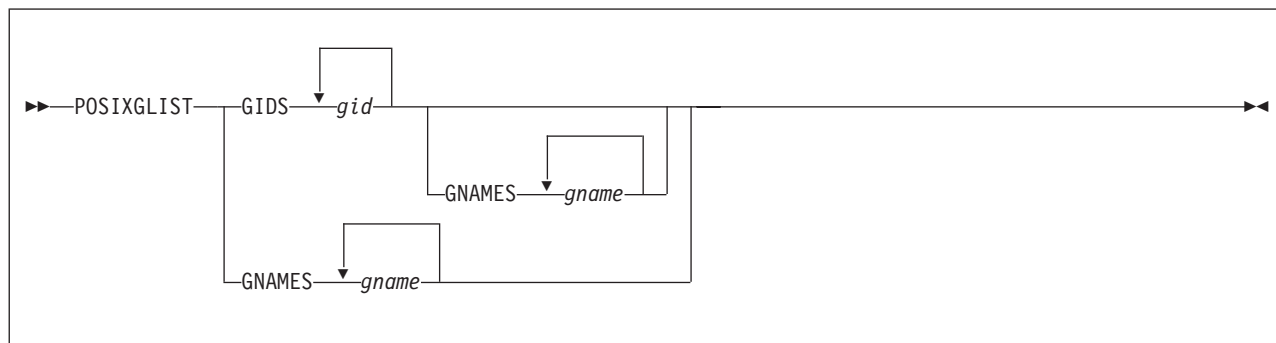
A group of 25 virtual machines with identical configurations could be defined by the following directory entry:

```
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 Control Statement (General)



### 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

If you specify the POSIXGLIST statement, it must precede any device statements you specify in a profile or user entry. (For a list of device statements, see Table 20 on page 448.)

Multiple POSIXGLIST statements are allowed within a user or profile entry. The POSIXGLIST information from a user entry completely replaces that from an included PROFILE.

The POSIXGLIST statement may be continued across multiple records in the source directory file. For detailed information about continued statement rules, see "Continued Directory Control Statements" on page 452. 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

#### **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 control statement may be overridden by information provided by the ESM. Consult your ESM documentation for more information.

## POSIXGLIST

2. A group's *gid* or *gname* may only be specified one time by GID or GNAME on the POSIXGLIST statements in a user entry or a profile; that is, duplicate *gids* or *gnames* 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 Control Statement (Control)

```
▶—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 control statement and before any USER or PROFILE definitions). 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 control 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 control 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.

## POSIXGROUP

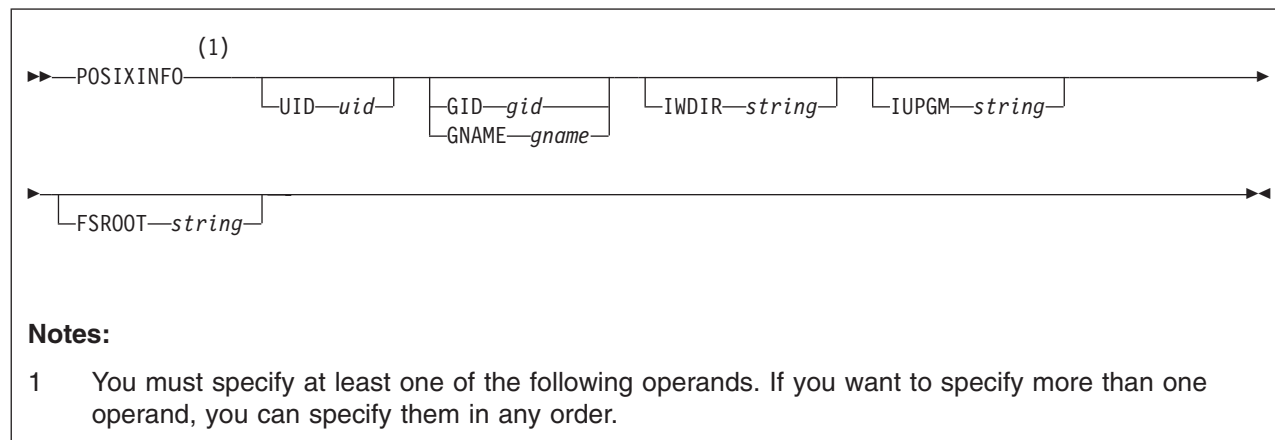
### Examples

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 Control Statement (General)



### 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 or user entry. (For a list of device statements, see Table 20 on page 448.)

Multiple POSIXINFO statements are allowed within a user or profile entry. Each operand may only be specified once within a user or profile entry. POSIXINFO values within a user entry override those in a profile entry.

The POSIXINFO statement may be continued across multiple records in the source directory file. For detailed information about continued statement rules, see "Continued Directory Control Statements" on page 452. 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

#### 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 control 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 non-blank 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*. See "Quoted String Operands" on page 452 for detailed information on how to specify these operands.

## Usage Notes

1. If your installation has installed an External Security Manager (ESM), the information specified on this directory control 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 specifying the UID in a profile entry is not recommended, 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 via a profile entry. Similar consideration should be given to assigning GID/GNAME with 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

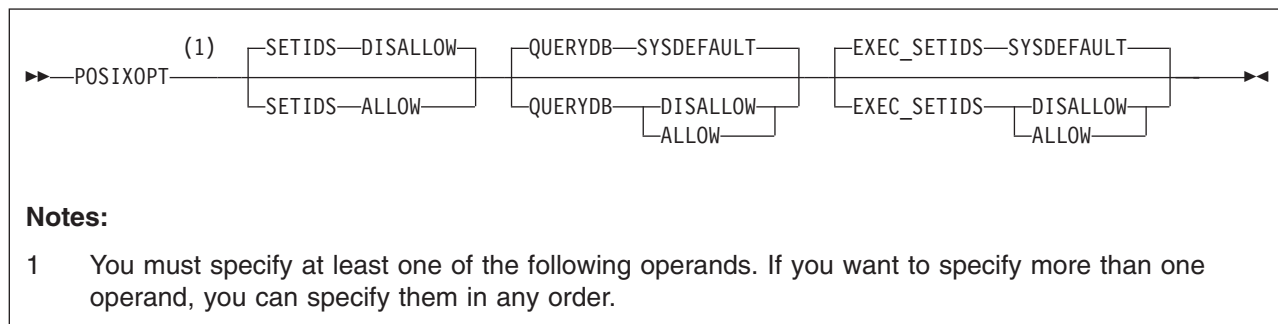
1. To define a userid CLYDE with 100 as UID, 200 as primary GID, /home/clyde as initial working directory, pxshell as initial user program, and ../VMBFS:VMSYS:ROOT/ as the file system root, code the following directory statements:

```
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 Control Statement (General)



### 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 or user entry. (For a list of device statements, see Table 20 on page 448.)

Multiple POSIXOPT statements are allowed within a user or profile entry. Each operand may only be specified once within a user or profile entry. POSIXOPT values within a user entry override those in a profile entry.

The POSIXOPT statement may be continued across multiple records in the source directory file. For detailed information about continued statement rules, see "Continued Directory Control Statements" on page 452.

### 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 user entry. 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. See “USER\_DEFAULTS System Configuration File Statement” on page 248 for details. 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 user entry.

**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. See “USER\_DEFAULTS System Configuration File Statement” on page 248 for details. 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 control 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.

## POSIXOPT

use the following POSIXOPT statement in the user's directory entry:

```
Posix0pt SetIds allow QueryDb allow Exec_Setids disallow
```

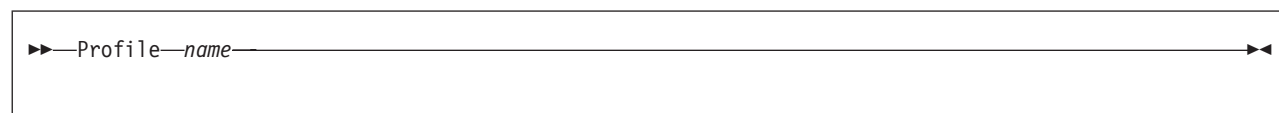
2. To specify that:

- 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 user's directory entry:

```
Posix0pt QueryDb disallow
```

## PROFILE Directory Control Statement (Control)



### Purpose

A PROFILE statement defines the start of a profile entry in the source directory.

### How to Specify

When PROFILE is specified, all profile entry definitions must follow the last DIRECTORY statement and the global definition section and precede the first USER statement. There are no restrictions on the number of profile entries that may be specified.

PROFILE statements are not allowed within a profile entry.

Directory profiles are implemented by defining a profile entry in the source directory and specifying an INCLUDE statement in the user entries that refer to the profile. (For more information on the INCLUDE statement, see page 487).

### Operands

*name*

specifies the name assigned to the profile entry and is used to reference 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 may be specified on a PROFILE statement.

### Usage Notes

Profile entries represent defaults to be set if not specified in the including user definition.

Basically, when these directory control statements are used in a profile definition, they perform the same function as they do when used in a user definition. However, they may operate slightly differently when used in both the profile and user definitions.

1. The following control statement types, when placed in the user definition, completely supersede their profile counterparts:

ACCOUNT	IPL	SPOOLFILE
ACIGROUP	MACHINE	STDEVOPT
AUTOLOG	POSIXGLIST	XAUTOLOG
CLASS	SHARE	XSTORE
CONSOLE		
D8ONECMD		

2. Area operands of the SCREEN statements in the user definition supersede their profile counterparts. This relationship also holds true when the user definition redefines a superset or subset of those areas defined by the profile.

## PROFILE

3. STDEVOPT, POSIXINFO, and POSIXOPT statement operands explicitly coded in a user entry supersede their profile counterparts. However, these statements' operands explicitly coded in a profile supersede uncoded defaults in a user entry.
4. When CPU statements are placed in the user definition, they supersede their profile counterparts where the CPU number is the same; where the CPU numbers are different, however, the definitions are simply appended. The order of concatenation is user entries followed by profile entries.
5. The XCONFIG statement, when placed in the user definition, supersedes its profile counterpart where the first (major) operand is the same; where the first operands are different, the definitions are combined.
6. When the following control statement types are used in a profile definition, they are processed exactly as if they were multiple occurrences of the statement types within the including user definition. (See Usage Note 4.)

APPCPASS      IUCV                      NAMESAVE      NOPDATA      OPTION

7. When the following control statement types are used in a profile definition, they are processed as normal except that duplicate definition of an address in the profile definition causes a duplication error. However, duplication of an address used in the including user definition 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 will be processed first; if this definition fails, the profile device definition for the duplicated address will be attempted.

CONSOLE      DASDOPT      DEDICATE      LINK              SPECIAL  
SPOOL

8. The following control statements are not allowed in a profile definition:

DIRECTORY      GLOBALDEFS      GLOBALOPTS      INCLUDE      LOAD  
MDISK              MINIOPT      POOL              POSIXGROUP      PROFILE  
USER

9. A profile entry begins with a PROFILE statement and ends with the next PROFILE statement or the first USER statement.

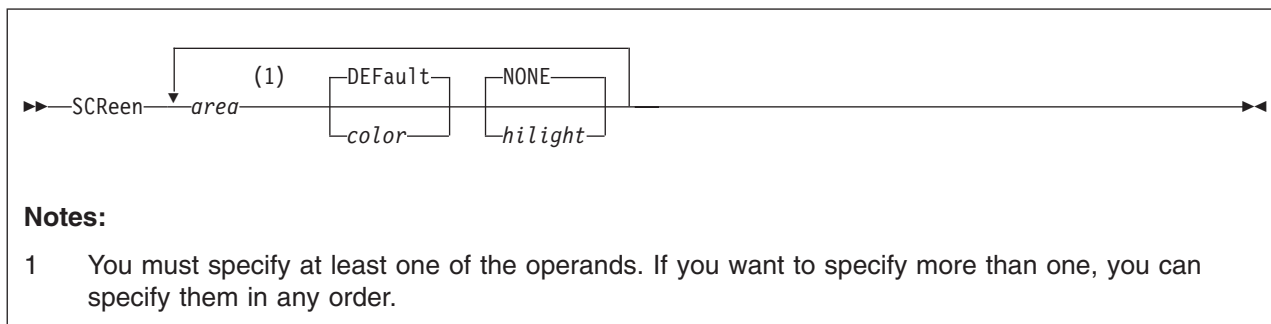
## Examples

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 valid
Profile grpluser
  Acigroup group1
  Ipl cms
  :                               ← other control statements common to this group
User john1 ...                   ← USER statement operands for this user
  Include grpluser
  :                               ← control 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 Control Statement (General)



### 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's directory entry define the extended color and extended highlighting options for the virtual machine console.

## How to Specify

You may include as many SCREEN statements as you need to define any areas of the screen you want, but the SCREEN statements must be contiguous. If used, SCREEN statements must follow the USER statement and precede the first device statement. (For a list of device statements, see Table 20 on page 448.)

Any number of SCREEN statements are allowed within a profile, and are added to any SCREEN statements in the user directory entry. Duplicate area specifications within a profile are not allowed. The user entry overrides a profile entry for any user-to-profile duplicate are a specification. This relationship also holds true when the user definition redefines a superset or subset of those areas defined by the included profile.

## Operands

*area*

is the area of the screen. Area can be:

<b>ALL</b>	designating the entire screen. If this area is specified, none of the following areas can be used.
<b>INArea</b>	the input area
<b>STAtarea</b>	the system status area
<b>OUTarea</b>	the output areas. If this area is specified, none of the following areas can be used.
<b>CPOut</b>	the output from CP
<b>VMOut</b>	the output from CMS or the virtual machine operating system running in the user's virtual machine
<b>INRedisp</b>	the input redisplay

*color*

is the color attribute to be assigned to an area of the screen. Color can be:

<b>BLUe</b>	<b>GREen</b>	<b>PINK</b>	<b>RED</b>
<b>TURquois</b>	<b>WHItE</b>	<b>YELlow</b>	<b>DEFault</b> (green and white)

*highlight*

is the extended highlight value to be assigned to an area of a screen. The

## SCREEN

highlight value can be:

<b>BLink</b>	blinking
<b>NONE</b>	(the default) no extended highlighting
<b>REVvideo</b>	reverse video
<b>UNDERlin</b>	underlining

## Usage Notes

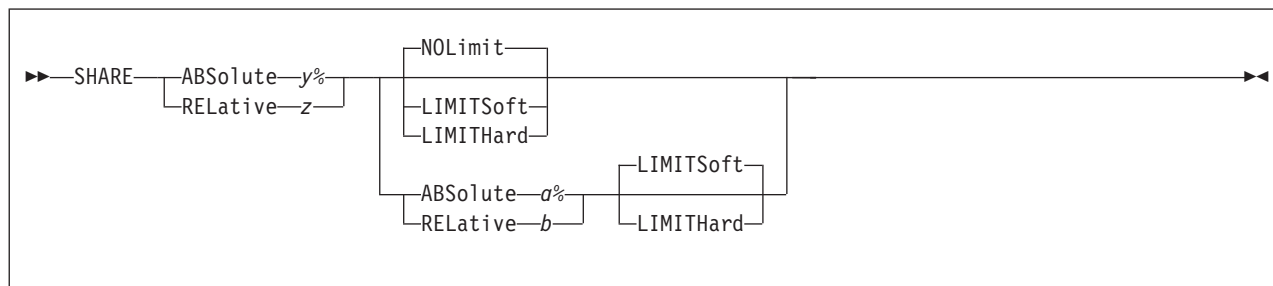
1. A default value of NONE is applied for any unspecified extended highlight attribute. DEFAULT is used for any unspecified extended color attribute. The DEFAULT color is monochrome (green and white).
2. If the ALL operand is used, it must be the only operand specified on the only SCREEN statement for 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 user's directory 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's directory entry:

```
SCReen OutArea red blink InArea RevVideo yellow
```

## SHARE Directory Control Statement (General)



### Purpose

The SHARE statement specifies a virtual machine's share of CPU power.

### How to Specify

If you specify the SHARE statement, it must precede any device statements you specify in a user's directory entry. (For a list of device statements, see Table 20 on page 448.)

You can specify only one value (ABSOLUTE or RELATIVE) per statement. Multiple SHARE statements for a user entry are allowed but not recommended. If multiple SHARE statements are found in a user entry, only the last one is used.

Multiple SHARE statements are allowed within a profile, but are only used if no SHARE statements are in the user directory entry. If multiple SHARE statements are in an included profile, only the last one is used.

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 absolute share of the all 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 the all 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.

## SHARE

### 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.

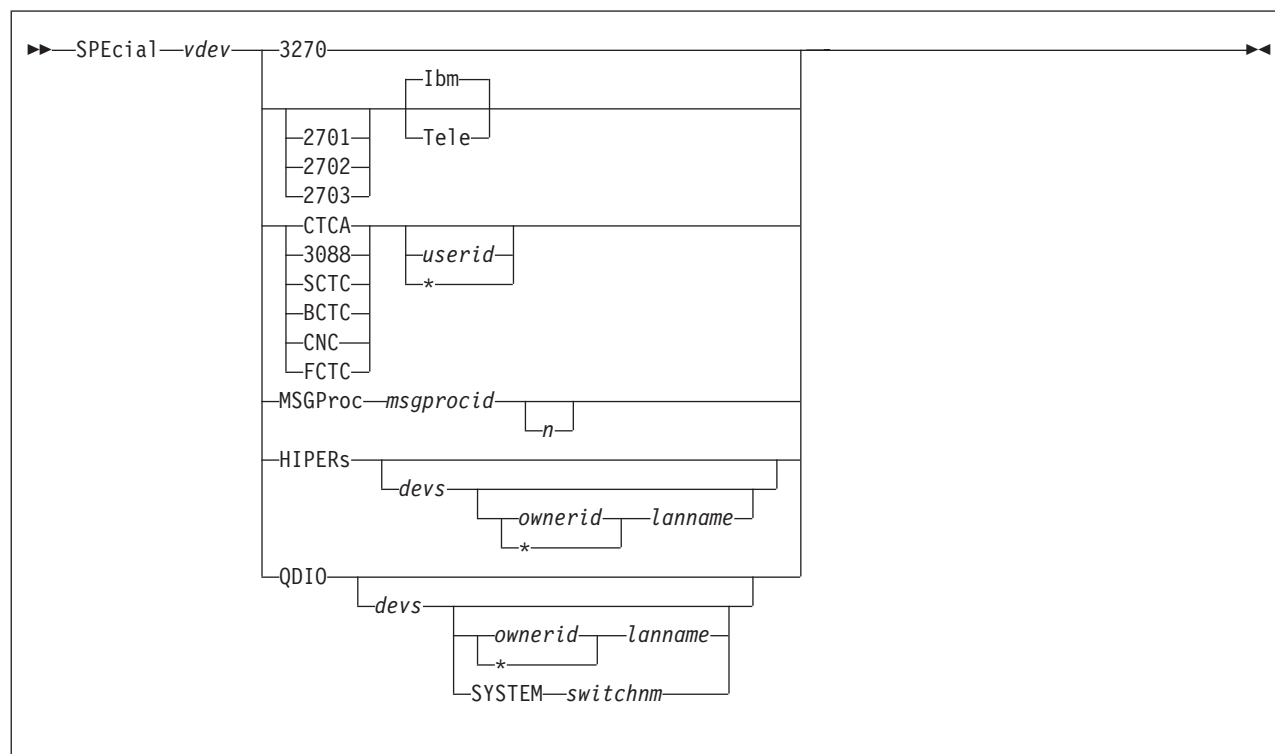
## Usage Notes

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 the *z/VM: Performance* book.
3. If a limit is specified without a maximum share being specified, the user's minimum share is also its maximum share.

## Examples

1. To assign a minimum absolute share of 50%, use the following SHARE statement in the virtual machine's directory entry:  
`Share Absolute 50%`
2. To assign a minimum relative share of 200, use the following SHARE statement in the virtual machine's directory entry:  
`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's directory entry:  
`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's directory entry:  
`Share RELATIVE 100 ABSOLUTE 75% LIMITHARD`

## SPECIAL Directory Control Statement (Device)



### 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 may 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. Refer to *z/VM: CP Commands and Utilities Reference* for details.

Use SPECIAL HIPER to create a virtual HiperSockets adapter in the virtual machine, and (optionally) connect it to a VM LAN segment. During LOGON processing, this statement is equivalent to a DEFINE NIC command followed by a COUPLE command.

### How to Specify

If you specify the SPECIAL statement, it must follow any general statements you specify in a user's directory entry. (For a list of general statements, see Table 20 on page 448.)

SPECIAL statements are allowed within a profile entry if no duplicate virtual device numbers are in the profile. Any virtual device number in a profile that duplicates a virtual device number in a user entry is resolved at logon time, with the user SPECIAL request presented to the system first.

### Operands

*vdev*

For most **SPECIAL** devices, this is the virtual device address of the device to

## SPECIAL

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.

### 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 may also specify:

#### *userid*

is the user ID of a virtual machine allowed to connect to this virtual CTCA, 3088, SCTC, BCTC, CNC or FCTC using the CP COUPLE command. For more information about the CP COUPLE command, see the *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.

### lbn

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 270x 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 user's directory entry, the user may only define message processors specified by SPECIAL MSGPROC statements.

The virtual message processor will only be defined if the following conditions exist:

- OPTION CFUSER is specified in the directory entry 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.

### 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 VM 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 VM 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 VM LAN segment 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 VM LAN. Note that *ownerid* may 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.

## SPECIAL

### 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 VM 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 VM 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 VM 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.

### 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 may be coupled by **LINKMAN**, specify one of the following **SPECIAL** statements in the virtual machine's directory entry:  

```
Special 2ff Ctca linkman  
Special 2ff 3088 linkman
```
4. To define a 3088 at virtual device number 3F0 that may be coupled to by any virtual machine, specify one of the following **SPECIAL** statements in the virtual machine's directory entry:  

```
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's directory entry:

```
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's directory entry:

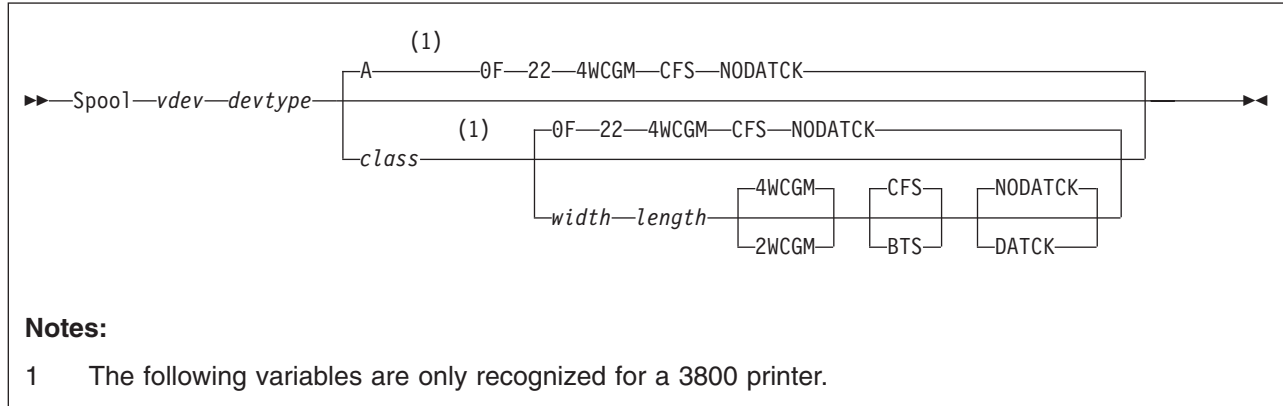
```
Special 301 2702 Ibm
Special 302 2702
Special 303 Ibm
```

**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's directory entry:

```
Special 500 hiper 8 system inews
```

## SPOOL Directory Control Statement (Device)



### Purpose

The SPOOL statement defines virtual unit record devices.

### How to Specify

If you specify the SPOOL statement, it must follow any general statements you specify in a user's directory entry. (For a list of general statements, see Table 20 on page 448.)

SPOOL statements are allowed within a profile entry if no duplicate virtual device numbers are in the profile. Any virtual device number in a profile that duplicates a virtual device number in a user entry is resolved at logon time, with the user SPOOL request presented to the system first.

When processing the SPOOL statement, 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 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	2540 <sup>Note 1</sup>	3800 <sup>Note 2</sup>	PCH <sup>Note 6</sup>
2501	3505	3800-1 <sup>Note 3</sup>	PUnch <sup>Note 6</sup>
3203	3525	3800-3 <sup>Note 4</sup>	RDR <sup>Note 7</sup>
3211	4245	Printer <sup>Note 5</sup>	Reader <sup>Note 7</sup>
3262	4248	PRT <sup>Note 5</sup>	VAFP <sup>Note 8</sup>

**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 default to a 1403 printer.
6. PCH or PUnch default to a 2540 punch.
7. RDR or Reader default to a 2540 reader.
8. Specify VAFP for a virtual advanced functional printer. For more information, see the DEFINE (Spooling Device) command in the *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.

## SPOOL

### 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's directory entry:

```
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's directory entry:

```
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's directory entry:

```
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's directory entry:

```
Spool 00e 3800 z 09 22 2wcm Cfs NoDatCk
```

## SPOOLFILE Directory Control Statement (General)

```
▶—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

Only one SPOOLFILE statement is allowed per user. If you specify the SPOOLFILE statement, it must precede any device statements specified in a user's directory entry and must follow the USER statement. (For a list of device statements, see Table 20 on page 448.) If the SPOOLFILE statement is omitted, the default maximum number of spool files is 9999.

One SPOOLFILE statement is allowed within a profile, but is used only if no SPOOLFILE statements are in the user directory 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. The actual maximum number of spool files as defined in a shared CSE directory is divided among the participating systems so that the total allowed on any one system may be far less than that specified by MAXSPOOL (see "Spool File Directory Statements" on page 341).

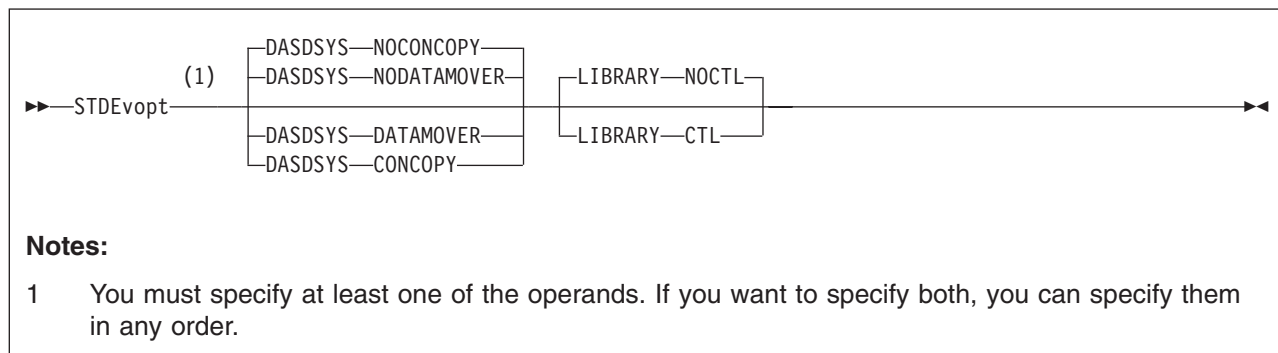
The reason for this is that MAXSPOOL actually defines the highest spool ID that a virtual machine can have.

### Examples

To indicate that the virtual machine has a limit of 1000 spool files, specify the following SPOOLFILE statement in the virtual machine's directory entry:

```
SpoolFile MaxSpool 1000
```

## STDEVOPT Directory Control Statement (General)



### Purpose

The STDEVOPT statement specifies the optional storage device management functions available to a virtual machine.

### How to Specify

If you specify the STDEVOPT statement, it must precede any device statements you specify in the user entry or profile. (For a list of device statements, see Table 20 on page 448.)

Multiple STDEVOPT statements are allowed within a profile entry. Conflicting operands 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 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 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's directory entry:
 

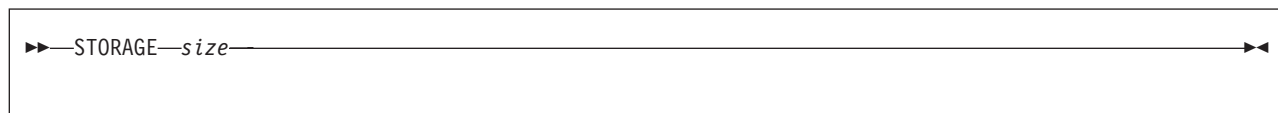
```
STDEvopt DASDSYS DATAMOVER
```
2. To specify that a virtual machine can control a 3494 or 3495 Tape Library Dataserver, specify the following STDEVOPT statement in the virtual machine's directory entry:
 

```
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's directory entry:
 

```
STDEvopt LIBRARY CTL DASDSYS DATAMOVER
```

---

## STORAGE Directory Control Statement (General)



### Purpose

The STORAGE statement specifies the default virtual storage size for a user.

### How to Specify

The STORAGE statement, if used, must follow a USER statement and must precede any device statements. A STORAGE statement is allowed in a profile if the USER statement default storage size field of each including virtual machine is either omitted or specified as an asterisk (\*). A STORAGE statement in a user directory entry overrides one in a profile.

### Operands

*size*

is the maximum storage size. specify it as  $nu$ , where  $n$  is a decimal number and  $u$  is the unit of measure:

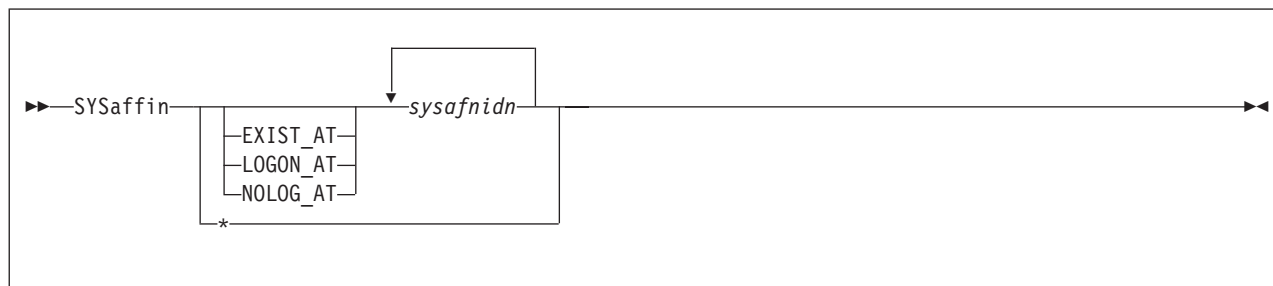
- K** kilobytes -  $2^{10}$
- M** megabytes -  $2^{20}$
- G** gigabytes -  $2^{30}$
- T** terabytes -  $2^{40}$
- P** petabytes -  $2^{50}$
- E** exabytes -  $2^{60}$

### Usage Notes

1. A default storage setting on a USER statement override a STORAGE statement in a profile
2. A STORAGE statement in a user's directory entry overrides one in a profile.



## SYSAFFIN Directory Control Statement (Control)



### 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.

### How to Specify

A SYSAFFIN statement has two forms, prefix and internal. The prefix form is placed before user definitions to describe where or how the user is to be applied to the systems of the complex. The internal form may 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 or profile definition.

Each SYSAFFIN prefix control statement applies only to the USER statement that immediately follows it.

A SYSAFFIN internal control statement applies to the statements that immediately follow it. It is active until the next SYSAFFIN prefix or internal control 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 control statements to which it applies, affect the system identified by *sysafnid*.

### Operands

#### **EXIST\_AT** *sysafnid*

specifies that the following user definition 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 (*sysafnid*) listed on the SYSAFFIN statement. When DIRECTXA is running on any other system, this user definition is not to be compiled into the object directory. This means that any resources owned by this user definition (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** *sysafnid*

specifies that the user directory entry 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

## SYSAFFIN

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 user directory entry 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 directory entries 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 control statements apply to. Valid *sysafnidns* are defined on the DIRECTORY Control Statement. \* indicates that the subsequent control statements apply to all systems.

## Usage Notes

1. The default system affinity for user definitions not otherwise prefixed by a SYSAFFIN control statement is LOGON\_AT \*. This means that the user definition is applied to and that the defined user can log on at all systems in the complex. The default system affinity for all other control statements is \*, meaning that all subsequent control 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. EXIST\_AT should not be used for systems sharing spool files in a CSE complex.
5. The internal form of the SYSAFFIN control statement may be used in user definitions, profile definitions, and the global definition section. 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 3380 vmaipl *01234-2064 vma
Directory 0b64 3380 vmbipl 104567-2064 vmb
Directory 0cF4 3380 vmcipl *15211-2064 vmc
Directory 0664 3380 vmdipl *00012-2064 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 ...

```

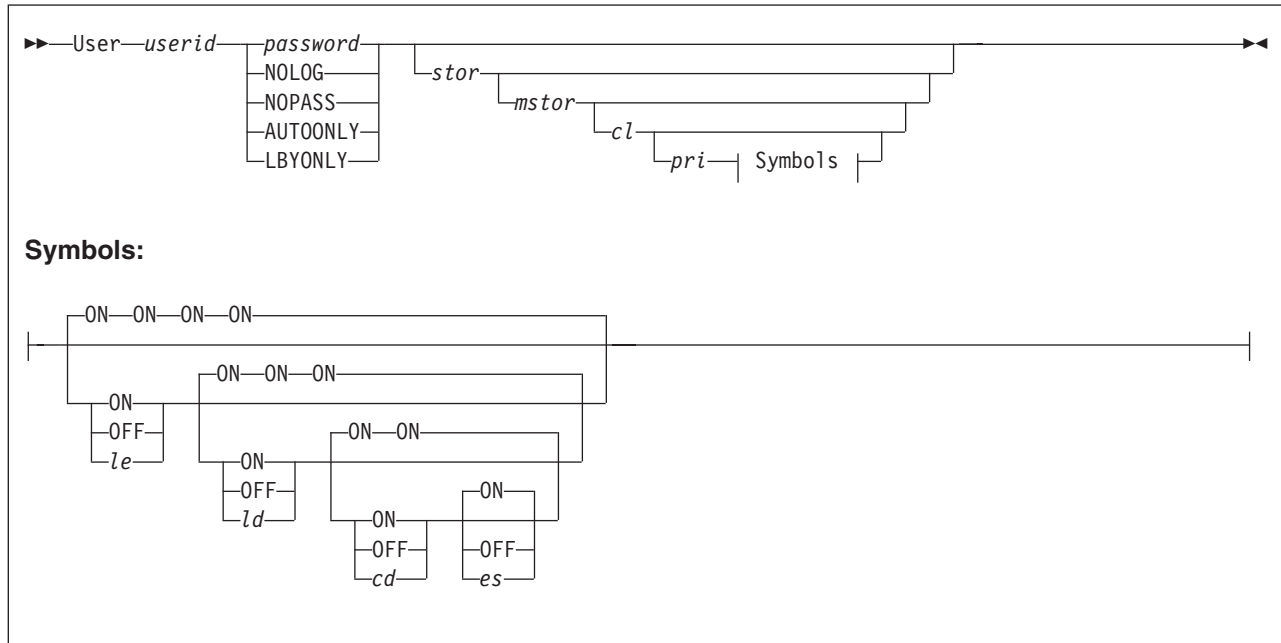
3. The following global definition section would define a default machine mode of XA on system vma and a default machine mode of ESA on system vmb.

```

Globaldefs
Sysaffin vma
Globalopts machine xa
Sysaffin vmb
Globalopts machine esa

```

## USER Directory Control Statement (Control)



### Purpose

The USER statement starts each virtual machine's directory entry. It 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

USER statements are not allowed within a profile entry.

When processing the USER statement, DIRECTXA checks for a maximum of ten tokens: userid, password, logon storage size, maximum storage size, privilege class, a place holder 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. LOGNxxxx, LOGLxxxx, LOGVxxxx, LOGNSYSC, LOGNSYSG, and SYSTEM are reserved for CP use.

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 *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 may 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.

The characters that you can use for user IDs are restricted by translation test tables in HCPTBL to:

**alphabetics**

A through Z

**numerics**

0 through 9

**others**

@ # \$ \_ (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. Refer to Appendix B in the *z/VM: CP Commands and Utilities Reference* for a discussion of the system defaults settings for logical line editing symbols

You can override the translation tables in HCPTBL. See the "TRANSLATE\_TABLE Statement" on page 243.

*password*

specifies a 1- to 8-character password that a user enters during the log on procedure.

**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. Refer to Appendix B in the *z/VM: CP Commands and Utilities Reference* for a discussion of the system defaults settings for logical line editing symbols

**NOPASS**

specifies that a user does not require a password to log on. It also specifies that a virtual machine may 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 may 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 may not be able to perform functions that require password validation, since 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.

*stor*

specifies the virtual machine's primary address space size at log on. Specify *stor* as *nu* where *n* is a one- to seven-digit decimal number and *u* is the unit of measure (see Table 25). The maximum value you can specify is 16E or 16384P, though the actual maximum size supported may be restricted by the model of the processor where the directory is used.

The storage field can also contain an asterisk (\*) as a placeholder. If it is omitted or specified as an asterisk, this indicates that the storage size is defined by a MAXSTORAGE statement in either the user's directory entry or a directory profile.

Values below 16 MB are rounded up to 64 KB boundaries. Values above 16 MB are rounded up to 1 MB boundaries.

Table 25. Storage Units of Measure

Units suffix	Maximum allowed as input
K - Kilobytes	2096128
M - Megabytes	9999999
G - Gigabytes	9999999
T - Terabytes	9999999
P - Petabytes	16384
E - Exabytes	16

**Note:** Allowing large numbers of V=V virtual machines to have large primary address space sizes may affect real storage availability. See usage note 1 on page 576.

*mstor*

specifies a maximum virtual machine primary address space size that a user can specify with the DEFINE STORAGE command. Specify *mstor* as *nu*, where *n* is a one- to seven-digit decimal number and *u* is the unit of measure (see Table 25). The default is 1M (1 MB).

The storage field can also contain an asterisk (\*) as a placeholder. If it is omitted or specified as an asterisk, this indicates that the maximum storage size is defined by a MAXSTORAGE statement in either the user's directory entry or a directory profile.

**Note:** Allowing large numbers of V=V virtual machines to have large primary address space sizes may affect real storage availability. See usage note 1 on page 576.

*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 may specify up to 32 classes (if they fit). Each character represents a single privilege class and may appear in any order, without duplication, and must not be separated by blanks. The class field can also contain an asterisk (\*) as a place holder. 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 (page 184) or by the SYSFCN macro in HCPSYS (page 673).

**Note:** If you have not used the user class restructure function to change your classes from the IBM-defined defaults, only classes A through G are valid. Refer to Chapter 16, “Redefining Command Privilege Classes,” on page 433 for information on defining your own user class structure. For a description of the different command classes, see the *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 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 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 about setting system-wide defaults, see 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 about setting system-wide defaults, see 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 about setting system-wide defaults, see 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,

## USER

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 about setting system-wide defaults, page 67.

## Usage Notes

1. For each virtual machine, CP creates a segment table to represent the virtual machine storage. For a virtual machine of 32 MB or less, CP creates the segment table inside the virtual machine definition block (VMDBK). For a larger virtual machine, CP creates the segment table outside the VMDBK in real storage, as follows:
  - For a virtual machine larger than 32 MB but less than or equal to 1024 MB, one real storage frame is allocated for the segment table.
  - For a virtual machine larger than 1024 MB, two consecutive real storage frames are allocated for the segment table.

CP creates the segment table at the start of the real storage frame, and any storage remaining beyond the end of the segment table may be used for CP free storage. However, if a large number of virtual machines are defined arbitrarily large, considerable real storage may be consumed by the segment tables.

2. The *cl* field may contain an asterisk (\*) as a place holder. 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 (page 184) or by the SYSFCN macro in HCPSYS (page 673).
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 described in the *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 (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 commandsspecify 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

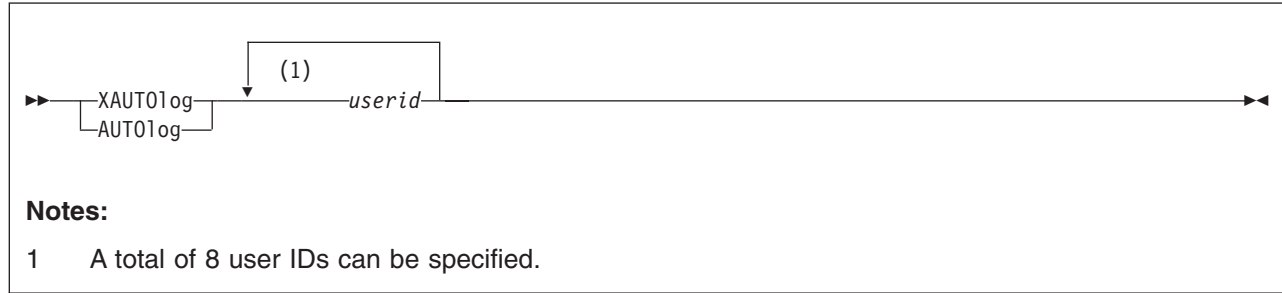
specify the following USER statement:

```
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
```

## XAUTOLOG Directory Control Statement (General)



### 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

If you specify the XAUTOLOG statement, it must precede any device statements you specify in a user's directory entry. (For a list of device statements, see Table 20 on page 448.) A directory entry may 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, but are used only if no XAUTOLOG (or AUTOLOG) statements are in the including user directory 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 directory entry 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's directory entry:

```
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's directory entry:

```
XautoLog oper1 oper2 oper3 jones
```

---

# XCONFIG Directory Control Statement (General)

## 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 definition, and on, at most, one XCONFIG statement within a PROFILE definition.

If a major option appears on an XCONFIG statement in both the USER definition and in an included PROFILE definition, then any operands specified in the USER definition override those specified (or defaulted) in the PROFILE definition. Operands that are specified (or defaulted) in the PROFILE definition but not also specified in the USER definition are used as specified (or defaulted) in the PROFILE definition.

## How to Specify

If you specify any XCONFIG statements, they must precede any device statements, both when specified within a USER definition and when specified within a PROFILE definition. (For a list of device statements, see Table 20 on page 448.)

## 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 user's CP directory entry, 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, or ESA 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 about the ALSERV macroinstruction, see the *z/VM: CP Programming Services* book.

### How to Specify

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 (CP) real storage. When a host access list is initially created for a virtual machine, sufficient host real storage for a 6-entry host access list is allocated for the 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 1 page of host real storage and a 1022-entry host access list can require up to 4 pages of host real storage.
3. The host real 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.

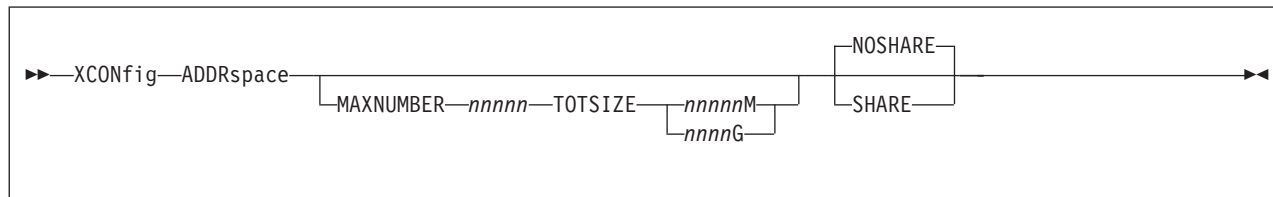
### Examples

To specify that a virtual machine is to have a 126-entry host access list, use the following statement in the user's directory entry:

## XCONFIG

Xconfig AccessList ALsize 126

## XCONFIG ADDRSPACE Operand



### Purpose

The XCONFIG ADDRSPACE operand specifies the maximum number of nonprimary address spaces (sometimes known as data spaces) and the total size in bytes of all nonprimary address spaces that the virtual machine can own simultaneously. These address spaces are created using the ADDRSPACE 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 ADDRSPACE macroinstruction. ESA, XA, and XC virtual machines can allow other virtual machines to share their address spaces by using the PERMIT function of the ADDRSPACE macroinstruction. (This is limited to sharing the virtual machine's primary address space.) For more information about the ADDRSPACE macroinstruction, see the *z/VM: CP Programming Services* book.

### How to Specify

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 nonprimary 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 *nnnnnM*

#### TOTSIZE *nnnnG*

specifies the maximum total size, in bytes, of all address spaces that this virtual machine can create and have existing concurrently. The variable *nnnnnM* is a decimal number from 1 megabyte to 99999 megabytes. The variable *nnnnG* 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 8192B.

#### NOSHARE

tells CP that the virtual machine cannot use the PERMIT function of the ADDRSPACE macroinstruction to make its address spaces available for access by other virtual machines. The default is NOSHARE.

## XCONFIG

### 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 large numbers of nonprimary address spaces and/or a large total address space size may affect real storage availability. For each address space, CP creates a segment table in real storage, as follows:
  - For each address space less than or equal to 1024 MB, one real storage frame is allocated for the segment table.
  - For each address space larger than 1024 MB, two consecutive real storage frames are allocated for the segment table.

CP creates the segment table at the start of the real storage frame, and any storage remaining beyond the end of the segment table may be used for CP free storage. However, if users are permitted to create large numbers of address spaces and/or very large size address spaces, considerable real storage may be consumed by the segment tables.

## 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 user's directory entry:

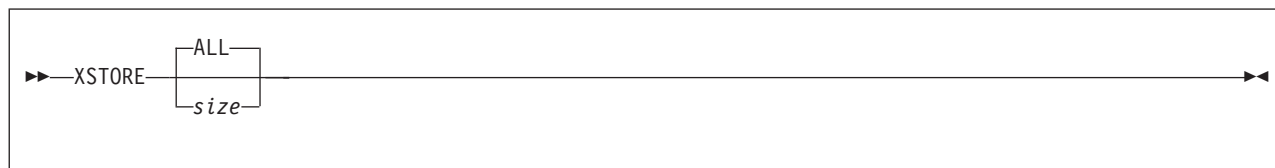
```
Xconfig AddrSpace MaxNumber 128 TotSize 1000M
```
2. To authorize a virtual machine to create up to 25 address spaces, with a total of 2 GB of storage, and allow the virtual machine to share its primary address space and any of its created address spaces with other virtual machines, use the following statement in the user's directory entry:

```
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 user's directory entry:

```
Xconfig AddrSpace Share
```



## XSTORE Directory Control Statement (General)



### Purpose

The XSTORE statement authorizes a virtual machine that supports Expanded Storage to use the Expanded Storage facility.

### How to Specify

If you specify the XSTORE statement, it must precede any device statements you specify in a user's directory entry. (For a list of device statements, see Table 20 on page 448.) Refer to Chapter 19, "Expanded Storage Planning and Administration," on page 591 for additional information about allocating Expanded Storage.

Multiple XSTORE statements are allowed within a profile but are used only if no XSTORE statements are in the user directory entry. If multiple XSTORE statements are in an included profile entry, only the first one is used.

When processing the XSTORE statement, DIRECTXA checks for a maximum of one token: the real Expanded Storage size. If you specify more than one token, DIRECTXA ignores the extra tokens and system processing continues as if you had not specified the extra tokens.

### Operands

#### ALL

tells CP that all of Expanded Storage configured on the system be allowed to be used for this user's guest partition. If you specify XSTORE without any operands, the default is ALL.

#### size

is the size expressed as *nnnnnnnn*M megabytes of real Expanded Storage to be allocated to the user's virtual machine on megabyte boundaries. The maximum size that can be specified is 16777216 MB.

The real Expanded Storage may be allocated to any number of virtual machines at the same time.

User and operator commands are available to manage the allocation of Expanded Storage after user logon (see the *z/VM: CP Commands and Utilities Reference*).

### Usage Notes

1. Several commands allow you to attach, detach, and display information about the Expanded Storage facility. These commands include ATTACH XSTORE, DETACH XSTORE, and QUERY XSTORE. For more information on these commands, see the *z/VM: CP Commands and Utilities Reference*.
2. If the request for Expanded Storage cannot be satisfied at logon time, the maximum amount that can be attached at this time is attached to the virtual machine.

## XSTORE

3. Multiple users may have Expanded Storage attached to their virtual machines.

## Examples

To authorize a virtual machine to use all of Expanded Storage, specify the following XSTORE statement in the virtual machine's directory entry:

```
Xstore
```

---

## Part 4. Storage Planning and Administration

<b>Chapter 18. Real Storage Planning and Administration</b> . . . . .	589
Planning Real Storage Needs . . . . .	589
Real and Virtual Storage Requirements for Generating a CP Module . . . . .	589
Determining the Size of the Dynamic Paging Area . . . . .	590
Virtual Machine Considerations . . . . .	590
<b>Chapter 19. Expanded Storage Planning and Administration</b> . . . . .	591
Overview . . . . .	591
Expanded Storage Considerations with Shared CMS Minidisks . . . . .	592
Attaching Expanded Storage to a Virtual Machine . . . . .	592
Mapping Expanded Storage . . . . .	593
Fragmentation . . . . .	593
Discontinuous Expanded Storage . . . . .	593
Allocating Expanded Storage to CP . . . . .	596
Allocating Retained Expanded Storage . . . . .	596
Pending Retained Expanded Storage . . . . .	597
Using Expanded Storage as a Minidisk Cache . . . . .	597
What Devices Can Be Cached . . . . .	600
Turning Caching Off . . . . .	600
Planning for and Using Minidisk Caching . . . . .	600
Special Considerations for CMS Minidisk Volumes Shared between Processors . . . . .	601
Performance Considerations . . . . .	602
<b>Chapter 20. Allocating DASD Space</b> . . . . .	603
Direct Access Storage Requirements . . . . .	603
CKD Device Geometry . . . . .	603
FBA Device Geometry . . . . .	604
Storage Calculations . . . . .	604
CP Module . . . . .	605
Warm Start Data . . . . .	606
Example . . . . .	607
Checkpoint Data . . . . .	607
Directory Space . . . . .	608
Example . . . . .	609
Directory Size Constraints . . . . .	610
Paging Space . . . . .	610
Example . . . . .	611
Spooling Space . . . . .	613
Allocating Space for CP Abend Dumps . . . . .	614
Named Saved System . . . . .	615
Spool Space Example . . . . .	615
Switching Operating Modes for 3390 Devices . . . . .	615
Mode Switch Procedure . . . . .	616
Migration Considerations for the 3390 Model 9 . . . . .	617
<b>Chapter 21. DASD Sharing</b> . . . . .	619
Sharing DASD among Multiple Virtual Machines by Using Virtual Reserve/Release . . . . .	619
When to Use Virtual Reserve/Release . . . . .	621
Sharing DASD without Using Virtual Reserve/Release . . . . .	622
Sharing DASD Using the CMS Shared File System . . . . .	622

Sharing DASD between One Virtual Machine and Other Systems Using Real Reserve/Release . . . . .	622
When to Use Real Reserve/Release . . . . .	624
Sharing DASD among Multiple Virtual Machines and Other Systems Using Concurrent Virtual and Real Reserve/Release . . . . .	624
When to Use Concurrent Virtual and Real Reserve/Release . . . . .	626
Restrictions for Using Reserve/Release . . . . .	626
Reserve/Release Summary . . . . .	626
Sharing DASD using the Multi-Path Lock Facility . . . . .	627
Cached DASD . . . . .	628
Cache Control. . . . .	628
Cache Control at the Subsystem Level . . . . .	629
Cache Control at the Device Level . . . . .	629
Cache Control at the Extent Level . . . . .	629
Defining a Minidisk on a Cached DASD . . . . .	629
Defining a Cached DASD as a Dedicated Device . . . . .	630
Using ESS Parallel Access Volumes . . . . .	631
Using PAV for Workload Balancing . . . . .	631
Using PAV for DASD Sharing . . . . .	632
z/VM Restrictions on Using PAV . . . . .	633
<b>Chapter 22. Defining and Managing SCSI FCP Disks . . . . .</b>	<b>635</b>
Overview . . . . .	635
Defining SCSI Devices . . . . .	637
Emulated FBA Disks on SCSI Disks . . . . .	637
Real SCSI Disks . . . . .	638
Additional Considerations . . . . .	639

---

## Chapter 18. Real Storage Planning and Administration

This chapter describes:

- The real storage requirements for CP
- How to define a dynamic paging area

---

### Planning Real Storage Needs

The amount of real storage you need depends upon the type of work your installation will be performing. A system might include many virtual machines. All of these virtual machines require 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 of the following:

- Add more real storage
- Run fewer virtual machines.

If you are using the system configuration file to define your system, you do not need to define the amount of real storage available to CP. CP assumes that it has all of the storage on the machine available to it. If this assumption is not true, you must specify the amount of real storage available to CP by specifying STORE as an IPL parameter passed by the Stand-Alone Program Loader (SAPL) to CP.

**Note:** The minimum amount of real storage required for operation of z/VM is 10 MB.

Your installation can, within limits, change the size of every area of storage except the CP module, which occupies approximately 10 MB. Use the CP command QUERY FRAMES to display the status of real storage and to determine how real storage is currently allocated. The *z/VM: CP Commands and Utilities Reference* book contains a description of the QUERY FRAMES command.

---

### Real and Virtual Storage Requirements for Generating a CP Module

The CP module requires approximately 10 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 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.

### Determining the Size of the Dynamic Paging Area

The dynamic paging area (DPA) holds the virtual storage of virtual machines and the storage CP needs for pageable modules, spool buffers, trace tables, virtual disks in storage, and other CP areas. The storage remaining after you allocate all other areas of storage is used for the DPA. When determining how large a DPA you need, you should consider your installation's paging capability. If your system has enough high performance DASDs or enough cache control units of a size for your system's paging demands, you may be able to generate a relatively small DPA and still achieve good performance. If, however, your installation has mostly lower performance DASDs you may need a larger DPA.

---

### Virtual Machine Considerations

Allowing large numbers of virtual machines to have large storage sizes (primary address spaces and/or data spaces) may affect real storage availability:

- For each primary address space larger than 32 MB but less than or equal to 1024 MB, one real storage frame is allocated for the segment table.
- For each nonprimary address space (data space) less than or equal to 1024 MB, one real storage frame is allocated for the segment table.
- For each primary or nonprimary address space larger than 1024 MB, two consecutive real storage frames are allocated for the segment table.

Although CP creates the segment table at the start of the real storage frame, and any storage remaining beyond the end of the segment table may be used for CP free storage, you should consider the combined effect of giving many users large storage sizes and/or allowing users to create many data spaces and/or large data spaces. Considerable real storage may be consumed by the segment tables.

---

## Chapter 19. Expanded Storage Planning and Administration

The Expanded Storage facility is an optional feature of certain processor models. It can be:

- Dedicated to CP
- Dedicated to one or more virtual machines for use by an operating system
- Partitioned for use between CP and one or more virtual machines.

This chapter tells you what Expanded Storage is and how to:

- Attach Expanded Storage to a virtual machine
- Map Expanded Storage
- Handle fragmentation of Expanded Storage
- Allocate Expanded Storage to CP
- Allocate retained Expanded Storage
- Use Expanded Storage as a minidisk cache.

---

### Overview

Expanded Storage acts as another level of real storage. It allows for the rapid transfer of 4 KB pages between itself and real storage. As real storage is divided into 4 KB page frames, Expanded Storage is divided into 4 KB units called blocks. It is available in increments whose size varies with the model of the processor.

You can partition Expanded Storage. Part of it may be attached to one or more virtual machines. Another part, the CP partition, is used for system paging and the caching of certain kinds of data. For paging, CP uses blocks available in the paging portion of the CP partition in preference to DASD slots.

To retain some or all of Expanded Storage exclusively for CP use, the class B operator can enter the RETAIN XSTORE command. The command assigns the specified amount of storage to the CP partition. The CP partition size limits the amount of Expanded Storage available for virtual machine partitions.

All or part of Expanded Storage can be retained for CP's exclusive use. This CP partition can occupy multiple discontinuous areas of Expanded Storage. All or part of the remainder can be dedicated to virtual machines.

If there is a request for additional Expanded Storage to be retained by CP and not enough contiguous Expanded Storage is available, the request can be satisfied by retaining multiple discontinuous areas of Expanded Storage. If a request to retain Expanded Storage for CP use is for a larger amount than is available because it is in use by one or more virtual machines, CP retains the amount of Expanded Storage currently available. It also issues a pending request for the remainder. The pending request is satisfied when enough Expanded Storage is detached from one or more virtual machines.

z/VM treats Expanded Storage as another level of real storage. All paging goes to Expanded Storage if possible. Old pages from Expanded Storage are migrated to DASD when paging use exceeds a high threshold. For the most part, direct paging to DASD occurs only if Expanded Storage is totally filled and migration cannot make room in Expanded Storage quickly enough to keep up with the paging demand.

## Expanded Storage Planning and Administration

Basically, you can configure Expanded Storage in any manner that meets your installation's needs. The granularity is in 1 MB sections, and you can attach it to as many virtual machine guests as you have megabytes of Expanded Storage.

In addition, CP uses Expanded Storage as a minidisk cache that shares Expanded Storage with paging.

---

### Expanded Storage Considerations with Shared CMS Minidisks

If you have DASD volumes that contain CMS minidisks that are shared among different processors, you must take special action before IPLing z/VM. For each DASD volume that contains CMS minidisks that are physically shared among processors, 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 definition prevents all minidisks on these 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 if any system other than the one that is caching data through the minidisk cache has write access to the volume.

For more information on minidisk caching, see "Using Expanded Storage as a Minidisk Cache" on page 597. For information on the RDEVICE system configuration file, see "RDEVICE Statement (DASD)" on page 195. For information on the SET SHARED and SET RDEVICE commands, see the *z/VM: CP Commands and Utilities Reference*. For more information on the DASD RDEVICE macroinstruction, see "DASD Types" on page 746.

---

### Attaching Expanded Storage to a Virtual Machine

You can use either the XSTORE user directory control statement or the ATTACH XSTORE command to attach Expanded Storage to a virtual machine for its direct use. The difference is that if you use the XSTORE user directory control statement, the area of Expanded Storage specified is automatically attached to the virtual machine at logon. With the ATTACH XSTORE command, the area is attached when the command is entered. For more information on the XSTORE user directory control statement, see "XSTORE Directory Control Statement (General)" on page 585.

When CP attaches Expanded Storage to a virtual machine, it looks for contiguous Expanded Storage that is not already attached to another virtual machine or assigned as CP-retained Expanded Storage. This search starts from Expanded Storage block 0.

The amount and location of the Expanded Storage that is attached depends on how Expanded Storage is currently allocated. The response from the ATTACH command indicates the amount that is actually attached. The QUERY XSTORE MAP command can be used to determine its location.

The term contiguous is emphasized because certain types of fragmentation can occur that permit CP to attach Expanded Storage that is larger than what would be truly contiguous. See "Fragmentation" on page 593 for more information about fragmentation.



---

## Mapping Expanded Storage

You should develop a map of Expanded Storage and determine which virtual machines should have access to this valuable resource. Any Expanded Storage that you attach for CP's exclusive use (retained) should be at the top of the map (highest block number), and any Expanded Storage attached to virtual machines should be drawn from the lowest block numbers. This ordering for the map matches the most likely format produced as the output of the `QUERY XSTORE MAP` command.

With this map, you can check that Expanded Storage requirements are not over-committed. You can use the class B `QUERY XSTORE MAP` command to determine how CP has actually attached Expanded Storage. This should match your map for this configuration.

Since the amount of Expanded Storage varies from one configuration to another, this map only serves one particular configuration. One map for each configuration may be useful. See "Attaching Expanded Storage to a Virtual Machine" on page 592 for a description of how CP attaches Expanded Storage as users log on with XSTORE user directory control statements or enter `ATTACH XSTORE` commands.

## Fragmentation

Two types of fragmentation can occur when Expanded Storage is attached to CP and virtual machines:

- `QUERY XSTORE MAP` shows a section of Expanded Storage as N/A (not applicable). This is caused by the hardware configuration. For example, the second and fourth sections of Expanded Storage could be partitioned to the other side of a multiprocessor.

This Expanded Storage is not available for use by CP or any virtual machine. When CP attaches Expanded Storage for a virtual machine, it can cross these areas to attach the amount requested. The amount of N/A Expanded Storage is not counted toward satisfying the request.

- The portion of Expanded Storage is attached to a virtual machine or is retained by CP.

This is true fragmentation. CP cannot attach Expanded Storage to a virtual machine that crosses an area already attached to a virtual machine or retained for CP.

## Discontinuous Expanded Storage

Generally, Expanded Storage is not expected to become fragmented. It is assigned when the system operator first IPLs the system and logs on critical virtual machines. In general, this situation remains static.

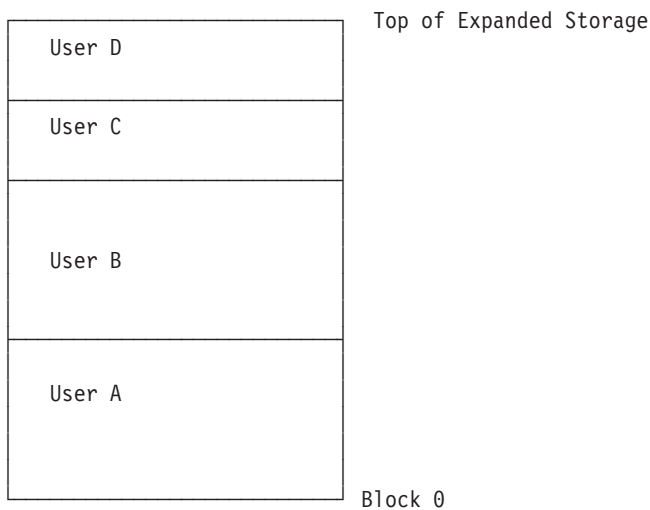
However, if Expanded Storage is attached to virtual machines, detached, and attached again with a different size, fragmentation can occur that the system operator may want to correct. By using the `QUERY XSTORE MAP` command, the system operator can keep track of how Expanded Storage is attached and determine whether fragmentation exists.

If fragmentation does occur, the operator can use the `RETAIN XSTORE`, `ATTACH XSTORE`, and `DETACH XSTORE` commands to rearrange the attachment of Expanded Storage. By doing this carefully, fragmentation can be reduced or eliminated.

## Expanded Storage Planning and Administration

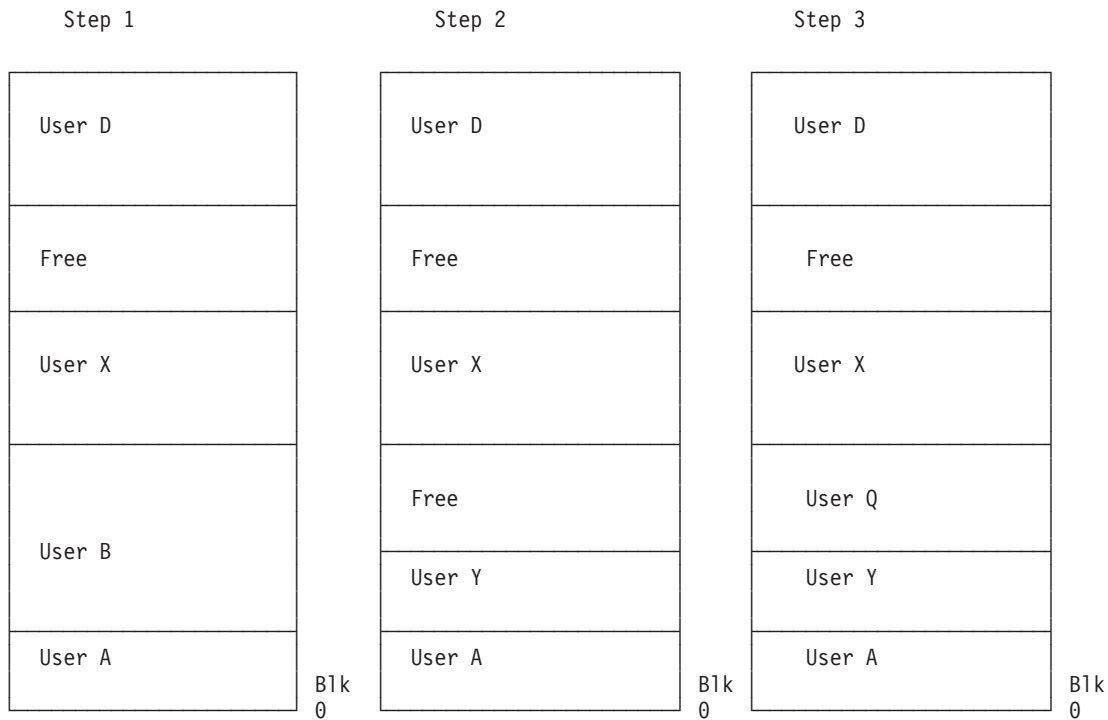
**Note:** The system operator must be certain that all virtual machines that have Expanded Storage attached and that will be manipulated by this rearrangement are not running a guest that would be affected by the detaching of Expanded Storage. In general, you should do this when the guest is no longer using Expanded Storage, the guest operating system is reset, or the logged-on Expanded Storage is released.

The example shown in Figure 30, Figure 31 on page 595, and Figure 32 on page 596 shows how a normal attachment can become fragmented and then rearranged. In all cases where a virtual machine owns Expanded Storage, no guest is running in that virtual machine at the time a detach is done.



*Figure 30. Map of Expanded Storage Allocation—No Fragmentation*

If User C logs off and User X logs on with a request for a smaller amount of Expanded Storage, a small amount of unattached Expanded Storage exists (see Figure 31 on page 595, step 1). This by itself is not fragmentation. However, if User B logs off and User Y logs on with a request for less than User C had but larger than the previously unattached amount, fragmentation occurs (see Figure 31 on page 595, step 2). The two discontinuous fragments cannot be joined. If User Q logs on and requests an amount equal to the sum of the two pieces, User Q's virtual machine only gets the amount associated with the larger piece (see Figure 31 on page 595, step 3). QUERY XSTORE or QUERY XSTORE USER also shows this fragmentation, since the last response has the total amount that can be attached that is greater than the maximum that can be attached to a single virtual machine.



*Figure 31. Map of Expanded Storage Allocation—with Fragmentation*

By using several commands, this situation can be corrected. First, detach the Expanded Storage associated with User Q. This is the virtual machine that received the wrong amount of Expanded Storage (see Figure 32 on page 596, step 1). Then, detach the Expanded Storage associated with User X. This is between the two free areas (see Figure 32 on page 596, step 2). This creates a large contiguous portion of Expanded Storage. Now, attach Expanded Storage to User Q and User X with their original amounts. The fragmentation is now removed (see Figure 32 on page 596, step 3).

## Expanded Storage Planning and Administration

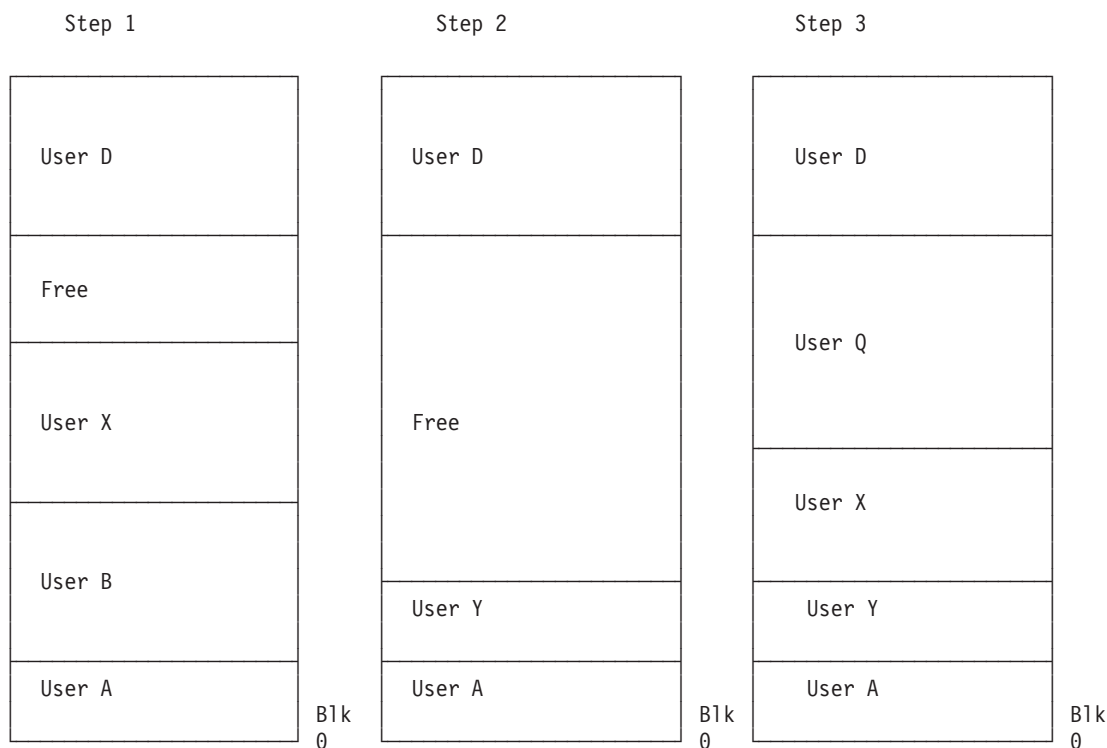


Figure 32. Map of Expanded Storage Allocation — Fragmentation Removed

This example is a fairly simple one. The fragmentation might be worse. It could also contain some sections that are marked CP(RETAINED). In the cases where CP(RETAINED) exists, you can enter `RETAIN XSTORE NONE` to start the process. After you have corrected the Expanded Storage fragmentation, use the `RETAIN XSTORE` command to reinstate the amount to be retained for CP.

---

### Allocating Expanded Storage to CP

System initialization attaches Expanded Storage to CP by default. Use the `RETAIN XSTORE` command to set aside Expanded Storage for the exclusive use of CP. Retained Expanded Storage is not available to be attached to virtual machines.

**Note:** The `RETAIN XSTORE` command is used to prevent a guest with an `XSTORE` user directory control statement in its directory entry from logging on and taking enough Expanded Storage from the system to cause performance problems. It controls inadvertent operational errors.

### Allocating Retained Expanded Storage

When the `RETAIN XSTORE` command is entered, CP sets aside Expanded Storage for CP's exclusive use starting with the highest block number. This corresponds to the top of the map discussed under "Mapping Expanded Storage" on page 593.

If the amount to be retained cannot be found at the highest block numbers, CP takes any Expanded Storage not assigned to virtual machines. This action could cause fragmentation. To minimize fragmentation, the system operator should enter `RETAIN XSTORE` immediately after the system initializes. If the map for the

configuration is followed, the presence of virtual machines logging on at the time should not cause a problem. Expanded Storage is attached to the virtual machines starting from the lowest block number.

### Pending Retained Expanded Storage

If not enough Expanded Storage is available to satisfy a RETAIN XSTORE command, the remaining amount is designated as pending retained Expanded Storage. When a virtual machine detaches Expanded Storage, CP uses some or all of the amount detached to fulfill the pending retain. If the operator wants to delete pending retain, the RETAIN XSTORE command should be entered again with the amount given on the command equal to the current amount retained.

**Note:** Since virtual machines can detach Expanded Storage in an unpredictable way, pending retain can also cause fragmentation.

---

### Using Expanded Storage as a Minidisk Cache

Expanded Storage can also 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 Expanded 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.

Real storage can also be used for minidisk caching.

Using Expanded Storage as a minidisk cache may also 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.

Expanded storage is faster than DASD control unit caches. Control unit caches transfer at channel speeds; Expanded Storage transfers at near central storage speeds.

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

## Expanded Storage Planning and Administration

- Overlaps another minidisk which may have data in the cache
- DASD is detached from the system
- The Expanded Storage partition that contains minidisk caching blocks is attached to a guest virtual machine.

Unnecessary cache invalidation decreases performance by reducing the cache hit ratio. To avoid problems, see “Planning for and Using Minidisk Caching” on page 600.

### ***CP Minidisk Cache Enhancements:***

With the enhancements to the CP minidisk cache feature in Version 5 Release 1 the following are potential items to consider when migrating from previous releases of z/VM or VM/XA SP.

- Reconfigure expanded storage as main storage if it is specifically for minidisk cache.

Some processors allow a portion of main storage to be configured as expanded storage. Installations that configure storage as expanded storage in order to use minidisk cache should consider reconfiguring this expanded storage as main storage.

- Review storage allocation for minidisk cache.

Prior to VM/ESA 1.2.2, the amount of expanded storage allocated for minidisk cache could be fixed or bounded (minimum and maximum) with the RETAIN XSTORE MDC command. Now the amount of storage, either real or expanded, can be fixed or bounded with the SET MDCACHE command. The RETAIN XSTORE MDC command is maintained for compatibility purposes. 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.

The arbiter was completely changed in VM/ESA 1.2.2. This eliminated some limitations of the previous arbiter. If the arbiter was ineffective for your system in the past, it may now be acceptable and should be tried. 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 both real and expanded storage usage.

A new method of influencing the arbiter is setting a bias with the SET MDCACHE STORAGE BIAS or SET MDCACHE XSTORE BIAS commands. 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 both MDC in real storage and MDC in expanded storage, 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. Likewise, the expanded storage arbiter usually makes a good trade-off between using expanded storage for MDC and using it for 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.

- When using expanded storage for minidisk cache, also use some real storage for minidisk cache.

In situations where expanded storage is to be used for most minidisk caching, it is better to allow some real storage to also be used for minidisk caching rather than setting the real MDC size to 0 or setting it off.

## Expanded Storage Planning and Administration

- Use SET MDCACHE or SET RDEVICE commands instead of SET SHARED to enable minidisk cache on volumes.

Prior to z/VM 4.4.0, you could 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 SHARE OFF command. Now 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.

Because many of the restrictions formerly associated with minidisk caching have been removed, it can now 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.

- Reformat some minidisks to smaller blocksize.

Each CMS file consists of at least one DASD record even if the file is very small. The capacity of 4 KB formatted minidisks that consist of mostly small files may be increased by reformatting them using 2 KB, 1 KB or 512 byte sizes. Prior to VM/ESA 1.2.2, reformatting to a smaller size would have made the minidisk ineligible for minidisk caching. Care must be taken, however, because as the record size gets smaller, the number of available bytes per track is reduced.

- Prepare for minidisk caching on devices shared between first and second level systems.

Prior to z/VM 4.4.0, most second level systems did not use minidisk cache because it required expanded storage, which most second level test systems do not have. Now that minidisk cache can use real storage instead of expanded storage, second level systems benefit. 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

## Expanded Storage Planning and Administration

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:

- DEVNO defined full-pack minidisks
- FBA minidisks that are not page-aligned
- Virtual disks in storage
- Dedicated or attached DASD
- Shared 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 user directory control statement in the CP directory. For more information, see “MINIOPT Directory Control Statement (Device)” on page 524.

## Planning for and Using Minidisk Caching

The sharing of Expanded Storage between a minidisk cache and paging is done automatically. To control the sizes of the minidisk cache portion of Expanded Storage, use the RETAIN or SET MDCACHE commands to set the minimum and maximum size of Expanded Storage. The size of the minidisk cache can be fixed by



setting the maximum equal to the minimum. Use the QUERY MDCACHE command to show the sizes. For the format and description of these commands, see the *z/VM: CP Commands and Utilities Reference*.

To display the current size of the minidisk cache and the minimum and maximum of its possible size, use the QUERY MDCACHE command.

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. The INDICATE command statistics are for both real and expanded storage MDC use.

**Attention:** If different systems share a minidisk, use the MINIOPT user directory control statement to avoid caching those minidisks. If the minidisk is used in read-only status, it can be shared between two systems. Special attention must be paid to volumes that will be shared in a CSE complex. See “Sharing DASD Volumes” on page 336 for more detail.

### Special Considerations for CMS Minidisk Volumes Shared between Processors

If you have DASD volumes that contain CMS minidisks that are shared among different processors (such as in a CSE complex), you must take action before IPLing VM/ESA. This also applies if you are running VM/ESA as V=V 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
- SHARED=YES on the RDEVICE macroinstruction in the HCPRIO ASSEMBLE file.

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, such as in a CSE complex.

Note, 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 or macroinstruction, 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

## Expanded Storage Planning and Administration

(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, the RDEVICE SHARED YES system configuration file statement, or the RDEVICE SHARED=YES macroinstruction can also be used to specify whether virtual reserve/release CCWs should be translated to real reserve/release CCWs (see Chapter 21, "DASD Sharing," on page 619).

### 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 VM/ESA 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
- Aligned FBA minidisks on page boundaries
- Consider moving the following to cached control units:
  - Minidisks frequently accessed through the noncacheable interfaces
  - Minidisks shared between first- and second-level systems
  - Real devices that are physically shared across processors
- Specify NOMDC on the MINIOPT user directory control 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 additional Expanded or 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 20. 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.

It also describes switching operating modes for 3390 devices. The 3390 Model 9 is not recommended for CP system areas and does not support 3380 track compatibility mode.

---

### 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 z/VM's requirements for disk space in general. It is also helpful to understand the differences and similarities between z/VM's 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 4K 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.

## Allocating DASD Space

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 control 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 4K 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 control statement. When assigning minidisk space, you must know the available extents of your disk in block numbers (see Table 27 on page 605).

**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 26 lists the storage capacity of various CKD devices. Table 27 on page 605 lists the storage capacity of FBA devices.

Table 26. 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

Table 26. CKD DASD Storage Capacities (continued)

DASD Type	Pages per Track	Tracks per Cylinder	Cylinders per Disk	Pages per Cylinder
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	32760 <sup>1</sup>	180
<b>Note:</b> <sup>1</sup> Value can be up to the maximum number of cylinders on the disk.				

**Note:** One page of CP storage is equivalent to 4096 bytes of contiguous disk space.

Table 27. 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 <sup>1, 2</sup>	–	777 <sup>3</sup>	111 <sup>3</sup>
<b>Notes:</b> <sup>1</sup> FBA blocks allocated for CP use must be within the first 64 GB (134,217,728 blocks) of the volume. <sup>2</sup> Due to CMS file system limitations for 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 is 32767 cylinders for a 3390 disk on an ESS device, or about 22 GB of data. IBM suggests that customers defining FBA SCSI disks for use by CMS 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 the <i>z/VM: CMS Commands and Utilities Reference</i> . The maximum size for FBA SCSI disks supported for use by CMS or GCS is 381 GB. <sup>3</sup> 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

## Allocating DASD Space

- Other customized text decks included in the CP module

The HCPLDR option PAGEB causes the CP module to be greatly larger.

The size of the system configuration file does not affect the size of the CP module. Therefore, if you are migrating 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 28 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 28. Minimum Space Required for the CP Module

Device	Number of Cylinders or Pages
3380	18 cylinders
3390	14 cylinders
FBA	2600 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 65535. 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:

$$n = \frac{\text{maxfiles}}{1022 \times \text{ppc}}$$

If the warm start volume is an FBA DASD, use this formula to calculate number of 4 KB pages required:

$$p = \frac{\text{maxfiles}}{1022}$$

$n$  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 26 on page 604. 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 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 230.

---

## 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 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 65535. 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:

## Allocating DASD Space

$$n = \frac{ppc+4+((nvl+119)/120)+((nrs+60)/61)+(((nau*10)+39)/40)}{ppc}$$

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 26 on page 604. 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: Guide for Automated Installation and Service*, 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:

$$\text{cylinders} = \frac{tchs}{ppc} * 2$$



For FBA DASD, use the formula below to calculate the minimum number of pages required to contain the object directory:

$$\text{pages} = \text{tcbs} * 2$$

*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 26 on page 604.

*tcbs*

is an estimate of the total number of pages required for directory control blocks, comprising the index pages, the PROFILE definitions, and the USER definitions. It can be found by using the following formula:

$$\text{tcbs} = 12 + ((512 + \text{cpus} \times 16 + \text{devices} \times 80 + \text{iucvs} \times 32) / 4096) \times \text{users}$$

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 directory entry.

*iucvs*

is the number of IUCV statements defined for a typical virtual machine.

*users*

is the number of virtual machines in the User Directory, including those defined using USER, PROFILE, and POOL statements

## Example

For an installation with 5000 users, the numbers might look like:

$$\text{users} = 5000$$

$$\text{cpus} = 1$$

$$\text{devices} = 20$$

$$\text{iucvs} = 2$$

For these values, the number of pages of storage required would be:

$$\text{tcbs} = 12 + ((512 + 1 \times 16 + 20 \times 80 + 2 \times 32) / 4096) * 5000$$

$$\text{tcbs} = 12 + 2676 = 2688$$

## Allocating DASD Space

$\text{cylinders} = 2688/180 \times 2 = 15 \times 2 = 30$

$\text{pages} = 2688 \times 2 = 5376$

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

The amount of paging space you need to allocate on disk depends on these characteristics of your installation:

- The number and size of logged-on virtual machines
- The number and size of data spaces
- The work load running in each virtual machine.

Use this formula to calculate the number of disk pages you need to allocate for paging space:

$\text{pages} = (\text{block paging factor}) * (\text{total virtual storage})$

Where:

$\text{total virtual storage} =$   
 $(\text{maximum number of logged-on users}) * (\text{average virtual machine size}) +$   
 $(\text{maximum number of defined data spaces}) * (\text{average data space size})$

### Notes:

1. Multiplying the total virtual storage by *block paging factor* (a value from 2 through 4) is needed to account for block paging. CP improves real storage management with block paging support. With this feature, CP or a virtual machine has faster access to disk for paging even without Expanded Storage attached. In block paging, multiple pages of related information move to and from disk as a single unit. The block paging feature requires more disk space to

transfer pages at maximum efficiency and speed. Therefore, you should allocate two to four times more disk space than the total virtual storage used by logged on virtual machines and data spaces.

2. Number and average size of data spaces refers to data spaces provided by VM data space support.
3. 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. For FBA devices to determine the number of blocks needed for paging space, multiply the number of pages by 8, since there are 8 512-byte blocks in a page.

### Example

Table 29 shows the amount of space in cylinders on a 3390 DASD that you need to allocate as paging space based on the number of users and their average virtual machine size. Table 30 on page 612 shows the amount of space in pages and in blocks on an FBA DASD that you need to allocate as paging space based on the same factors. Note that:

- The number of cylinders or pages shown are the minimums required.
- Two is the multiplication factor used for block paging.
- The calculations shown do not take into account that storage may be shared by several users. You should take this factor into account.

**To achieve best performance, use multiple DASD volumes.** For example, if you need to allocate 1000 cylinders on a 3390, try allocating 100 cylinders on each of 10 different 3390s, or 200 cylinders on each of five different disks. Paging in CP is based on *MLOAD* values whereby a device is chosen if its *MLOAD* value is equal to or less than the system average *MLOAD* value. *MLOAD* is the number of queued paging requests for a disk volume times the device service time.

Table 29. 3390 Cylinders Needed for Paging Space Based on Formula

Scenario	Number of Cylinders
100 users; average virtual machine size 500 KB	139
100 users; average virtual machine size 2 MB	569
500 users; average virtual machine size 2 MB	2845
500 users; average virtual machine size 2 MB; 10 users with 2 data spaces each; average data space size 25 MB	4267
10 users; average virtual machine size 16 MB	455
10 users; average virtual machine size 32 MB	910

## Allocating DASD Space

Table 29. 3390 Cylinders Needed for Paging Space Based on Formula (continued)

Scenario	Number of Cylinders
----------	---------------------

### Examples:

1. For example, 2845 cylinders is obtained from the following computation:

$$(2 \times [(500 \times 2 \times 256) + (0 \times 0 \times 256)]) / 180 = 2845$$

multiplier for block paging ——— | users | virtual machine size (MB) | number of data spaces | average data space size (MB) | pages in 1MB | 3390 cylinders | pages per cylinder

**Note:** The actual number obtained by the formula is 2844.44. Rounding up to the next whole cylinder gives 2845 cylinders.

2. The following example shows the calculation for the same installation using VM data space support:

$$(2 \times [(500 \times 2 \times 256) + ((2 \times 10) \times 25 \times 256)]) / 180 = 4267$$

multiplier for block paging ——— | users | virtual machine size (MB) | number of data spaces | users | average data space size (MB) | pages in 1MB | 3390 cylinders | pages per cylinder

**Note:** The actual number obtained by the formula is 4266.67 Rounding up to the next whole cylinder gives 4267 cylinders.

A properly configured 3390 DASD (see note below) in an ESS 2105-F20 and used only for paging can be expected to handle up to 200 pages a second. This assumes that there is adequate space defined on the volume to support block paging, as described earlier in this chapter. This information can be used to estimate the number of volumes required.

**Note:** “Properly configured” means a volume that is allocated to paging only and whose control unit is attached to other volumes that contain paging areas, spooling areas, and guest minidisks. Do not mix spooling and paging on the same volume unless the rates are below 25 pages per second for paging and spooling together. Temporary disk space should be isolated to its own volumes or distributed with minidisks on volumes that do not contain paging or spooling areas. These recommendations allow VM to use seldom-ending channel programs to increase the efficiency of its paging I/O. Placing other types of data on paging volumes may reduce the effectiveness of these channel programs.

Table 30. FBA Pages Needed for Paging Space Based on Formula

Scenario	FBA	
	Number of Pages	Number of Blocks
100 users; average virtual machine size 500 KB	25000	200000

Table 30. FBA Pages Needed for Paging Space Based on Formula (continued)

Scenario	FBA	
	Number of Pages	Number of Blocks
100 users; average virtual machine size 2 MB	102400	819200
500 users; average virtual machine size 2 MB	512000	4096000
500 users; average virtual machine size 2 MB; 10 users with 2 data spaces each; average data space size 25 MB	768000	6144000
10 users; average virtual machine size 16 MB	81920	655360
10 users; average virtual machine size 32 MB	163840	131720

**Examples:**

1. For example, 512000 pages is obtained from the following computation:

$$(2 \times [(500 \times 2 \times 256) + (0 \times 0 \times 256)]) = 512000 \text{ pages}$$

multiplier for block paging — (2)

users — (500)

virtual machine size (MB) — (2)

number of data spaces — (0)

average data space size (MB) — (0)

pages in 1MB — (256)

2. The following example shows the calculation for the same installation using VM data space support:

$$(2 \times [(500 \times 2 \times 256) + ((2 \times 10) \times 25 \times 256)]) = 768000 \text{ pages}$$

multiplier for block paging — (2)

users — (500)

virtual machine size (MB) — (2)

number of data spaces — (2)

users — (10)

average data space size (MB) — (25)

pages in 1MB — (256)

**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

## Allocating DASD Space

- NLS files.

## Allocating Space for CP Abend Dumps

CP abend dumps share spooling space with spool files. Thus, dumps can be sent to spool as long as enough spool space overall is available to contain the category of dump requested. During system initialization, CP sets aside enough disk space to contain a hard abend dump, which changes dynamically as the system operates. A class B user can change the current dump setting with the SET DUMP command. The SET DUMP command also allows the class B user to specify a preferred order for multiple disk volumes to be used for allocation of CP abend dump space. In addition, the amount of disk space will be determined by the dump option specified, that is, CP or ALL. When the CP option is specified, CP sets aside enough disk spool space for a CP-only real storage dump.

During system initialization, enough space should be available for a CP-only dump. Only if the SET DUMP command is entered with the ALL option will enough spool space be set aside for the contents of all of real storage, up to 2 GB.

Table 31 shows you how much dump space in cylinders (CKD DASD) or 4 KB pages and blocks (FBA DASD) you need to allocate for an entire dump of real storage, based on how much real storage you have and what type of DASD you use.

Table 31. Amount of Dump Space Based on Real Storage and Type of DASD

Real Storage Size	3390 Cylinders	FBA	
		Number of Pages	Number of Blocks
4 MB	6	1039	8312
8 MB	12	2063	16504
16 MB	23	4112	32896
32 MB	46	8213	65704
64 MB	92	16416	131328
128 MB	183	32821	262568
256 MB	365	65632	525056

CP uses dump space to write a dump of real storage if CP abends. Allocating dump space is optional. In general, your dump space should be large enough to contain an entire dump of real storage.

To calculate the approximate amount of storage needed on a disk, use the following formula:

$$\text{number of pages needed} = \text{number of pages of real storage below 2G} + x$$

The  $x$  in the formula above represents 9 pages for the dump file information record, 1 page for the symptom record, 1 page for the bit map for each 128 MB of real storage, 1 page for storage key data for every 16 MB, and 3 pages for an adjustment, for a total of 15.

This formula applies to relatively small dumps. For larger dumps the  $x$  in the formula may have to be increased.

## 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:

$$\text{information pages} = 1 + \text{virtual machine pages} / 4096$$

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

$$\begin{array}{ccccccc} 2226 & \times & 12 & = & 26712 & \times & 15 & = & 400680 & / & 25 & = & 16,028 & \text{files/pack} \\ | & & | & & | & & | & & | & & | & & & \\ \text{Number of} & & \text{Pages/Track} & & \text{Tracks/Cyl.} & & \text{Pages/File} & & & & & & & \\ \text{Cylinders} & & & & & & & & & & & & & \end{array}$$

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

$$\begin{array}{ccccccc} 1,672,881 & / & 8 & = & 209,110 & - & 2 & = & 209,108 & / & 13 & = & 16,085 & \text{files/pack} \\ | & & | & & | & & | & & | & & | & & & \\ \text{Number of} & & \text{Blocks/Page} & & \text{Reserved Pages} & & \text{Pages/File} & & & & & & & \\ \text{Blocks} & & & & & & & & & & & & & \end{array}$$

---

## 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

## Allocating DASD Space

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 616.
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

## Allocating DASD Space

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.

---

## Chapter 21. 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.
- Among CMS minidisks on different systems using CSE. For more information on this, see Chapter 10, “Setting Up Cross System Extensions (CSE),” on page 333.

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.

It also includes sections on the restrictions for using reserve/release, using the Multi-Path Lock Facility (MPLF), using cached DASD as high-speed storage devices, and using ESS Parallel Access 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 and on smaller minidisks.

**Note:** Virtual reserve/release is also supported for virtual disks in storage, which are minidisks allocated from system storage rather than mapped to real DASD.

To share a minidisk among virtual machines, you must do the following:

1. Code the MDISK user directory control statement in the virtual machine's directory entry. For example, the following MDISK user directory control 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 user directory control 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 user directory control statement in another virtual machine's directory. The LINK user directory control 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 user directory control 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 user directory control 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 vmlpass
```

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 vmlpass
```

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 either the system configuration file using RDEVICE statements or in the HCPRIO ASSEMBLE file using an RDEVICE macroinstruction. All DASD that hold minidisks must be attached to SYSTEM, either at system initialization by appearing in either the system configuration file using CP\_OWNED or USER\_VOLUME\_LIST statements or in the HCPSYS ASSEMBLE file using a SYSCPVOL or SYSUVOL macroinstruction or later by issuing the command ATTACH *rdev* TO SYSTEM.

For more information about these system configuration file statements, see "RDEVICE Statement" on page 188, "CP\_OWNED Statement" on page 73, and "USER\_VOLUME\_LIST Statement" on page 255. For more information about these macroinstructions, see Appendix B, "Defining Your System (HCPSYS Macroinstructions)," on page 643 and Appendix C, "Defining I/O Devices Using HCPRIO," on page 705.

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) and the RDEVICE macroinstruction (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
```

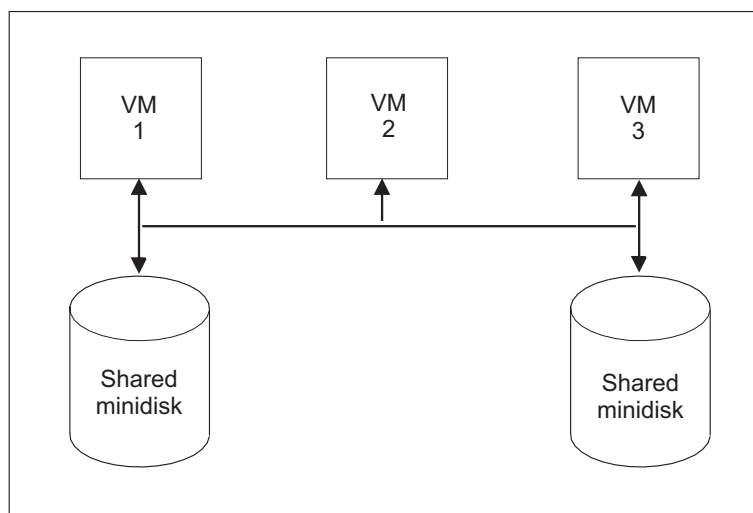
the device at real device number 197 cannot be shared with other operating systems. However, you should enter SET SHARED OFF only after careful

consideration. See the *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 622. 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 624 and “Spool File Directory Statements” on page 341.

Figure 33 shows two virtual machines sharing a minidisk defined on a DASD through virtual reserve/release.

#### Real Machine



Three virtual machines sharing two different minidisks through virtual reserve/release support and one channel path

Figure 33. 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

## DASD Sharing

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.

---

### 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 user directory control 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 the *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 the *z/VM: CMS File Pool Planning, Administration, and Operation* book 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 (in either the RDEVICE system configuration file statement or

macroinstruction) at real device number 872. The DEDICATE user directory control 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:

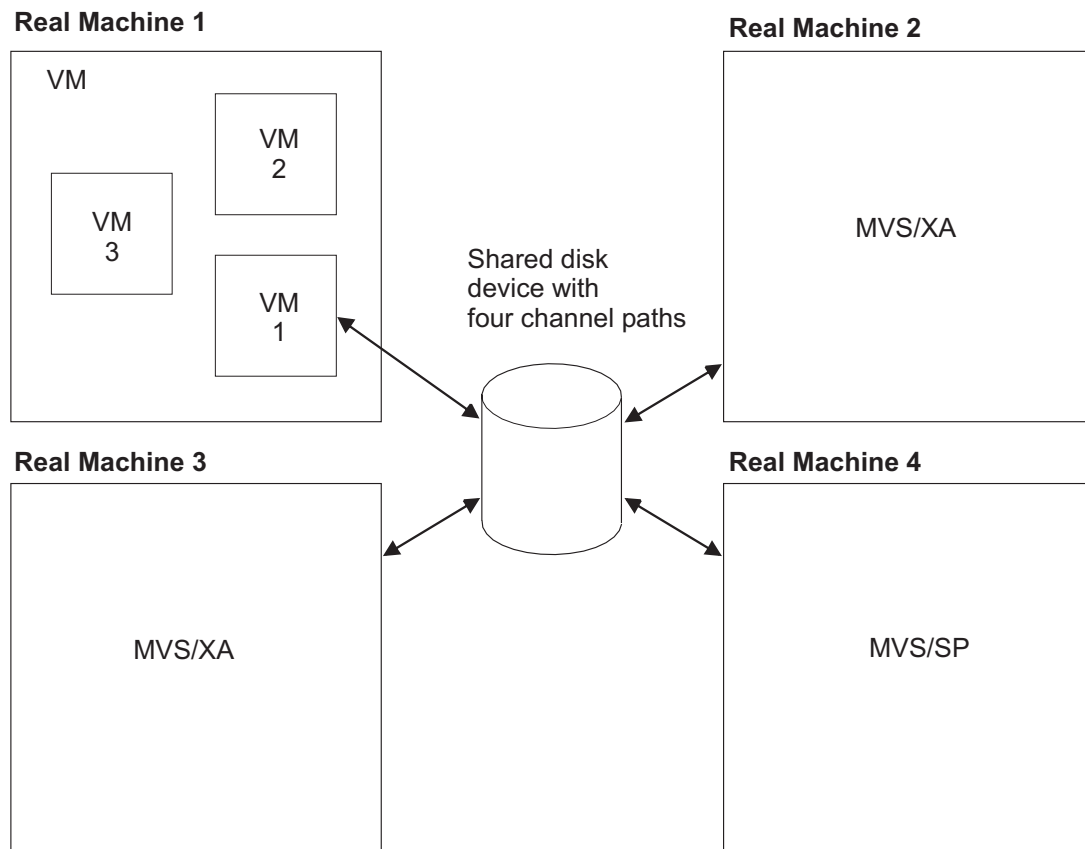
```
attach 872 to vm1 as 872
```

is equivalent to the previous DEDICATE user directory control 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 34 on page 624.

## DASD Sharing



Virtual machine 1 sharing a disk device with three operating systems in three real machines using real reserve/release support

Figure 34. Real Reserve/Release

## When to Use Real Reserve/Release

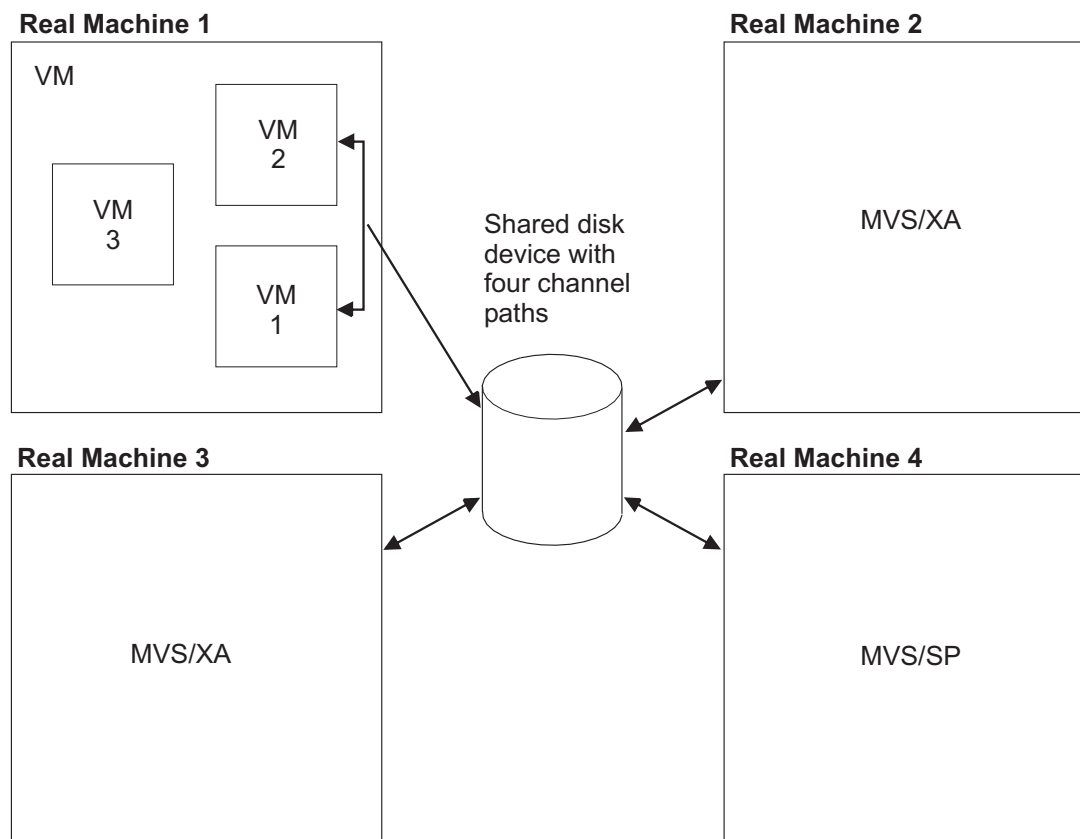
spot id=urrr xref= 'When to Use Real Reserve/Release'. In general, you should use real reserve/release when you need to share a DASD among one virtual machine and other systems. Note, a dedicated DASD cannot be shared among virtual machines but can be shared among one virtual machine and other systems.

---

## 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 35 on page 625 shows VM sharing a DASD between two virtual machines and a native system.





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 35. 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 619, 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 user directory control 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 user directory control 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 619.

## DASD Sharing

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.

---

## 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.

---

## Reserve/Release Summary

Table 32 summarizes the differences between different settings of SET SHARED and the V attribute of the MDISK user directory control statement.

*Table 32. 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.

---

## 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 the D/T3990 model 6 DASD control unit. The MPLF provides locking serialization for data on the DASD, and the operating systems or programs use the D/T3990 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 D/T3990 model 6 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 control 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 control 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 D/T3990 model 3 MPLF RPQ, while real MPLF support is based on the D/T3990 model 6.

Like the real MPLF, simulated MPLF requires the user to be authorized by the LKFAC option of the directory OPTION control 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

## DASD Sharing

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 the *z/VM: General Information* book 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 36 shows a storage control unit logically attached to a string of 3380 DASD.

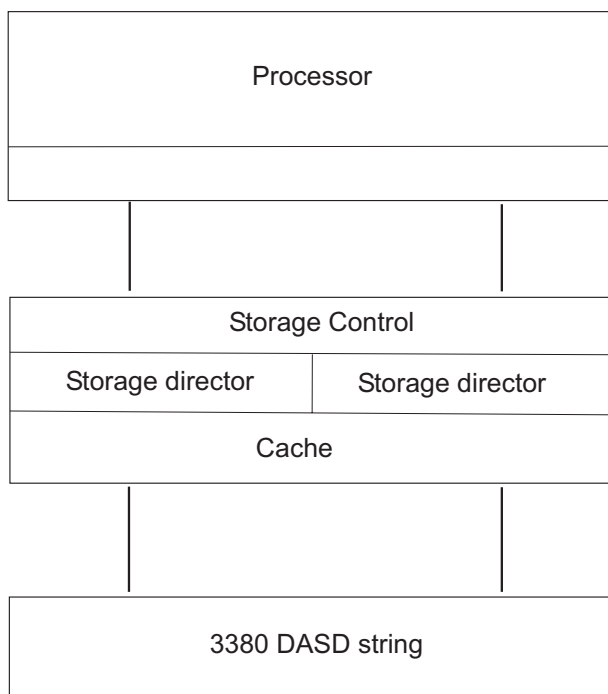


Figure 36. 3380 Cached DASD

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 combinations of 3880 and 3990 storage controls with 3380 and 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 control 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 624).

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 a 3990 Storage Models 3 or 6.

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 or macroinstruction. For example, you would code the RDEVICE system configuration file statement as:

```
RDEVICE 197-19A TYPE DASD SHARED YES
```

## DASD Sharing

Or, you would code the RDEVICE macroinstruction as:

```
RDEVICE DEVNO=(197,4),DEVTYPE=3390,SHARED=YES
```

Both of these statements define 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 user directory control statement (and, optionally, a MINIOPT user directory control 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 user directory control 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 619.

**Full-Pack Minidisk:** To define a 3390 Model 3 as a full-pack minidisk, you need in your virtual machine’s directory entry an MDISK user directory control statement and, optionally, a DASDOPT user directory control 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 user directory control statement and its operands (DEVCTL, SYSCTL, and NOCTL) determine which types of CCWs the device can accept. If you do not include a DASDOPT user directory control statement after the MDISK user directory control statement, the default is DEVCTL. For more information on the DASDOPT user directory control statement, see “DASDOPT Directory Control Statement (Device)” on page 472.

## Defining a Cached DASD as a Dedicated Device

You can give your virtual machine exclusive use of a cached DASD with the DEDICATE user directory control statement. In your virtual machine’s directory, code a DEDICATE user directory control statement and optionally a DASDOPT user directory control 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 user directory control statement, the default for any 3990 is DEVCTL.

## Using ESS Parallel Access Volumes

z/VM provides support for the Parallel Access Volumes (PAV) feature of the IBM Enterprise Storage Server (ESS). The ESS must be defined to z/VM as 3390 Model 2, 3, or 9 DASD on a 3990 Model 3 or 6 Storage Controller. 3380 track-compatibility mode for the 3390 Model 2 or 3 DASD is also supported. z/VM supports the PAV feature only for guest use where the device is dedicated (attached) to the guest.

The PAV feature allows you to configure base and alias (logical) DASD volumes. Figure 37 illustrates the relationship between base and alias volumes. The base volume is the real (physical) DASD space. It has a unique subchannel ID. Alias volumes (X and Y in this example) are logical volumes that map the physical space occupied by the base. The alias volumes do not have their own space; they are “shadows” of the base. However, each alias volume also has a unique subchannel ID. This allows concurrent I/O to a base volume and all of its associated alias volumes.

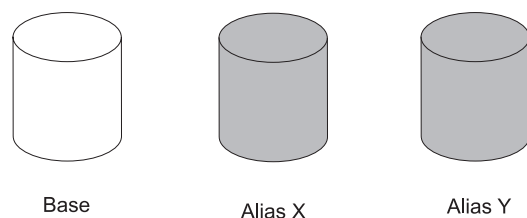


Figure 37. Base and Alias Parallel Access Volumes

For z/VM, base volumes are defined in IOCP as UNIT=3990 (or 2105) on the CNTLUNIT statement and UNIT=3390B (or 3380B) on the IODEVICE statement. Alias volumes are defined as UNIT=3990 (or 2105) on the CNTLUNIT statement and UNIT=3390A (or 3380A) on the IODEVICE statement. Each volume (base or alias) can be assigned any available VM real device number. Use the ESS configuration console to initially define which volumes are base volumes, which volumes are alias volumes, and which alias volumes are associated with each base. Use the CP QUERY PAV command to view the current allocation of base and alias volumes.

## Using PAV for Workload Balancing

The main purpose of the PAV feature is to increase system throughput and reduce I/O bottlenecks by allowing multiple concurrent I/O operations to the same physical DASD space. This allows the guest to use the PAV feature for workload balancing.

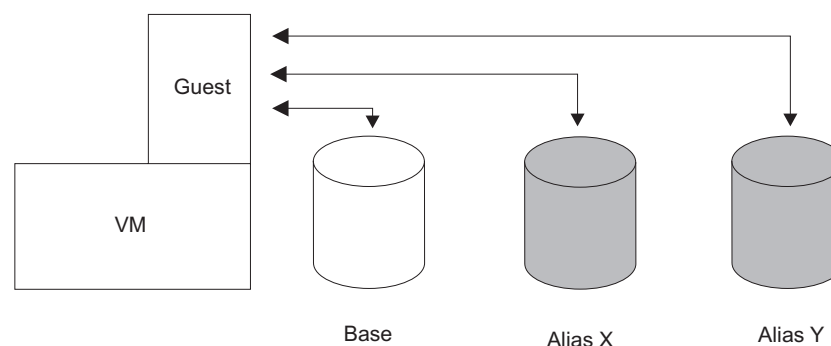


Figure 38. Using Parallel Access Volumes for Workload Balancing

## DASD Sharing

Figure 38 on page 631 illustrates the use of PAV for workload balancing. For this application, the base volume and all its associated alias volumes (X and Y in this example) are dedicated to the guest with the CP ATTACH command. The virtual device numbers and subchannel IDs assigned to the volumes are immaterial. To maximize workload performance, the guest can schedule I/O on whichever of the base and alias volumes seems most appropriate, and can schedule I/O concurrently on some or all of the associated volumes.

To use PAV in this manner, the guest must contain support for managing and serializing the workload data across the volumes. z/VM acts only as the “pipe” between the guest and the hardware. **Once the necessary base and associated alias volumes are attached to the guest, the guest must manage their use.**

An example of a guest operating system that contains the necessary support for PAV workload balancing is z/OS. In addition to the workload support, a z/OS guest has the ability to dynamically move an alias volume from its current base volume to a different base volume if the workload demands it. On z/VM, you can use the CP QUERY PAV command to keep track of the dynamic movement of alias volumes performed by a z/OS guest.

## Using PAV for DASD Sharing

Another use of the PAV feature is for sharing data across multiple operating systems running as guests on z/VM.

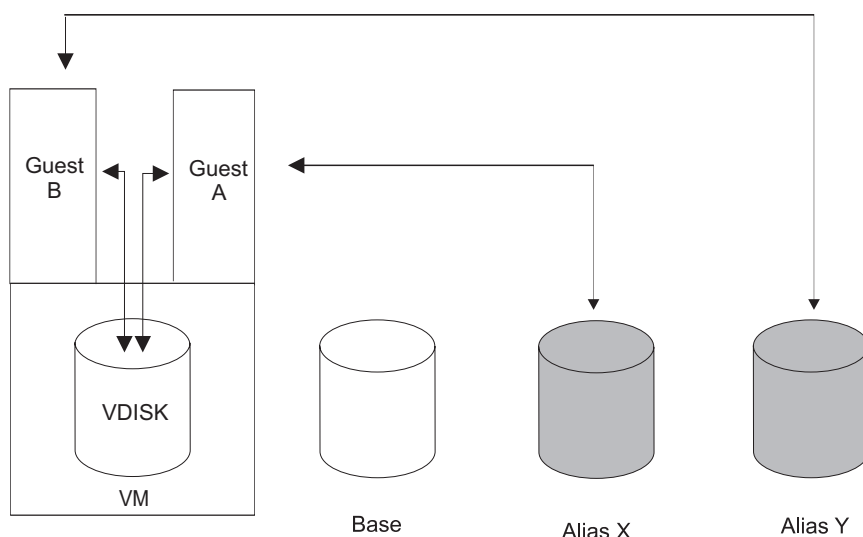


Figure 39. Using Parallel Access Volumes for DASD Sharing

Figure Figure 39 illustrates the use of PAV for sharing data among multiple guests. In this example, alias X is attached (with the CP ATTACH command) to Guest A. Alias Y is attached to Guest B. Both alias X and alias Y are associated with the same base volume. The base volume is not attached to either guest in this example, but it can also be used for sharing. The virtual disk in storage shown in the example is linked to both Guest A and Guest B through the z/VM user directory.

The linked virtual disk in storage is important because it demonstrates a method for serializing the data that is shared between the two guests. The virtual disk in storage could contain, for example, the VSE lock file. The virtual disk in storage would not be necessary in a read-only environment. In that case, the alias volumes could be attached to the guests with the R/O option on the ATTACH command. A



method to serialize access to the alias volumes must be devised, because reserve/release is not supported on alias volumes (the CCWs will be rejected). By using a linked virtual disk in storage, reserve/release can be used against the virtual disk in storage to control access to the alias volumes. Also, free (not attached) base and alias volumes associated with the volumes being shared must be clear of all write activity.

**It is the sole responsibility of the guests to manage and serialize any data that is shared using alias volumes.** z/VM provides only the “pipe” for data exchange between the hardware and the guests.

## **z/VM Restrictions on Using PAV**

1. An alias volume cannot be IPLed.
2. An alias volume cannot be attached to the SYSTEM user ID.
3. An alias volume cannot be attached to a guest if its associated base volume is currently attached to SYSTEM. Alternatively, a base volume cannot be attached to SYSTEM if any of its associated alias volumes are attached to one or more guests.
4. An alias volume will not come on-line to z/VM without an associated base volume. Also, a base volume must have at least one associated alias volume for z/VM (for example, the QUERY PAV command) to recognize the device as a Parallel Access Volume.
5. A base volume cannot be changed or deleted with the SET RDEVICE, DELETE RDEVICE, DELETE DEVICE, or MODIFY DEVICE command unless all associated alias volumes have been deleted with the DELETE RDEVICE command.

## DASD Sharing

---

## Chapter 22. Defining and Managing SCSI FCP Disks

---

### Overview

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 zSeries 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. Directory, paging, and spooling allocations, however, must reside within the first 64 GB (16,777,216 blocks) of a CP-formatted volume. Other types of CP allocations (TDSK, PERM, and PARM) may exist beyond the first 64 GB.

**Note:** Due to CMS file system limitations for 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 is 32767 cylinders for a 3390 disk on an ESS device, or about 22 GB of data. IBM suggests that customers defining FBA SCSI disks for use by CMS 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 the *z/VM: CMS Commands and Utilities Reference*. The maximum size for FBA SCSI disks supported for use by CMS or GCS is 381 GB.

An emulated FBA 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 40 on page 636 for an illustration of these required elements.

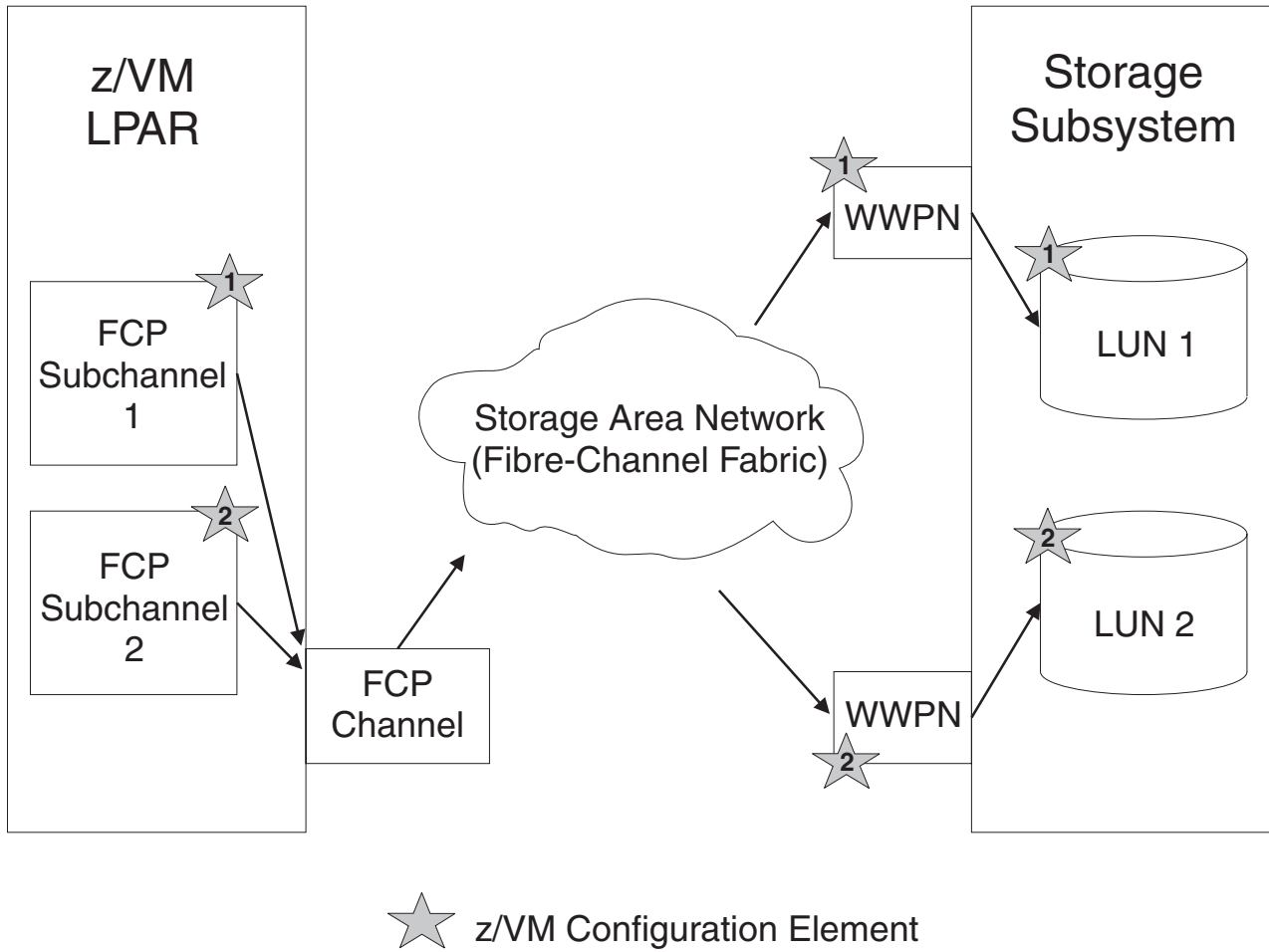


Figure 40. Required Elements for SCSI FCP definition

Figure 41 on page 637 illustrates the overall system architecture related to z/VM's 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 40 are contained in the "FCP Channel and Storage Area Network" portions of Figure 41 on page 637.

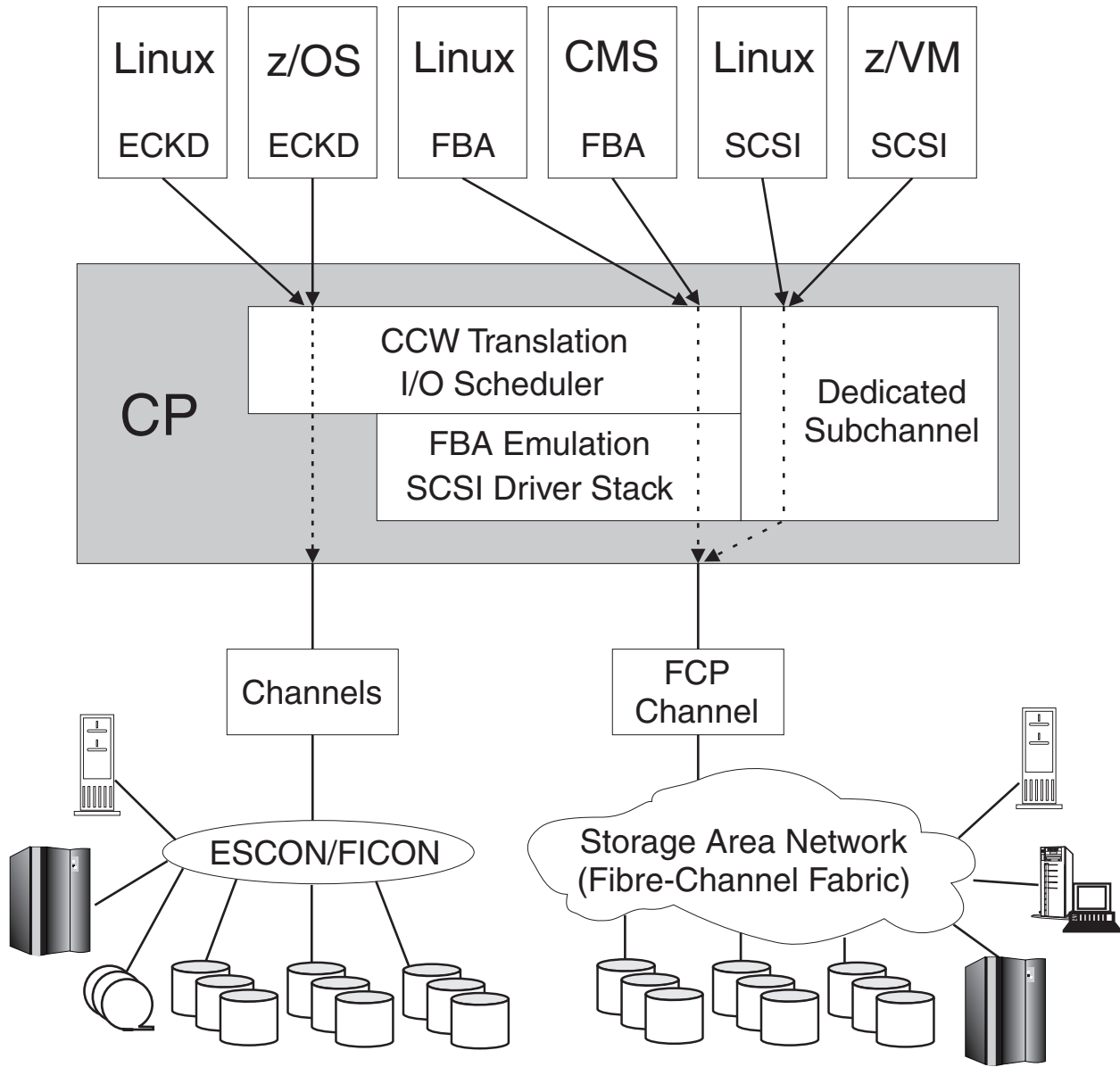


Figure 41. SCSI System Architecture

## 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* 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

## Defining and Managing SCSI FCP Disks

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” on page 635: 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 commands can be used to manage emulated FBA disks:

- SET EDEVICE, to create, modify, or clear an emulated-device definition
- DELETE EDEVICE, to delete an emulated-device definition
- QUERY EDEVICE, to query an emulated-device definition.

See the *z/VM: CP Commands and Utilities Reference* for more information on these commands.

**Note:** The SET EDEVICE and DELETE EDEVICE commands are not allowed when HCM and HCD are controlling z/VM’s 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 the *z/VM: CP Commands and Utilities Reference* for more information on these commands.

---

## Additional Considerations

The following are restrictions for emulated SCSI devices:

- For CP volumes (formatted with CPFMTXA), paging, spooling, and directory allocations must not be allocated above page 16,777,216 (representing 64 GB of disk storage) on the emulated FBA disk.
- As with all FBA minidisks, Reserve and Release CCW commands issued to an emulated FBA disk are supported only among guests within the same z/VM system image. If a guest operating system has access to either a dedicated FBA disk or to an emulated FBA full-pack minidisk that has been defined as shared among multiple z/VM images, then Reserve and Release CCW commands will be rejected.
- A single FCP channel can be shared by multiple logical partitions and multiple virtual machines, but 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 will appear to the SCSI controller as being the same host system (or SCSI initiator), potentially leading to problems with concurrent-write access to the disk and with error-recovery procedures. 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.
- IBM strongly recommends that you never assign different logical unit numbers (LUNs) to the same logical device within a SCSI controller. Otherwise, shared access to a device may happen without being detected, and unpredictable results can occur.





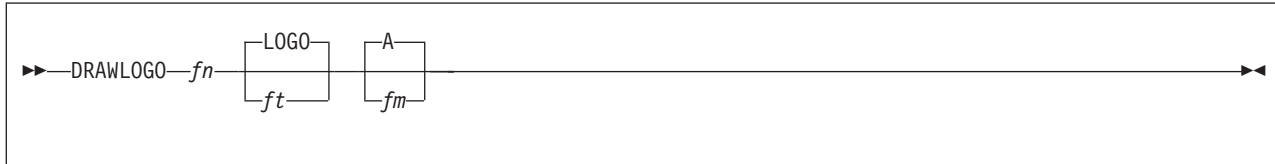
---

## 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 MAINT 2C2 disk.

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 on how to use this utility, see the “Creating Logo Screens” on page 282.

## 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. To put the new logo configuration file into production, use the CP REFRESH command. See the *z/VM: CP Commands and Utilities Reference*.
3. DRAWLOGO invokes the X\$DRWL\$X XEDIT macro, which must be available.

## Return Codes

In addition to the return code listed, DRAWLOGO may return a return code from the CMS LISTFILE and CMS XEDIT commands. See the *z/VM: CMS Commands and Utilities Reference* for a list of return codes that the CMS LISTFILE and CMS XEDIT commands generate.

### Code Meaning

- |          |                                  |
|----------|----------------------------------|
| <b>4</b> | No minidisk accessed in R/W mode |
|----------|----------------------------------|

## Appendix B. Defining Your System (HCPSYS Macroinstructions)

This appendix describes how to code the system control file, HCPSYS ASSEMBLE, and describes each of the HCPSYS macroinstructions, including the purpose, syntax, and examples, and how to use them.

If you do not define your system using a system configuration file, you must code the system control file, HCPSYS ASSEMBLE. You can code in the HCPSYS ASSEMBLE file any of the macroinstructions listed in Table 33. You must code the ones listed as **Required**; the others are optional. Table 33 shows you where information on each macroinstruction can be found, and whether it is required or optional, and gives a brief description of each macroinstruction. Although the macroinstructions are in alphabetic order, this is not the order in which you would code them in HCPSYS ASSEMBLE. See “Coding HCPSYS ASSEMBLE” on page 644 for the order in which the macroinstructions must appear in the file.

Table 33. HCPSYS Macroinstructions

Macro	Required or Optional	Function	Can Be Replaced by System Configuration File Statement
CSELDEV (page 646)	Optional	Changes the defaults for the CSE track location and format for particular device types.	XLINK_DEVICE_DEFAULTS
CSELVOL EXCLUDE (page 649)	Optional	Defines the DASD volumes to be excluded from the cross system link operation.	XLINK_VOLUME_EXCLUDE
CSELVOL INCLUDE (page 650)	Optional	Defines the DASD volumes to be included in the cross system link operation.	XLINK_VOLUME_INCLUDE
CSESYS (page 653)	Optional	Describes the topography of a CSE complex.	XLINK_SYSTEM_EXCLUDE XLINK_SYSTEM_INCLUDE XSPOOL_SYSTEM
CSETRACE (page 658)	Optional	Establishes the number of pages of CP storage allocated for the CSE trace tables.	XSPOOL_TRACE
CSEUSER (page 659)	Optional	Defines virtual machines that will not participate in cross system spooling and cross system message and query commands.	XSPOOL_XLIST_INPUT XSPOOL_XLIST_OUTPUT
SYSACNT (page 661)	Optional	Specifies the user IDs of up to two virtual machines for which the CP accounting system service (*ACCOUNT) is to accumulate records.	SYSTEM_USERIDS
SYSADDIN (page 663)	Optional	Specifies a list of installation-added entry points in CP which are to be called during system initialization.	CP_ADDON_INITIALIZE _ROUTINES
SYSCPVOL (page 664)	<b>Required</b>	Defines and generates a list of CP-owned DASD volumes. (CP-owned DASD volumes include the CP system residence volume and those DASD volumes that contain real system paging, spooling, dump, directory, and temporary disk space.)	CP_OWNED

## Defining Your System

Table 33. HCPSYS Macroinstructions (continued)

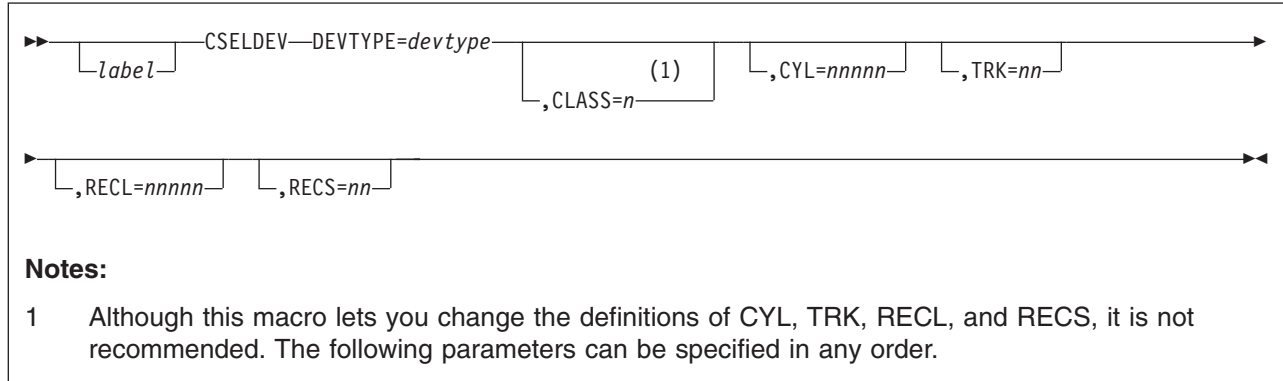
Macro	Required or Optional	Function	Can Be Replaced by System Configuration File Statement
SYSDUMP (page 667)	Optional	Defines the user ID of the virtual machine that receives system dump files.	SYSTEM_USERIDS
SYSEND (page 668)	<b>Required</b>	Ends the HCPSYS ASSEMBLE file.	Not Applicable
YSEREP (page 669)	Optional	Specifies the user IDs of up to two virtual machines for which the *LOGREC system service is to accumulate EREP records.	SYSTEM_USERIDS
SYSEXCL (page 671)	Optional	Defines the volumes that are to be excluded from the user volume list.	USER_VOLUME_EXCLUDE
SYSFCN (page 673)	Optional	Contains the privilege classes authorized to perform internal CP functions that are controlled by privilege class.	PRIV_CLASSES
SYSFORM (page 675)	Optional	Creates a list of user form names and their corresponding operator form numbers.	FORM_DEFAULT USERFORM
SYSID (page 677)	Optional	Defines a system identifier for the processor on which you run VM/ESA.	SYSTEM_IDENTIFIER SYSTEM_IDENTIFIER_DEFAULT
SYSINCL (page 680)	Optional	Defines user volumes by means of a generic volume identifier.	USER_VOLUME_INCLUDE
SYSJRL (page 682)	Optional	Specifies the inclusion of the journaling or password suppression facility.	FEATURES JOURNALING
SYSMAXU (page 686)	Optional	Specifies the user logon limit.	FEATURES
SYSOPR (page 687)	<b>Required</b>	Defines the user ID of the primary system operator.	SYSTEM_USERIDS
SYSOPTS (page 689)	Optional	Sets CP initialization options.	Not Applicable
SYSPCLAS (page 691)	Optional	Specifies classification titles for specific classes of printed output.	PRINTER_TITLE
SYSRES (page 693)	<b>Required</b>	Describes the layout of the CP system residence disk.	FEATURES SYSTEM_RESIDENCE
SYSSTORE (page 697)	<b>Required</b>	Defines your real storage configuration.	STORAGE
SYSSYMP (page 699)	Optional	Specifies the user IDs of up to two virtual machines for which the *SYMPTOM system service is to accumulate symptom records.	SYSTEM_USERIDS
SYSTIME (page 701)	<b>Required</b>	Specifies information required to set the hardware time-of-day (TOD) clock.	TIMEZONE_BOUNDARY TIMEZONE_DEFINITION
SYSUVOL (page 703)	Optional	Generates a list of DASD that contain minidisks for users.	USER_VOLUME_LIST

## Coding HCPSYS ASSEMBLE

This appendix contains a section for each HCPSYS macroinstruction. Each section describes the function, syntax, operands (required and optional), and coding examples for the macroinstruction. Some rules you must follow when you code the HCPSYS ASSEMBLE file are:

- SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME are required and must be the first macroinstructions in HCPSYS ASSEMBLE. Except for being first in the file, they do not have to appear in any order.
- The SYSEND macroinstruction must be the last instruction in the HCPSYS ASSEMBLE file.
- The optional macroinstructions must appear after SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME, and before SYSEND. Except for this rule and the following rule, the optional macroinstructions can appear in any order.
- All CSExxxx macroinstructions must appear after all SYSxxxx macroinstructions and before the SYSEND macroinstruction.
- Do not code START, CSECT, or END assembler instructions—the macroinstruction expansions generate these instructions. If you do code these instructions, MNOTEs are generated.
- You may code labels on the macroinstructions to identify them. The macroinstruction expansions ignore the labels.

## CSELDEV (Optional)



## Purpose

Code the CSELDEV macroinstruction to change the defaults for the CSE track location format for particular device types. You may specify as many CSELDEV macros as you need. If CSELDEV macros are coded, they must appear after the CSESYS macro and before the SYSEND macro.

See Usage Note 5 on page 648 for a warning about changing the defaults for this macro.

## Parameters

### DEVTYPE=

indicates that a device type is to be changed.

### devtype

specifies the type of DASDs that are to have their default values changed.

### CLASS=n

whenever DEVTYPE=device is specified, *n* defines the cylinder size group to which the particular device being described belongs.

When DEVTYPE=3380 is specified, *n* should have the following values (the default value is 3):

CLASS	Cylinders
0	1770
1	1770
2	1770
3	2655

When DEVTYPE=3390 is specified, *n* should have the following values:

CLASS	Cylinders
1	1113
2	2226
3	3339
9	10017 <sup>1</sup>

<sup>1</sup> Value can be up to the maximum number of cylinders on the DASD.

When DEVTYPE=9345 is specified, *n* should have the following values:

CLASS	Cylinders
1	1440
2	2156

It is always correct to specify a class value which represents a number of cylinders that is higher than the actual number of cylinders on the device or to take the default.

**CYL=nnnnn**

defines the cylinder where the CSE track is located for every DASD defined as *devtype* within the above syntax diagram. The value of *nnnnn* should be from 0 to the maximum cylinder on the volume. The default is 0. See Usage Note 5 on page 648.

**TRK=nn**

defines the track on which the CSE track is located for every CKD DASD defined as *devtype* within the above syntax diagram. The variable *nn* should be in the range of 0 to 64 but cannot be 0 if CYL=0. The default is 1. See Usage Note 5 on page 648.

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 TRK=*nn* is not used for an ECKD capable CKD device.

**RECL=nnnnn**

defines the number of cylinders made available for CSE 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 links to minidisks starting on a cylinder with a number higher than *nnnnn* are denied. The value of *nnnnn* should be in the range of 203 to the maximum cylinder on the volume. Table 34 on page 652 shows the map record defaults for each device type. See Usage Note 5 on page 648.

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. 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 CSE. See Usage Note 5 on page 648.

**RECS=nn**

defines the number of map records on the CSE track. You should use as many records as will fit on one track. Table 34 on page 652 shows the map record defaults for each device type. See Usage Note 5 on page 648.

## Usage Notes

1. As many CSELDEV macros as are needed may be specified.
2. The CYL, TRK, RECL, and RECS keywords allow you to specify CSE track 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.
3. If the CSELDEV CLASS operand is not coded for 3380 devices, CSE uses a value of 2655 cylinders for all type K 3380 devices and 1770 for non-type K devices.

## CSELDEV

4. The default parameters for model K devices are those associated with 3380 CLASS=3, which are 2655 cylinders and 12 map records. If you want to make all 3380s use the same parameters as model Ks, add the three following CSELDEV macros to each HCPSYS ASSEMBLE file:

```
CSELDEV  DEVTYP=3380,CLASS=0,RECL=2655,RECS=8
CSELDEV  DEVTYP=3380,CLASS=1,RECL=2655,RECS=8
CSELDEV  DEVTYP=3380,CLASS=2,RECL=2655,RECS=8
```

5. Although the CSELDEV macro lets you change the definitions of cylinder, track, record length, and map records for the CSE track, it is *not* recommended. This lets you provide link support for new devices with new geometries, should that become necessary in the future.

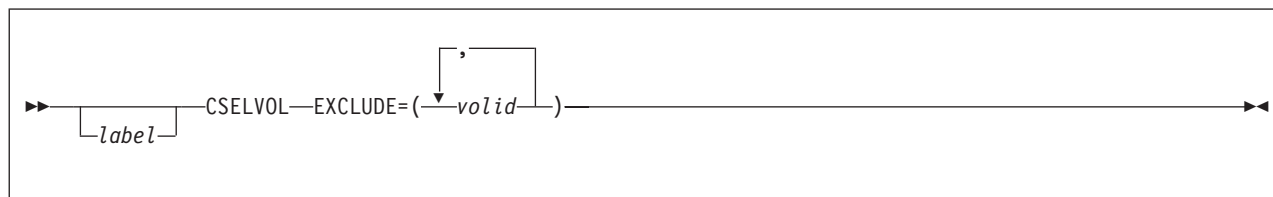
**Attention:** Current upper limits may exceed capacities of present devices and cause errors when attempting to access beyond a particular device limit. Therefore, any change of defaults should be done with caution.

## Migration Aids

You can achieve the function of the CSELDEV macroinstruction by using the XLINK\_DEVICE\_DEFAULTS statement in a system configuration file. See “XLINK\_DEVICE\_DEFAULTS Statement” on page 259.



## CSELVOL EXCLUDE (Optional)



### Purpose

Code the CSELVOL EXCLUDE macroinstruction to define DASD volumes to be excluded from the cross system link operation. You may enter as many CSELVOL EXCLUDE macros as you need. If CSELVOL EXCLUDE macros are coded, they must appear after the CSESYS macro and before the SYSEND macro. They should be identical with the CSELVOL EXCLUDE macros for the other systems you are generating. If you omit the CSELVOL EXCLUDE macro, the default excludes all volumes residing on the system that are not specified in the CSELVOL INCLUDE macro.

### Parameters

#### EXCLUDE

indicates that the volumes specified as *valid* are *not* to be shared through cross system link. These volumes, if mounted and shared among systems, have no extended link protection.

**Note:** If CSELVOL INCLUDE or EXCLUDE is not specified, all DASD volumes are included in cross system link.

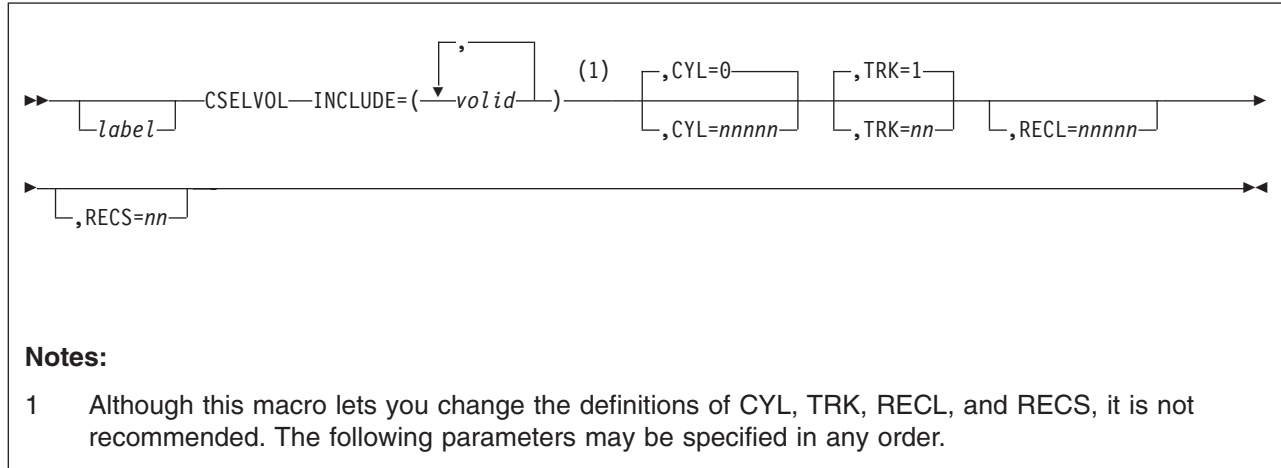
#### *valid*

specifies the 1- to 6-character volume identifier of a volume to be excluded from sharing by cross system link. You can use generic volume IDs. A generic volume identifier is a 1- to 6-character pattern that can use asterisks (\*) in place of any number of characters and percent signs (%) in place of exactly one character. For example, CSELVOL EXCLUDE=(HC%P\*) excludes all volumes which begin with HC and have a P as their fourth character.

### Migration Aids

You can achieve the function of the CSELVOL EXCLUDE macroinstruction by using the XLINK\_VOLUME\_EXCLUDE statement in a system configuration file. See "XLINK\_VOLUME\_EXCLUDE Statement" on page 266.

## CSELVOL INCLUDE (Optional)



## Purpose

Code the CSELVOL INCLUDE macroinstruction to define the DASD volumes to be included in the cross system link operation. You may enter as many CSELVOL INCLUDE macros as you need. If CSELVOL INCLUDE macros are coded, they must appear after the CSESYS macro and before the SYSEND macro. They should be identical with the CSELVOL INCLUDE macros for the other systems you are generating. If you do not use this macro, *all* volumes are included.

See Usage Note 5 on page 651 for a warning about changing the defaults for this macro.

## Parameters

### INCLUDE

indicates that the volumes specified as *valid* are to be shared through cross system link.

### *valid*

specifies the 1- to 6-character volume identifier of a volume to be shared by cross system link. You can use generic volume IDs. A generic volume identifier is a 1- to 6-character pattern that can use asterisks (\*) in place of any number of characters and percent signs (%) in place of exactly one character. For example, CSELVOL INCLUDE=(HC%P\*) includes all volumes which begin with HC and have a P as their fourth character.

### CYL=*nnnnn*

defines the cylinder where the CSE track is located for every *valid*. The value of *nnnnn* should be from 0 to the maximum cylinder on the volume. The default is 0. Keep in mind that some devices do require multiple cylinders for the CSE area. See Table 34 on page 652 for the cylinder defaults. If a volume which is to be included in cross system link is to be used as an IPL device for the stand-alone dump utility, a cylinder other than the default of 0 must be specified for that volume. See Usage Note 5 on page 651.

### TRK=*nn*

defines the track on which the CSE track is located for every CKD DASD

defined as *valid* within the above syntax diagram. The value of *nn* should be in the range of 0 to 64, but it cannot be 0 if CYL=0. The default is 1. See Usage Note 5.

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 TRK=*nn* is not used for an ECKD capable CKD device.

#### RECL=*nnnnn*

defines the number of cylinders made available for CSE use. This value should be at least equal to the number of cylinders on the volume. If it is smaller than the number of cylinders on the volume, links to minidisks starting on a cylinder with a number higher than *nnnnn* are denied. The value of *nnnnn* should be in the range of 203 to the maximum cylinder on the volume. Table 34 on page 652 shows the map record defaults for each device type. See Usage Note 5.

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. 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 CSE. See Usage Note 5.

#### RECS=*nn*

defines the number of map records on the CSE track. The variable *nn* should be at least equal to the number of systems that share the volume. If *nn* is smaller than the number of sharing systems, then all links to minidisks on this volume are denied. It is recommended that you use as many records as can fit on one track. Refer to Table 34 on page 652 for the default values. See Usage Note 5.

## Usage Notes

1. You can specify as many CSELVOL INCLUDE macros as you need. As many CSELVOL macros as are required may be specified.
2. All the CSELVOL INCLUDE macros must precede any CSELVOL EXCLUDE macros.
3. If CSELVOL INCLUDE or EXCLUDE macros are not specified, *all* DASD volumes are included in cross system link.
4. If CSELVOL INCLUDE macros are coded in HCPSYS, volumes not specified in the CSELVOL INCLUDE statements are not included in cross system link.
5. Although the CSELVOL INCLUDE macro lets you change the definitions of cylinder, track, record length, and map records for the CSE track, we do *not* recommend that you do so. Leaving the defaults alone enables you to provide link support for new devices with new geometries, should that support 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.

## CSELVOL INCLUDE

Table 34 shows the map record defaults for each device type.

Table 34. Map Record Defaults for Each Device Type

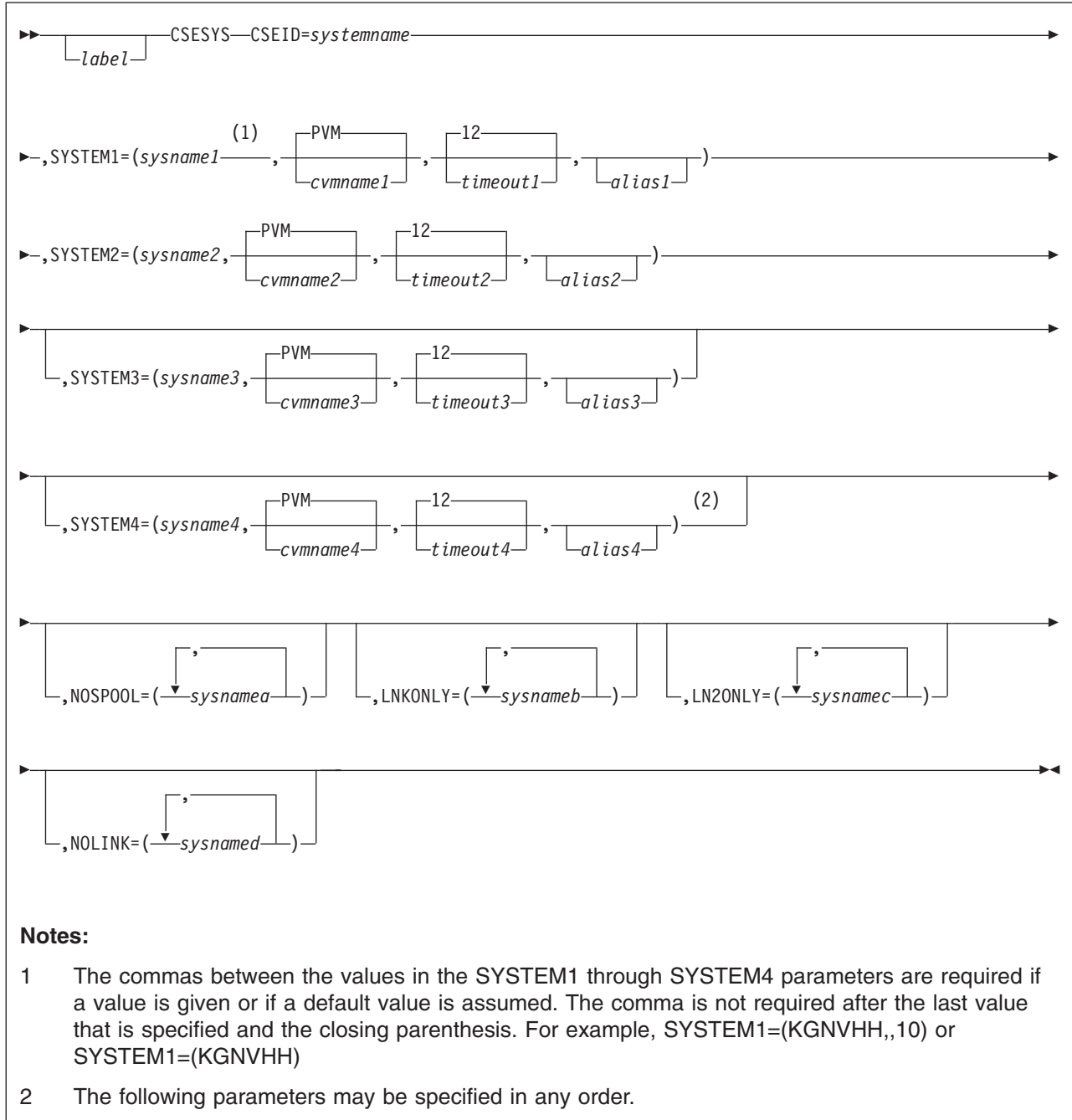
Device Type	Number of Map Records	Map Record Length	Number of Cylinders for CSE Area
3380 non-type K	11	1770	1
3380 type K	8	2655	1
3390-1	12	1113	1
3390-2	8	2226	1
3390-3	8	3339	1
3390-9	56	65520	9
OTHER	8	1024	1

6. The values selected for CYL and TRK must be valid for the device type or types on which the specified volumes reside but are not checked by this macro, since the device type is unknown.
7. 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 fit on a track of such “*other*” device.
8. You must use the XLINK FORMAT command to format any *volids* specified. (See “Generating the Cross System Link Facility” on page 349.) Otherwise when you IPL a system that uses CSE, CP does not permit access to, or minidisks on, that volume identifier. For more information about the XLINK FORMAT command, see *z/VM: CP Commands and Utilities Reference*.
9. The number of cylinders for a CSE area (in Table 34) cannot be changed. They describe the number of cylinders needed to hold the CSE information for ECKD DASD only. If the last cylinder on the volume is specified for the CSE area and multiple cylinders are needed, CSE will not wrap to the beginning of the volume for the other CSE area cylinders.

## Migration Aids

You can achieve the function of the CSELVOL INCLUDE macroinstruction by using the XLINK\_VOLUME\_INCLUDE statement in a system configuration file. See “XLINK\_VOLUME\_INCLUDE Statement” on page 268.

## CSESYS (Optional)



## Purpose

Code the CSESYS macroinstruction to specify the topography of the CSE complex. The CSESYS macroinstruction is optional; if you code it, you must place it after all the SYSxxxx macroinstructions in HCPSYS ASSEMBLE, but before the SYSEND macroinstruction. Only one CSESYS macroinstruction may be specified.

## Parameters

### **CSEID=**

establishes the identification of one system in the CSE complex. When the system is defined for CSE LNKONLY, *systemname* must be one of the names specified in SYSTEM[1,2,3,4] keywords but *must not* match the system name specified in the SYSID macroinstruction.

Normally, the specification of all other CSExxx macroinstructions for every system in the same complex is identical. CSEID is the only operand that must be different, so that every system's CSEID is unique.

### *systemname*

is a 1- to 8-character name that identifies the hardware system where this CP nucleus with CSE active is to run, as it appears in the SYSID macroinstruction in HCPSYS.

Although more than one name can be specified in the SYSID macroinstruction of HCPSYS, for integrity reasons a CP nucleus with CSE active depends on an exact relationship of all systems in the complex. Because of this, cross system spooling and CSE commands are only active when executing on the hardware system specified in CSEID.

*systemname* must also be specified as the system name in one of the SYSTEM[1,2,3,4] keywords.

### **SYSTEM1,SYSTEM2,....,SYSTEM4**

define the individual systems in the CSE complex. These operands are referred to collectively as SYSTEM[1,2,3,4] operands. At least two systems, SYSTEM1 and SYSTEM2, must be specified. All SYSTEM[1,2,3,4] operands must be coded in the same order in the HCPSYS for every system in the CSE complex.

### *sysname1,sysname2,....,sysname4*

is a 1- to 8-character name that identifies each system to be included in the complex. One of these system names must also be specified as *systemname* in the CSEID operand. By default, these systems will be included in both cross system spooling and cross system link. However, the LNKONLY, LN2ONLY, NOLINK, and NOSPOOL operands allow additional control over which systems are to be included in, or excluded from, cross system link and cross system spooling.

System names should not be ALL or \*.

### *cvmname1,cvmname2,....,cvmname4*

specifies the 1- to 8-character user ID of the communication virtual machine (CVM) used by CSE. This operand is optional. This user ID defaults to PVM.

### *timeout1,timeout2,....,timeout4*

specifies the number of seconds that CSE is to wait for a response from another system. This waiting period is not associated with any particular transaction but applies to the overall status of the communication link. It comes into play only if this system has sent requests to another, and no response to any of those requests was received during the period specified. If the timeout period expires, any incomplete requests are terminated with an error. CSE assumes that the CVM link to that system (or the system itself) is down. The operator is notified of this event by a message.

No cross system spool processing can take place between these two systems until the connection is restored. However, cross system link processing continues. This operand is optional. The default is 12 seconds.

*alias1,alias2,...,alias4*

specifies a second 1- to 8-character name that can be used in CP commands instead of *sysname[1,2,3,4]* to identify the system. This operand is optional.

#### **NOSPOOL=**

specifies that the systems within the parentheses are excluded from sharing spool files with this system but continue to participate in cross system link and cross system commands. Each system name specified in NOSPOOL must also have been specified in one of the SYSTEM[1,2,3,4] operands.

*sysnamea*

specifies the 1- to 8-character name of a system to be excluded from sharing spool files.

It is not meaningful to code the name of the *main system* in NOSPOOL. (The *main system* is the system that was specified in the CSEID operand.) However, since it may be convenient to do so, no error message is issued if this is done.

#### **LNKONLY=**

specifies that the systems within the parentheses are included in cross system link only. None of these systems can share spool file access with any other system in the complex, nor can they send or receive CSE commands. LNKONLY, if coded, must appear after the required CSEID, SYSTEM1, and SYSTEM2 operands. Each system specified by LNKONLY is included in cross system link only if the system is not also specified on the NOLINK operand.

*sysnameb*

specifies the 1- to 8-character names of the systems to be included in cross system link only. This name must not be the same as any of the *sysname[1,2,3,4]* operands specified in the SYSTEM1 through SYSTEM4 keywords, since these are automatically included in cross system link unless also specified in a NOLINK statement. All systems with the LNKONLY parameter must be coded in the same order for all systems in the CSE complex. If they are not identical, multiple R/W access could be allowed for the shared DASD and may result in data loss.

**Note:** You cannot generate a complex of exclusively LNKONLY systems, since SYSTEM1 and SYSTEM2 must always be specified. When you generate HCPSYS for a system that operates in LNKONLY mode, use the *systemname* from one of the SYSTEM[1,2,3,4] keywords as the operand for CSEID.

#### **LN2ONLY=**

specifies that the systems within the parentheses are included in cross system link only. None of these systems can share spool file access with any other system in the complex, nor can they send or receive CSE commands. LN2ONLY performs the same operation as LNKONLY. LN2ONLY, if coded, must appear after the required CSEID, SYSTEM1, and SYSTEM2 operands. Each system specified by LN2ONLY is included in cross system link only if the system is not also specified on the NOLINK operand.

*sysnamec*

specifies the 1- to 8-character names of the systems to be included in cross system link only. This name must not be the same as any of the *sysname[1,2,3,4]* operands specified in the SYSTEM1 through SYSTEM4 keywords, since these are automatically included in cross system link unless also specified in a NOLINK statement. All systems with the LN2ONLY parameter must be coded in the same order for all systems in the CSE

complex. If they are not identical, multiple R/W access could be allowed for the shared DASD and may result in data loss.

**Note:** You cannot generate a complex of exclusively LN2ONLY systems, since SYSTEM1 and SYSTEM2 must always be specified. When you generate HCPSYS for a system that operates in LN2ONLY mode, use the *systemname* from one of the SYSTEM[1,2,3,4] keywords as the operand for CSEID.

### **NOLINK=**

specifies that the systems within the parentheses are excluded from cross system link and cannot access each other's minidisks by using cross system link. This has no effect on their participation in cross system spooling or CSE commands.

If a system name is specified in both a LNKONLY (or LN2ONLY or SYSTEM[1,2,3,4]) and a NOLINK parameter, the NOLINK specification overrides LNKONLY and LN2ONLY.

### *sysnamed*

specifies the 1- to 8-character names of systems to be excluded from cross system link. This name must be the same as one of the *sysname*[1,2,3,4] operands specified in the SYSTEM1 through SYSTEM4 keywords.

**Note:** Use of NOLINK is not compatible with the operation of DIRMAINT Release 5 within the complex, since DIRMAINT depends on the shared CMS minidisk protection provided by CSE.

## Usage Notes

1. CSESYS is an optional macro, coded in HCPSYS only if CSE is to be active.
2. If CSE is active, one of the SYSTEM[1,2,3,4] operands must be the same as the CSEID operand.
3. To include one or more systems in cross system link only, use LNKONLY, LN2ONLY, or both LNKONLY and LN2ONLY.
4. The maximum number of systems that can be defined for cross system link is 56 if the CSE complex uses only shared ECKD devices. If any shared CKD devices are used, then the number of systems supported depends on the device type. If more than 56 systems are specified for cross system link, HCPSYS ASSEMBLE does not assemble, and an MNOTE is issued.
5. If message 2890E is received when the system is IPL<sub>ED</sub> or when a volume is attached to the system, check that the CSE system and volume tables in HCPSYS are correct and that the CSE track is on the correct volume. If the system is running an earlier VM release than z/VM 1.1 and this is an ECKD device, enter XLINK FORMAT for the volume.
6. **Migration Incompatibility Note** — In previous releases of VM, you could specify different system IDs on the SYSID (page 677) and the CSESYS macroinstructions. If you decide to migrate your HCPSYS ASSEMBLE file to the z/VM Release 2.1 system configuration file, you can only have one system ID. To specify the system ID in the system configuration file, use the SYSTEM\_IDENTIFIER statement (page 226) or the SYSTEM\_IDENTIFIER\_DEFAULT statement (page 228). Use the system ID you define on these statements when you use the following system configuration file statements:



Statement	Function	Page
XLINK_SYSTEM_EXCLUDE	Tells CP which systems not to include in the cross system link and which systems CP should prevent from accessing each other's minidisks. (This statement has no effect on the system's participation in cross system spooling or cross system commands.)	263
XLINK_SYSTEM_INCLUDE	Tells CP which systems to include in the cross system link.	264
XSPPOOL_SYSTEM	Tells CP the names of the systems participating in cross system commands and spooling operations.	271

For more information about the system configuration file, see page Chapter 6, "The System Configuration File," on page 51.

## Examples

In this example of a three-system complex, two systems (KGNVHH and KGNVHB) are sharing fully, with the third system (KGNRMD) included only in minidisk sharing. SYSTEM1 is known as KGNVHH and SYSTEM2 as KGNVHB in the SYSID macroinstruction in the HCPSYS ASSEMBLE file and in the CSEID operand of the CSESYS macro. The system that is to be included only in minidisk sharing is defined as KGNRMD in the SYSID macroinstruction of the HCPSYS ASSEMBLE file. This LNKONLY system called KGNRMD may have a CSESYS macroinstruction that looks like this:

```
CSESYS CSEID=KGNVHH
      ,SYSTEM1=(KGNVHH.....
      ,SYSTEM2=(KGNVHB.....
      ,LNKONLY=(KGNRMD)
```

or

```
CSESYS CSEID=KGNVHB
      ,SYSTEM1=(KGNVHH.....
      ,SYSTEM2=(KGNVHB.....
      ,LNKONLY=(KGNRMD)
```

The CSESYS macroinstruction for system KGNVHH looks like:

```
CSESYS CSEID=KGNVHH
      ,SYSTEM1=(KGNVHH.....
      ,SYSTEM2=(KGNVHB.....
      ,LNKONLY=(KGNRMD)
```

The CSESYS macroinstruction for system KGNVHB looks like:

```
CSESYS CSEID=KGNVHB
      ,SYSTEM1=(KGNVHH.....
      ,SYSTEM2=(KGNVHB.....
      ,LNKONLY=(KGNRMD)
```

**Note:** You cannot generate a complex of exclusively LNKONLY and LN2ONLY systems, because SYSTEM1 and SYSTEM2 must always be specified.

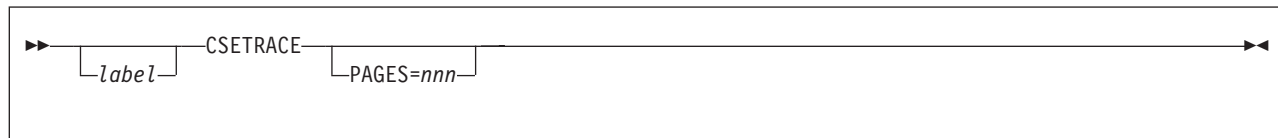
## Migration Aids

You can achieve the function of the CSESYS macroinstruction by using the following statements in a system configuration file:

- XLINK\_SYSTEM\_EXCLUDE (page 263)
- XLINK\_SYSTEM\_INCLUDE (page 264)
- XSPPOOL\_SYSTEM (page 271).

---

## CSETRACE (Optional)



### Purpose

Code the CSETRACE macroinstruction to establish the number of pages of CP storage allocated for the CSE trace tables. If CSETRACE is coded, it must appear in the HCPSYS ASSEMBLE file immediately after the CSESYS macro. If the CSETRACE macro is not specified, 4 pages are allocated.

### Parameters

#### PAGES

specifies the number of pages that CP is to allocate for the CSE trace tables when the system is initialized.

*nnn*

is an integer in the range of 0 to 255. If *nnn*=0, there are no pages allocated for CSE trace tables.

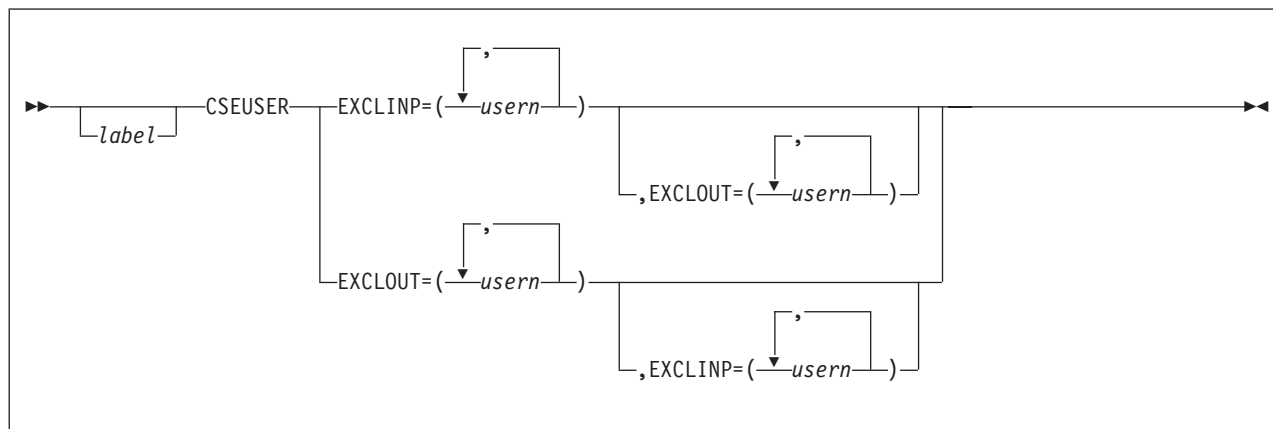
### Usage Notes

1. The first trace table page provides enough space to hold 16 synchronization requests and 111 CVM communication requests. Each additional page provides space for 127 more CVM communication requests. When this number is exceeded, the trace tables wrap around, and the previously recorded entries are overwritten.

### Migration Aids

You can achieve the function of the CSETRACE macroinstruction by using the XSPPOOL\_TRACE statement in a system configuration file. See “XSPPOOL\_TRACE Statement” on page 273.

## CSEUSER (Optional)



### Purpose

Code the CSEUSER macroinstruction to define virtual machines that will *not* participate in cross system spooling and cross system message and query commands. If CSEUSER macros are coded, they must appear after the CSESYS macro and before the SYSEND macro. Within ASSEMBLER limitations, any number of CSEUSER macros may be specified.

Input and output spool files owned by user IDs listed in the CSEUSER macro are not accessible to the other systems in the CSE complex.

### Parameters

#### EXCLINP

indicates that input (reader) spool files owned by the user IDs specified as *usern* are not to be accessible to the other systems in the CSE complex.

#### EXCLOUT

indicates that output (printer and punch) spool files owned by the user IDs specified as *usern* are not to be accessible to the other systems in the CSE complex.

#### *usern*

specifies the 1- to 8-character user ID of the virtual machine whose input or output files are to be excluded from access to cross system spooling. A maximum of 255 characters, including commas, are allowed in *usern* definitions.

### Usage Notes

1. User IDs listed in the CSEUSER macro have operational characteristics that are different from those that are not listed. The users associated with these user IDs:
  - Can log on to more than one system in the complex at the same time. Users whose user IDs are not listed here are prevented from doing so.
  - Receive messages generated by the CP MSG, MSGNOH, WNG, and SMSG commands, but only from users on the same system they are on, unless the user entering the CP command specified the AT parameter.

In addition, when these users:

## CSEUSER

- Enter a CP QUERY *userid* or QUERY NAMES command, only the system on which the command was entered is queried, unless the AT parameter was used.
  - Are the target of a CP QUERY *userid* command, only the system on which the command was entered is queried, unless the AT parameter was used.
2. These service virtual machines operate only in a single system within the CSE complex. CSE automatically excludes:
    - The system operator (SYSOPR macro in HCPSYS).
    - The communication virtual machine (CVM). See “Preparing the Communication Virtual Machine (CVM)” on page 350 for information on how to specify a CVM virtual machine.

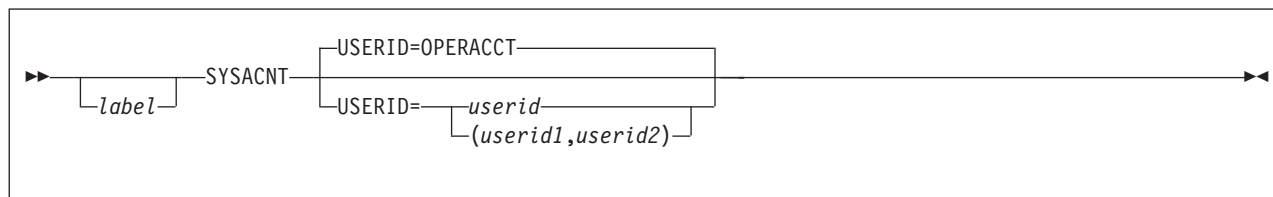
Do not code any CSEUSER macro for these virtual machines. Your installation may decide that other service virtual machines should also be excluded, but for ease of operation and administration of the CSE complex, keep exclusions to a minimum.
  3. The CSEUSER EXCLINP list for every system in the CSE complex must be the same; the CSEUSER EXCLOUT list for every system in the CSE complex must be the same.

## Migration Aids

You can achieve the function of the CSEUSER macroinstruction by using the following statements in a system configuration file:

- XSPPOOL\_XLIST\_INPUT—See “XSPPOOL\_XLIST\_INPUT Statement” on page 274.
- XSPPOOL\_XLIST\_OUTPUT—See “XSPPOOL\_XLIST\_OUTPUT Statement” on page 276.

## SYSACNT (Optional)



### Purpose

Code the SYSACNT macroinstruction to specify user IDs of up to two virtual machines for which the CP accounting system service (\*ACCOUNT) is to accumulate records. When the virtual machines thus identified to CP are connected to \*ACCOUNT, accounting records are sent to them.

In addition, these virtual machines are automatically logged on during CP initialization.

The SYSACNT macroinstruction is optional; if you code it, you must place it after the first five required macroinstructions in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SEND macroinstruction. If this macro is not coded, CP processes the virtual machine having the user ID OPERACCT as the accounting record recording virtual machine.

**Note:** Any virtual machine identified on the SYSACNT macro *must* contain an IUCV directory control statement that authorizes the application running in the virtual machine to connect to the \*ACCOUNT system service. Failure to supply this required authority results in the failure of the connection request from the virtual machine. If the RETRIEVE utility is used to collect accounting records and this authority is not coded in the directory, the RETRIEVE utility will fail.

### Parameters

**USERID=OPERACCT**

**USERID=userid**

**USERID=(userid1,userid2)**

specifies the user IDs of the virtual machines for which the \*ACCOUNT system service is to accumulate and checkpoint accounting records. Either one or two user IDs may be specified. The user IDs are alphanumeric character strings of up to 8 characters. OPERACCT is the default user ID.

### Usage Notes

1. If the user ID specified on the SYSACNT macro has an entry in the user directory, that user ID is logged on at system initialization.

### Examples

1. To specify OPERACCT as the user ID of the accounting virtual machine, you can do one of the following:
  - Omit the SYSACNT macro
  - Code the SYSACNT macro as follows:
 

```
SYSACNT USERID=OPERACCT
```

## **SYSACNT**

2. To specify DISKACNT as the user ID of the accounting virtual machine, code the SYSACNT macro as follows:

```
SYSACNT USERID=DISKACNT
```

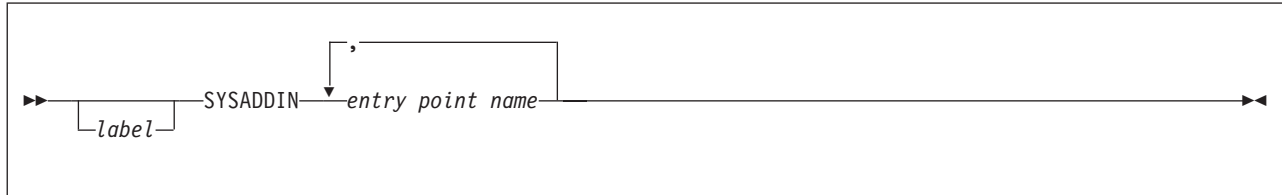
3. To specify OPERACCT and DISKACNT as the user IDs of the accounting record recording virtual machines, code the SYSACNT macro as follows:

```
SYSACNT USERID=(OPERACCT,DISKACNT)
```

## **Migration Aids**

You can achieve the function of the SYSACNT macroinstruction by using the SYSTEM\_USERIDS statement in a system configuration file. See “SYSTEM\_USERIDS Statement” on page 232.

## SYSADDIN (Optional)



### Purpose

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

The SYSADDIN macroinstruction is optional; if you code it, you must place it after the first five required macroinstructions in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SYSEND or CSESYS macroinstruction.

### Parameters

#### *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

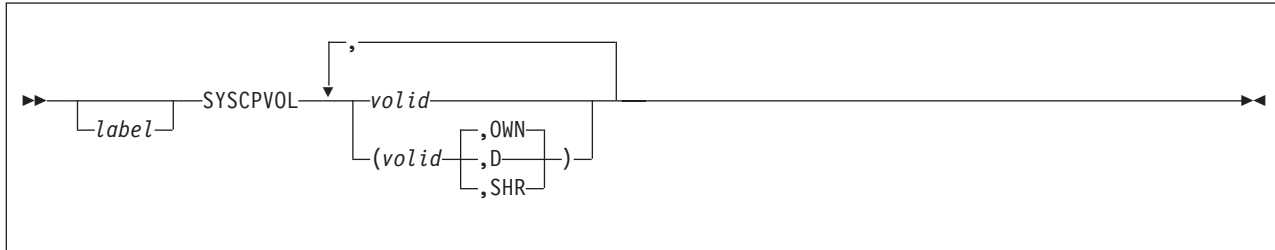
To specify LOCALMOD, USERMOD, and XMOD as entry points to be started during system initialization code the SYSADDIN macro as follows:

```
SYSADDIN LOCALMOD,USERMOD,XMOD
```

### Migration Aids

You can achieve the function of the SYSADDIN macroinstruction by using the CP\_ADDON\_INITIALIZE\_ROUTINES statement in a system configuration file. See “CP\_ADDON\_INITIALIZE\_ROUTINES Statement” on page 72.

## SYSCPVOL (Required)



### Purpose

Code the SYSCPVOL macroinstruction to define and generate a list of up to 255 CP-owned DASD volumes. CP-owned volumes are the CP system residence volume and volumes that contain real system paging, spooling, dump, directory, or temporary disk space. SYSCPVOL is required and must appear as one of the first five macroinstructions in the HCPSYS ASSEMBLE file.

When CSE is not used, all the volumes listed in SYSCPVOL containing SPOL space may be used to read and write spool files. (Spool files are written only once, at the time they are created.)

When CSE is active, shared access to spool files for all systems in the complex is provided by physically sharing the spooling DASD. Although physically this is both read and write sharing, logically the sharing is on a read-only basis. This means that some volumes must be identified as available for writing the spool files that are created on this system, and others must be identified as available only for reading spool files created on other systems. The SYSCPVOL macro allows this by specifying the SHR and OWN operands.

**Note:** It is recommended that the system residence volumes and paging volumes not be shared among systems in a CSE complex.

### Parameters

#### *valid*

specifies the 1- to 6-character volume identifier. All *valids* with SPOL space must be specified in exactly the same position in the SYSCPVOL lists for all systems in the CSE complex. An example follows, but it does not define as many volumes as would be expected in a production system.

**D** indicates that the spool space on volume *valid* is to be reserved exclusively for dumps.

#### **SHR**

indicates that the identified volume is owned by another system in the CSE complex. This system cannot create or destroy spool files on it. It can, however, read spool files previously written by the owning system.

#### **OWN**

indicates that the volume identified is owned by this system (the system you are generating) and that only this system can create or destroy spool files on this volume. The other systems in the CSE complex, which have *valid* coded with the SHR parameter, have shared access to this system's spool files because they can read the data on this volume. OWN is the default and need not be specified.



## Usage Notes

1. Before CP can use a DASD, you must format, label, and allocate space on the DASD. You may do this by using the Device Support Facilities program (ICKDSF) or the CMS utility CPFMTXA. However, ICKDSF is the recommended method of CP volume maintenance and is required for certain DASD types. ECKD capable CKD devices should be formatted without filler records using ICKDSF. For more information on the ICKDSF program, see the *Device Support Facilities User's Guide and Reference* book. For more information on the CPFMTXA utility, see the *z/VM: CMS Commands and Utilities Reference* book.
2. You must specify the system residence volume identification on the SYSCPVOL macroinstruction. You must also identify, as CP-owned, volumes that contain checkpoint and warm-start data, paging, spooling, dump, directory, or temporary disk space. You can identify all other volumes by coding the SYSUVOL macroinstruction or by logically attaching (ATTACH command) the entire device.
3. If a volume is specified in the SYSCPVOL statement but is not mounted when CP is loaded, CP considers that volume unavailable. CP continues processing, if possible. The system operator may mount and attach the volume (using the class A ATTACH command) to the system when it is needed. This provides extra space for future growth.
4. If you want to add new volumes to the system after system generation, add new volume identifications to the end of the SYSCPVOL list. Then reassemble HCPSYS ASSEMBLE, rebuild the CP nucleus, and reload CP (see the generation procedures in *z/VM: Guide for Automated Installation and Service*). This procedure allows you to warm start the system without disturbing the relative entry number that locates system spool buffers.

**Attention:** If you delete any volume that contains spool space from the SYSCPVOL list, or any volume that contains spool space is moved to a different position in the SYSCPVOL list, a clean start is required. A clean start causes deletion of spool files and system data files, including named saved systems and saved segments. Files that are 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 SYSCPVOL list are restored correctly.

5. If you specify D for a volume and the spool space on that volume is already in use for spool files when the system is IPLed, those spool files remain on that volume until an SPXTAPE or SPOOL command moves them.
6. For a singlez/VM system, that is, one that is not associated with other systems by means of CSE, all CP volumes should be owned by that system, and SHR should not be specified. Since OWN is the default, no change to the traditional coding of SYSCPVOL is required.
7. The following notes apply only when CSE is in use:
  - a. All volumes that have SPOL extents must be included in the SYSCPVOL macro for each system, with the same relative position in each system's SYSCPVOL macro. It is suggested that all volumes with SPOL extents be grouped by system and placed first in the SYSCPVOL list.
  - b. If D is specified for *valid*, CSE treats *valid* the same way as if OWN were specified; only this system may allocate spool space (for dumps) on this volume.

The variable *valid* must be coded with the SHR parameter in the SYSCPVOL lists of all the other systems in the CSE complex.

## SYSCPVOL

- c. In the SYSCPVOL list, you cannot change the order of the volumes that contain SPOL space unless a clean start is performed on all systems in the complex. If you do change the order, you may have to use SPXTAPE to avoid losing spool files. However, if you add new SPOL space volumes to the end of the SYSCPVOL lists of the associated systems without disturbing the position and sequence of existing volumes, a clean start is not required.
8. At system initialization (IPL), if more than one DASD volume has the same volume serial (*valid*) and that *valid* is in the CP or user volume list, the volume with the lowest device number is attached to the system. This rule does not apply to duplicates of the system residence (IPLed) volume.
9. The list of CP-owned volumes generated by CP\_OWNED statements in the system configuration file completely supersedes the list of CP-owned volumes generated by the SYSCPVOL macroinstruction.

## Examples

1. To designate the spool space on volume CPDSK1 for dumps only, code the SYSCPVOL macro as follows:  
SYSCPVOL CPDRM1, (CPDSK1,D), CPDSK2
2. To designate spool space on volumes to be used for CSE code the SYSCPVOL macro as follows:

SYSCPVOL list for system KGNRMD1:

```
SYSCPVOL RD1RES, (SPOOL1,OWN),  
            (SPOOL2,SHR), (SPOOL3,SHR),  
            (SPOOL4,OWN), (SPOOL5,SHR),  
            (SPOOL6,SHR), (SPOOL7,OWN),  
            (SPOOL8,SHR), (SPOOL9,SHR)
```

SYSCPVOL list for system KGNRMD2:

```
SYSCPVOL RD2RES, (SPOOL1,SHR),  
            (SPOOL2,OWN), (SPOOL3,SHR),  
            (SPOOL4,SHR), (SPOOL5,OWN),  
            (SPOOL6,SHR), (SPOOL7,SHR),  
            (SPOOL8,OWN), (SPOOL9,SHR)
```

SYSCPVOL list for system KGNRMD3:

```
SYSCPVOL RD3RES, (SPOOL1,SHR),  
            (SPOOL2,SHR), (SPOOL3,OWN),  
            (SPOOL4,SHR), (SPOOL5,SHR),  
            (SPOOL6,OWN), (SPOOL7,SHR),  
            (SPOOL8,SHR), (SPOOL9,OWN)
```

### 5.Explanation

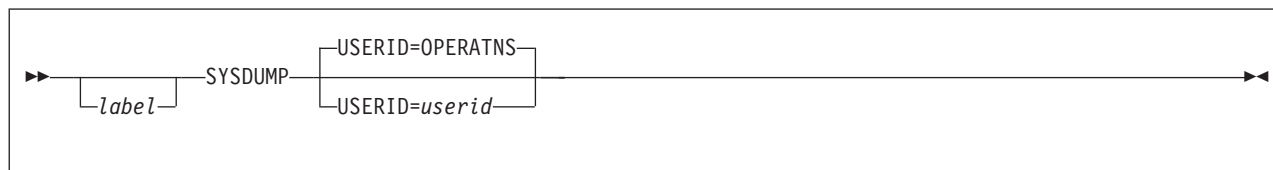
Each system owns three volumes that contain CP spooling space. The position of the spooling volumes is the same in each list. The only nonspooling volumes listed are the system residence volumes RD1RES, RD2RES, and RD3RES. These are three different volumes, but they occupy the same position in each list in order to keep the position of the spooling volumes constant. Rather than allow an entry to default to OWN, it has been explicitly stated in each case. This makes visual checking of the lists less difficult.

**Note:** System residence volume IDs should be unique.

## Migration Aids

You can achieve the function of the SYSCPVOL macroinstruction by using the CP\_OWNED statement in a system configuration file. See “CP\_OWNED Statement” on page 73.

## SYSDUMP (Optional)



### Purpose

Code the SYSDUMP macroinstruction to identify the user ID of the virtual machine that receives system dump files. The SYSDUMP macroinstruction is optional; if you code it, you must place it after the first five required macroinstructions in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SYSEND macroinstruction. If this macro is not coded, the virtual machine having the user ID OPERATNS receives the system dump files.

### Parameters

#### **USERID=OPERATNS**

#### **USERID=userid**

is the user ID of the virtual machine that receives spooled system dumps. The variable *userid* must be an alphanumeric character string of up to 8 characters long. OPERATNS is the default user ID.

### Usage Notes

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 SYSDUMP macroinstruction.

### Examples

- To specify OPERATNS as the user ID of the virtual machine that receives system dump files you can do one of the following:
  - Omit the SYSDUMP macro
  - Code the SYSDUMP macro with no parameters
  - Code the SYSDUMP macro as follows:
 

```
SYSDUMP USERID=OPERATNS
```
- To specify MRDUMP as the user ID of the virtual machine that receives system dump files, code the SYSDUMP macro as follows:
 

```
SYSDUMP USERID=MRDUMP
```

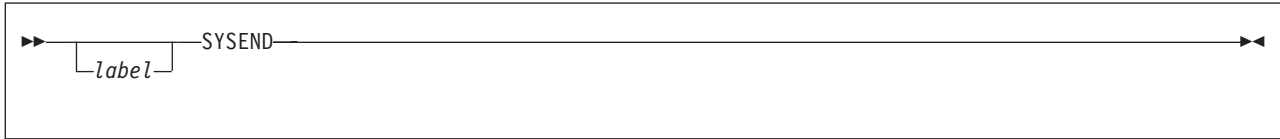
### Migration Aids

You can achieve the function of the SYSDUMP macroinstruction by using the SYSTEM\_USERIDS statement in a system configuration file. See “SYSTEM\_USERIDS Statement” on page 232.

## SYSEND

---

### SYSEND (Required)



### Purpose

Use the SYSEND macroinstruction to end the HCPSYS assembly. The SYSEND macroinstruction is required and must appear as the last macroinstruction in HCPSYS ASSEMBLE.

### Examples

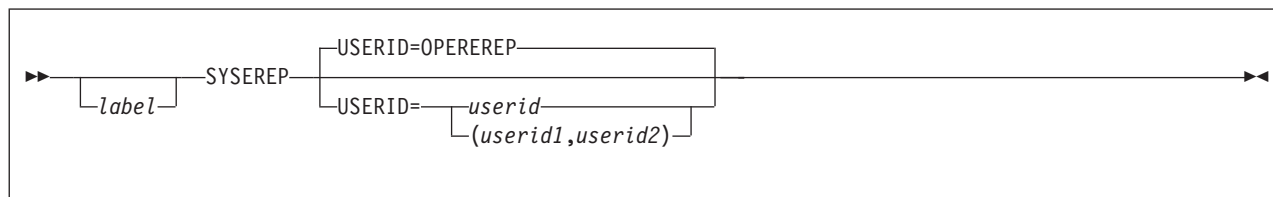
To end the HCPSYS assembly, code the SYSEND macro as follows:

```
SYSEND
```

### Migration Aids

There is no equivalent statement in a system configuration file for the SYSEND macroinstruction.

## SYSEREP (Optional)



### Purpose

Code the SYSEREP macroinstruction to identify the user IDs of one or two virtual machines for which the \*LOGREC system service is to accumulate records. When the virtual machines thus identified to CP connect to the \*LOGREC system service, EREP records are sent to them.

These virtual machines are automatically logged on during CP initialization.

The SYSEREP macroinstruction is optional; if you code it, you must place it after the first five required macroinstructions in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SYSEND macroinstruction. If this macro is not coded, CP processes the virtual machine having the user ID OPEREREP as the error recording virtual machine.

**Note:** Any virtual machine identified on the SYSEREP macro *must* contain an IUCV directory control statement authorizing the application running in the virtual machine to connect to the \*LOGREC system service. Failure to supply this required authorization will result in the failure of the connection request from the virtual machine. If the RETRIEVE utility is used to collect error records and this authorization is not coded in the directory, RETRIEVE will fail.

### Parameters

**USERID=OPEREREP**

**USERID=userid**

**USERID=(userid1,userid2)**

are the user IDs of the virtual machines for which the \*LOGREC system service is to accumulate and checkpoint error records. One or two user IDs can be specified. The variable *userid* must be an alphanumeric character string of up to 8 characters. OPEREREP is the default user ID.

### Usage Notes

If the user ID specified on the SYSEREP macroinstruction has an entry in the user directory at system initialization, the associated virtual machine is logged on to the system.

### Examples

- To specify OPEREREP as the user ID of the error recording virtual machine, you can do one of the following:
  - Omit the SYSEREP macro
  - Code the SYSEREP macro as follows:
 

```
SYSEREP USERID=OPEREREP
```

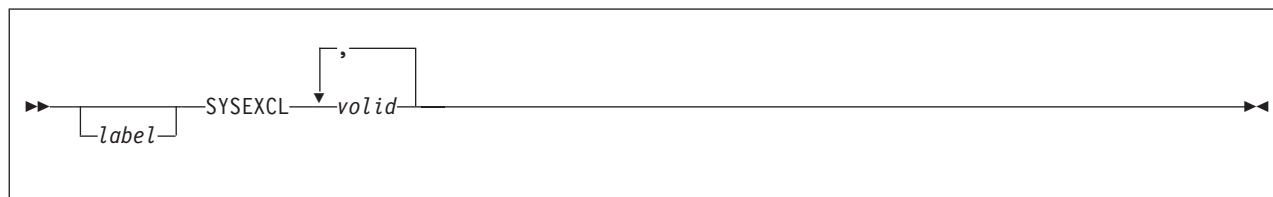
## **YSEREP**

2. To specify MISSEREP as the user ID of the error recording virtual machine, code the YSEREP macro as follows:  
YSEREP USERID=MISSEREP
3. To specify OPEREREP and MISSEREP as the user IDs of the error recording virtual machines, code the YSEREP macro as follows:  
YSEREP USERID=(OPEREREP,MISSEREP)

## **Migration Aids**

You can achieve the function of the YSEREP macroinstruction by using the SYSTEM\_USERIDS statement in a system configuration file. See “SYSTEM\_USERIDS Statement” on page 232.

## SYSEXCL (Optional)



### Purpose

Code the SYSEXCL macroinstruction 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 volume specified on the SYSUVOL macro and any volume whose volume identifier (*valid*) matches a generic volume identifier specified on the SYSINCL macro, unless the volume is also specified on the SYSEXCL macro.

The SYSEXCL macroinstruction is optional; if you code it, you must place it after the first five required macroinstructions in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SYSEND or CSESYS macroinstruction.

### Parameters

*valid*

is the user volume identification. The variable *valid* is a 1- to 6-character identifier. Each character can be a letter, number, or hyphen.

The variable *valid* can also be specified as a generic volume identifier. A generic volume identifier is specified as 1 to 5 characters followed by an asterisk (\*). For example, SYSEXCL EM\* excludes all volumes whose volume identifiers begin with the letters “EM”.

### Usage Notes

1. If a volume identifier matches a *valid* pattern specified on both the SYSINCL and SYSEXCL macros, CP does *not* attach the volume to the system during initialization.
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 code a SYSINCL macro, you may want to specify dedicated devices on the SYSEXCL macro.

### Examples

1. To define as user volumes all volumes except VMTST1 and VMTST2, code the following macros:
 

```
SYSINCL *
SYSEXCL VMTST1,VMTST2
```
2. To define as user volumes all volumes whose identifiers begin with “US” except for any whose identifiers begin with “USER”, code the following macros:
 

```
SYSINCL US*
SYSEXCL USER*
```

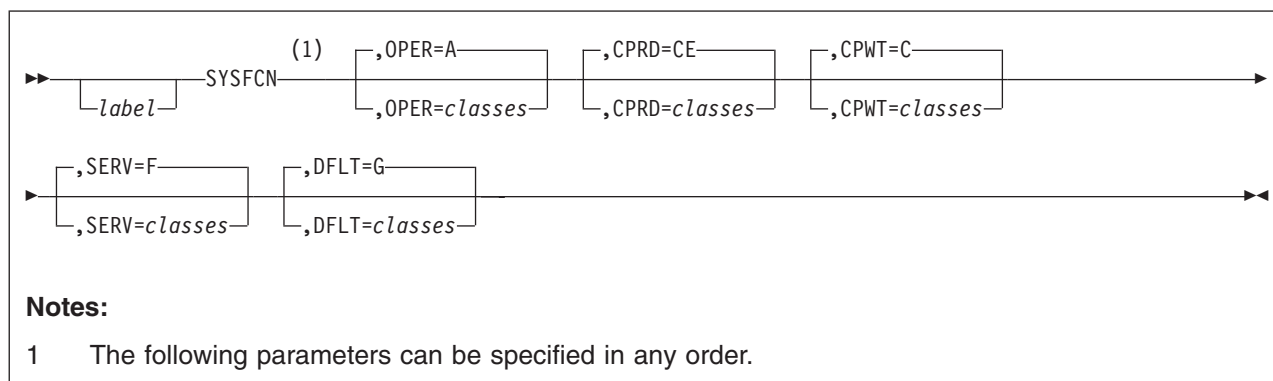
## SYSEXCL

### Migration Aids

You can achieve the function of the SYSEXCL macroinstruction by using the USER\_VOLUME\_EXCLUDE statement in a system configuration file. See “USER\_VOLUME\_EXCLUDE Statement” on page 251.



## SYSFCN (Optional)



## Purpose

Code the SYSFCN macroinstruction to change the privilege class 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.

A default SYSFCN macro is supplied in the sample HCPSYS ASSEMBLE file. If you want to change some or all of the privilege classes assigned to these CP functions, you must include a SYSFCN macro statement in HCPSYS that specifies the changes you want to make.

The SYSFCN macroinstruction is optional; if you code it, you must place it after the first five required macroinstructions in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SYSEND macroinstruction. If SYSFCN is not coded, all the default privilege classes are used.

## Parameters

**OPER=classes**

**OPER=A**

specifies the classes for the primary system operator. The default is A.

**CPRD=classes**

**CPRD=CE**

specifies the classes authorized to issue IOCP READ. The default is CE.

**CPWT=classes**

**CPWT=C**

specifies the classes authorized to issue IOCP WRITE. The default is C.

**SERV=classes**

**SERV=F**

specifies the classes authorized to perform intensive error recording. The default is F.

**DFLT=classes**

## SYSFCN

### **DFLT=G**

specifies the default classes for users who do not have a class defined in their directory entries. The default is G.

## Usage Notes

Coding the OPER=*classes* operand allows any user ID with any of the specified classes to become the system operator ID if the primary operator logs off.

## Examples

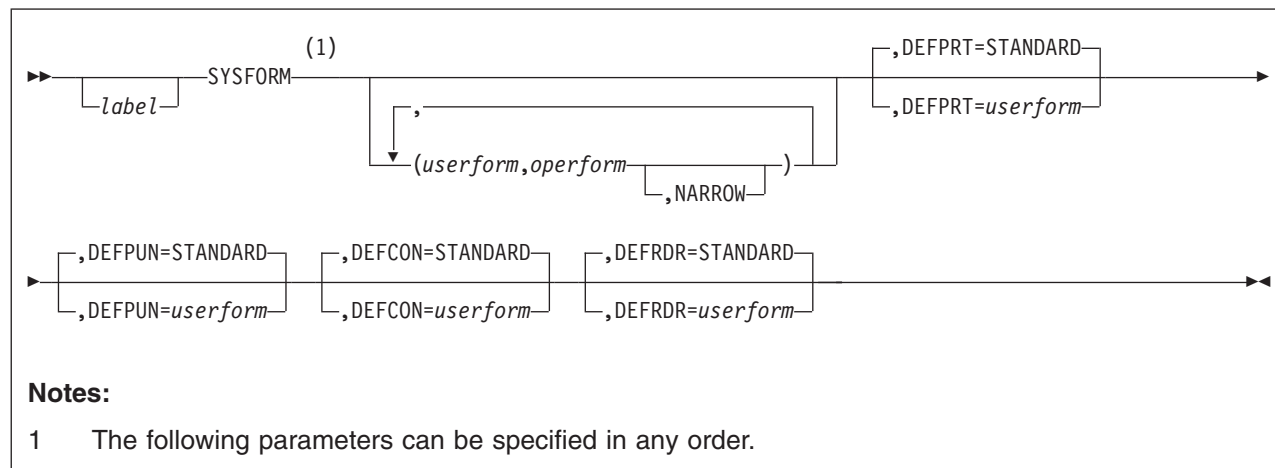
To specify the class of the system operator as ABD, code the SYSFCN macro as follows:

```
SYSFCN OPER=ABD
```

## Migration Aids

You can achieve the function of the SYSFCN macroinstruction by using the PRIV\_CLASSES statement in a system configuration file. See “PRIV\_CLASSES Statement” on page 184.

## SYSFORM (Optional)



### Purpose

Code the SYSFORM macroinstruction to create a list of user form names and their corresponding operator form numbers. You can also use SYSFORM to (a) specify forms as NARROW, so that a narrow separator page is printed and (b) generate default user form names to be used when files are created on virtual printers, virtual punches, virtual consoles, or real card readers.

The SYSFORM macroinstruction is optional; if you code it, you must place it after the first five required macroinstructions in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SYSEND macroinstruction.

### Parameters

*userform*  
 is a 1- to 8-character user form name.

*operform*  
 is a 1- to 8-character operator form number.

**NARROW**  
 specifies that the form be 86 columns wide. If you specify NARROW, separator information does not print beyond the 86th column.

**DEFPRT**  
 specifies the default user form for virtual printer spool files. If DEFPRT is not specified, the default is DEFPRT=STANDARD.

**DEFPUN**  
 specifies the default user form for virtual punch spool files. If DEFPUN is not specified, the default is DEFPUN=STANDARD.

**DEFCON**  
 specifies the default user (and, implicitly, operator) forms for virtual console spool files. If DEFCON is not specified, the default is DEFCON=STANDARD.

**DEFRDR**  
 specifies the default user (and, implicitly, operator) forms for files created on a real card reader. If DEFRDR is not specified, the default is DEFRDR=STANDARD.

## SYSFORM

### Usage Notes

The *userform* and *operform* operands establish the correspondence between the user form and operator form. If no user form and operator form pairs are specified, no form list is generated, and CP makes no distinction between user forms and operator forms.

### Examples

1. To use CP's default values you can do one of the following:

- Omit the SYSFORM macro from HCPSYS
- Code the SYSFORM macro with no operands.

CP assigns default forms of STANDARD for the printer, punch, console, and card reader.

2. You can specify the following default forms:

- LISTING for virtual printer files and real card reader files
- PLAIN for virtual punch files.

To do this, code the SYSFORM macro as follows:

```
SYSFORM DEFprt=LISTING,DEFpun=PLAIN,DEFrdr=LISTING
```

3. To specify the same default values as in the previous example:

- a. Create a list of user form names (for example, LISTING, DOCUMENT, and PLAIN) and the corresponding operator form numbers (G1PW and G1PN).
- b. Specify that G1PN is to be no more than 86 columns wide.
- c. Code the SYSFORM macro as follows:

```
SYSFORM (LISTING,G1PW), *  
        (DOCUMENT,G1PN,NARROW), *  
        (PLAIN,G1PN,NARROW), *  
        DEFprt=LISTING, *  
        DEFpun=PLAIN, *  
        DEFrdr=LISTING
```

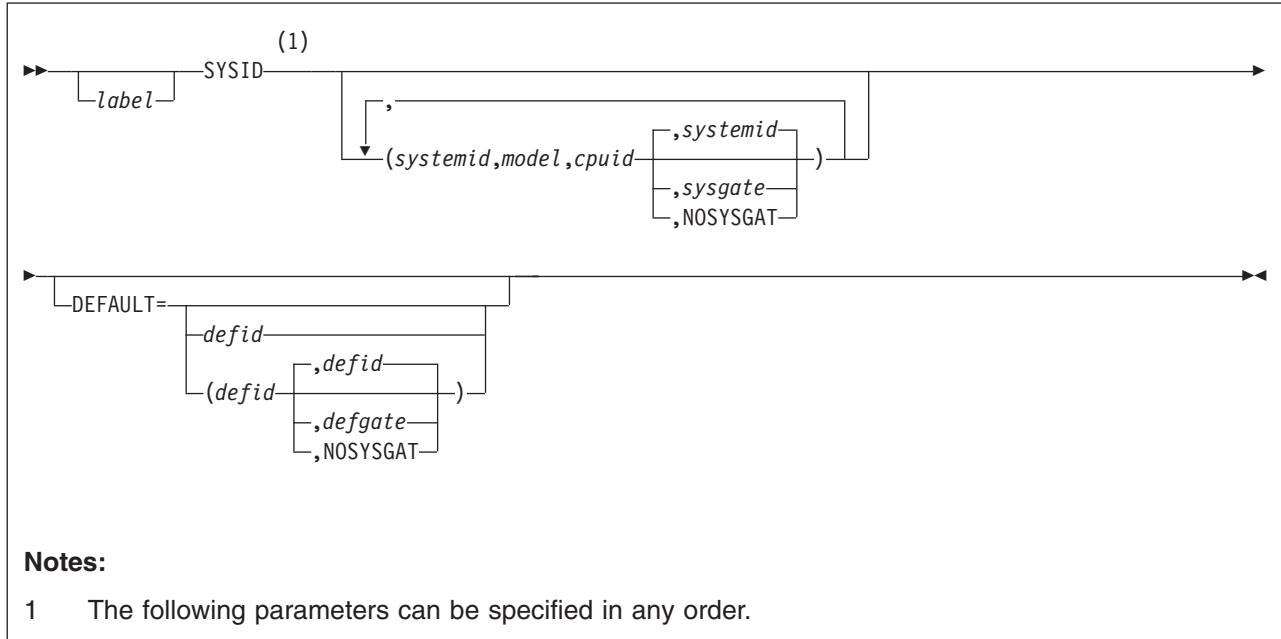
Note that if you need more than one line to code the SYSFORM macro, you *must* code a continuation character in column 72 *and* start the next line in column 16.

### Migration Aids

You can achieve the function of the SYSFORM macroinstruction by using the following statements in a system configuration file:

- FORM\_DEFAULT—See “FORM\_DEFAULT Statement” on page 150.
- USERFORM—See “USERFORM Statement” on page 247.

## SYSID (Optional)



## Purpose

Code the SYSID macroinstruction to create a system identifier for the processor on which you run z/VM. If your installation has several processors, you can create system identifiers for each one. At CP initialization, CP matches the processor model and CPU identification numbers with your system identifier. Then, the system identifier appears on printed output separator pages and in the status area of 3270 display screens.

The SYSID macro 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. See the *z/VM: Connectivity* book for more information about gateways.

By default, the system gateway name is the same as the system ID for the system; this is the recommended naming convention. However, if this default conflicts with another gateway name, you can use SYSID to specify a unique system gateway name for the system.

The SYSID macroinstruction is optional. If you code it, you must place it after the first five required macroinstructions in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SYSEND macroinstruction.

**Note:** If you do not code the SYSID macroinstruction, the z/VM system will not have a system identifier or system gateway name. Also, if the processor model and CPU identification number do not match any SYSID entries and the DEFAULT operand is not coded, a system identifier or system gateway will not be identified.

## Parameters

### *systemid*

is the 1- to 8-character system identification.

### *model*

is a 3- or 4-digit hexadecimal number that identifies the processor unit model number.

### *cpuid*

is a 6-digit hexadecimal identification number of the processor in the processor complex. If you specify a 5-digit number, the high-order digit is set to 0. Only 5- or 6- digit numbers are accepted.

### *sysgate*

is the 1- to 8-character name of the system gateway that will be identified when this system initializes. The *sysgate* name is optional; if not specified, the system gateway name defaults to the *systemid* for the system. ISFC will then use this value as its node name within a CS collection.

If you specify NOSYSGAT as the *sysgate* value, a system gateway name is not identified for the system. Also, the specified system cannot join a CS collection.

### **DEFAULT=**

identifies the defaults that are used if VM is initialized on a processor that is not listed in the SYSID macroinstruction.

### *defid*

is the one- to eight-character default system identification. CP uses the *defid* value when VM is loaded on a system whose model and CPU identification number are unknown. If you do not code the DEFAULT operand, VM will not have a default system identifier.

### *defgate*

is the one- to eight-character default name of the system gateway. If you do not specify a *defgate* value, the specified *defid* value is used as the default system gateway name.

If you specify NOSYSGAT as the *defgate* value, a system gateway name is not identified for the system. Also, the specified system cannot join a CS collection.

## Usage Notes

1. **Migration Incompatibility Note** — In previous releases of VM, you could specify different system IDs on the CSESYS (page 653) and the SYSID macros. If you migrate your HCPSYS ASSEMBLE file to the system configuration file, you can only have one system ID.

To specify the system ID in the system configuration file, use the SYSTEM\_IDENTIFIER statement (page 226) or the SYSTEM\_IDENTIFIER\_DEFAULT statement (page 228).

Use the system ID you define on these statements when you use the following system configuration file statements:

Statement	Function	Page
XLINK_SYSTEM_EXCLUDE	Tells CP which systems not to include in the cross system link and which systems CP should prevent from accessing each other's minidisks. (This statement has no effect on the system's participation in cross system spooling or cross system commands.)	263

Statement	Function	Page
XLINK_SYSTEM_INCLUDE	Tells CP which systems to include in the cross system link.	264
XSPOOL_SYSTEM	Tells CP the names of the systems participating in cross system commands and spooling operations.	271

For more information about the system configuration file, see page Chapter 6, “The System Configuration File,” on page 51.

## Examples

1. If your installation has one processor and you want to specify ONLYSYS as the system identifier for printer separator pages and in the system screen logo, code the SYSID macro as follows:

```
SYSID DEFAULT=ONLYSYS
```

The system gateway for this system will also be identified as ONLYSYS.

2. As the following example shows, you can specify values for several systems when coding the SYSID macro:

```
SYSID (SYSTEM1,2064,073499,SYS1),      *
      (SYSTEM2,2064,123664),          *
      (SYSTEM3,2064,123512,NOSYSGAT), *
      DEFAULT=(OTHERSYS,OTHERGAT)
```

When SYSTEM1 initializes (processor model 2064, identification number 073499), the system gateway SYS1 will be identified. Because a *sysgate* value is not specified for SYSTEM2, the system gateway name for this system will also be identified as SYSTEM2. On SYSTEM3, the NOSYSGAT value indicates that a system gateway will not be identified when this system initializes. Because the DEFAULT parameter was coded, any processor not specified on SYSID will be identified as system OTHERSYS; its system gateway will be identified as OTHERGAT.

**Note:** If you need more than one line to code the SYSID macro, you *must* code a continuation character in column 72 *and* start the next line in column 16.

## Migration Aids

You can achieve the function of the SYSID macroinstruction by using the following statements in a system configuration file:

- SYSTEM\_IDENTIFIER—See “SYSTEM\_IDENTIFIER Statement” on page 226.
- SYSTEM\_IDENTIFIER\_DEFAULT—See “SYSTEM\_IDENTIFIER\_DEFAULT Statement” on page 228.

---

## SYSINCL (Optional)



### Purpose

Code the SYSINCL macroinstruction to define user volumes by means of 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 identifier matches a generic volume identifier specified in the SYSINCL list, but will not use them for paging, spooling, directory or temporary disk space. A volume must be attached to the system before users can link to minidisks it contains.

The SYSINCL macroinstruction is optional; if you code it, you must place it after the first five required macroinstructions in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SYSEND or CSESYS macroinstruction.

### Parameters

- v is the generic volume identifier. The variable v consists of 0 to 5 characters. Each character can be a letter, number, or hyphen. The variable v must be followed by an asterisk. A single asterisk (\*) attaches all volumes to the system.

### Usage Notes

1. You can specify volumes that you do *not* want to define as user volumes by specifying them on the SYSEXCL macro (see page 671).
2. If a volume identifier matches a generic volume identifier specified on both the SYSINCL and SYSEXCL macros, CP does *not* attach the volume to the system during initialization.
3. If a volume is normally attached to the system through the SYSINCL list, CP does not notify the operator if the volume is not mounted. If a user volume is necessary for normal system operation, specify it on the SYSUVOL macro so that the operator is notified during system initialization that the volume is not mounted.

### Examples

1. To define as user volumes all volumes not specified by the SYSEXCL macro, so that CP attaches them to the system during initialization, code the SYSINCL macro as follows:
 

```
SYSINCL *
```
2. To define as user volumes all volumes whose volume identifiers begin with either "MVS", "USER", or "CMS", so that CP attaches them to the system during initialization, code the SYSINCL macro as follows:
 

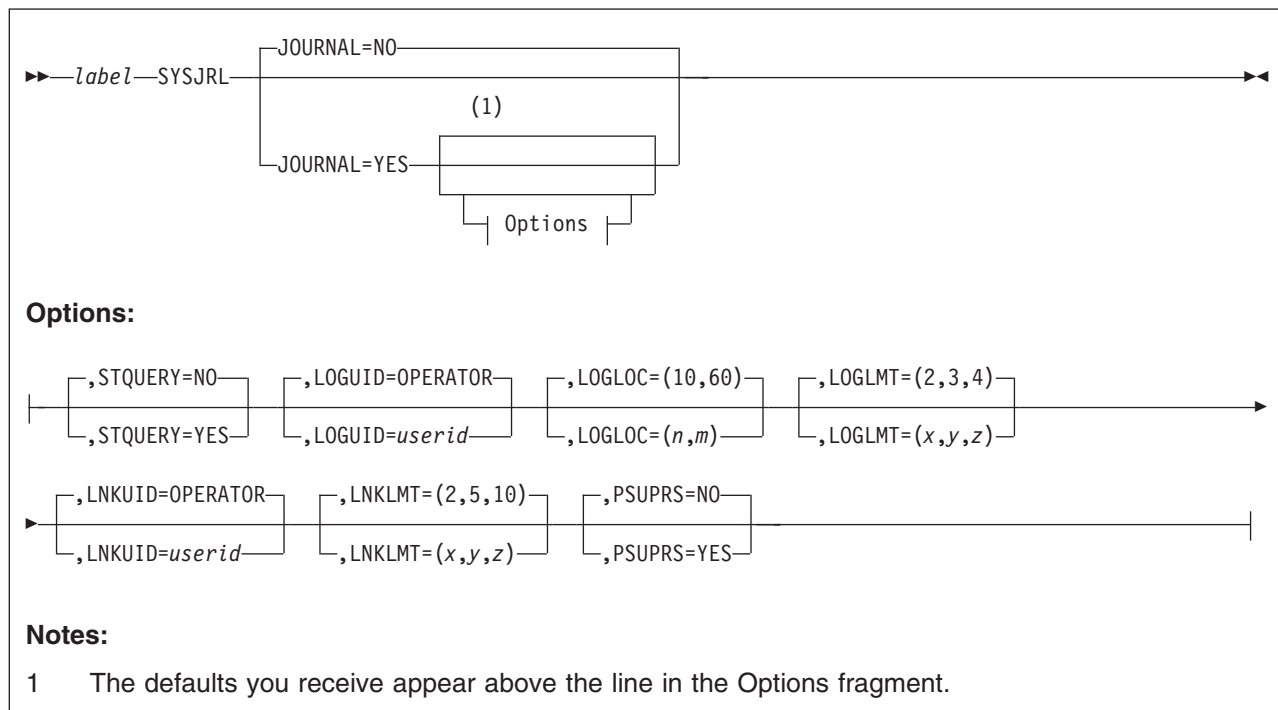
```
SYSINCL MVS*,USER*,CMS*
```



## Migration Aids

You can achieve the function of the SYSINCL macroinstruction by using the USER\_VOLUME\_INCLUDE statement in a system configuration file. See “USER\_VOLUME\_INCLUDE Statement” on page 253.

## SYSJRL (Optional)



## Purpose

Code the SYSJRL macroinstruction to specify the inclusion of the journaling or password suppression facility, or both.

The SYSJRL macroinstruction is optional; if you code it, you must place it after the first five required macroinstructions in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SYSEND macroinstruction. If the SYSJRL macro is not coded, all the default values are used.

## Parameters

### JOURNAL=NO JOURNAL=YES

specifies whether the journaling facility is to be operative in the system being generated.

### STQUERY=NO STQUERY=YES

specifies whether the ability to use the SET and QUERY commands to set and query the journaling function should be part of the system being generated. STQUERY=YES can be specified only if JOURNAL=YES is also specified.

### LOGUID=OPERATOR LOGUID=userid

specifies the user ID that should receive the indication that an invalid logon password count has been reached or exceeded. If this user ID is disconnected or logged off, the system operator receives the message generated.

### LOGLOC=(10,60)

**LOGLOC=(*n,m*)**

specifies the maximum number of invalid password attempts and the delay time until another logon is allowed. The variable *n* is the maximum number of invalid password attempts. The value of *n* may range from 1 to 255. The default is 10 logon attempts.

The variable *m* is the delay time in minutes until the next logon attempt. The value of *m* may range from 0 to 255. The default is 60 minutes.

If anyone makes more than *n* unsuccessful attempts to log on to a given user ID or terminal, an error message is issued. Nobody can log on to that user ID, or at the terminal where the attempts were made, until *m* minutes have passed. Anyone who tries receives an error message.

**LOGLMT=(2,3,4)****LOGLMT=(*x,y,z*)**

specifies the invalid LOGON/AUTOLOG/XAUTOLOG threshold. The values specified apply to all unsuccessful logon attempts at a single terminal and for a single user ID.

The variable *x* is the value that when reached causes CP to generate a type 04 accounting record for that and each subsequent LOGON/AUTOLOG/XAUTOLOG that contains an invalid password.

The variable *y* is the value that when reached causes CP to send a message to the virtual machine with the user ID specified by the LOGUID operand for that and each subsequent LOGON/AUTOLOG/XAUTOLOG that contains an invalid password.

The variable *z* is the value that determines how many logon attempts CP is to permit before it returns the terminal to the VM logo. When more than *z* logon attempts have been made with invalid passwords from a single terminal, CP displays a new VM logon screen and allows a new logon sequence to be started.

The variable *z* also indicates how many AUTOLOG and XAUTOLOG attempts with invalid passwords CP allows. No further AUTOLOG or XAUTOLOG commands can be entered by a user who has entered more than *z* AUTOLOG or XAUTOLOG commands with invalid passwords for the duration of the logon session. To dynamically set the system limit for the number of incorrect passwords a user can enter on the AUTOLOG and XAUTOLOG commands, use the SET CMDLIMIT command. To query the system limit for the number of incorrect passwords that are allowed, use the QUERY CMDLIMIT command.

**Note:** The variable *z* can be any decimal number from 1 to 255. Variables *x* and *y* can be any decimal number from 0 to 255. The number 0 is a special case indicating that the applicable function should be bypassed. For example, if LOGLMT=(0,5,5) is specified, no accounting records are generated.

**LNKUID=OPERATOR****LNKUID=*userid***

specifies the user ID that should receive the indication that an invalid link password count has been reached or exceeded. If this user ID is disconnected or logged off, the system operator receives the message generated.

**LNKLMT=(2,5,10)****LNKLMT=(*x,y,z*)**

specifies the invalid link password threshold. The value specified applies to a single user ID for a single logon session. The variable *x* is the value that when

reached causes a type 06 accounting record to be generated for that and each subsequent LINK command that contains an invalid password. The variable *y* is the value that when reached causes a message to be sent to the user ID specified by the LNKUID operand for that and each subsequent LINK command that contains an invalid password. The variable *z* is the value that when reached causes the LINK command to be disabled for the current logon session.

**Note:** The variable *z* (which in VM/XA has a limit of 10) can be any decimal number from 1 to 255. Variables *x* and *y* can be any decimal number from 0 to 255. The number 0 is a special case indicating that the applicable function is to be bypassed. For example, if LNKLMT=(2,0,10) is specified, no message is sent.

#### **PSUPRS=NO**

#### **PSUPRS=YES**

specifies whether the facility that suppresses the password on the command line is initially part of the system being generated. Coding PSUPRS=YES prevents passwords entered on the AUTOLOG, LINK, LOGON, or XAUTOLOG command line from being accepted. The status of the password suppression facility can be modified after system initialization with the SET PASSWORD command. Specifying PSUPRS=YES provides additional security for your installation.

## Examples

Code the SYSJRL macro as follows

```
SYSJRL JOURNAL=YES,      *
      STQUERY=YES,      *
      LOGUID=LOGGER,    *
      LOGLOC=(5,120),   *
      LOGLMT=(5,5,5),   *
      LNKUID=LOGGER,    *
      LNKLMT=(5,5,5),   *
      PSUPRS=YES
```

to specify the following system security options:

- The journaling facility is operative.
- The ability to use the SET and QUERY commands to set and query journaling options is available.
- User ID LOGGER receives the indication that an invalid logon password count has been reached or exceeded.
- A maximum of five invalid password attempts are allowed to a given user or terminal after which an error message is issued and logons from that user ID and terminal are not allowed for 2 hours.
- After five unsuccessful logon attempts, CP writes a type 04 accounting record and sends a message to LOGGER, and no more than five AUTOLOG and XAUTOLOG attempts with invalid passwords are allowed.
- User ID LOGGER should receive the indication that an invalid link password count has been reached or exceeded.
- If five invalid link passwords are entered by a single user for a single logon session, CP generates a type 06 accounting record, sends a message to LOGGER (specified by LNKUID) and disables the link command for that user for the current logon session.
- The password suppression on the command line is initially part of the system.

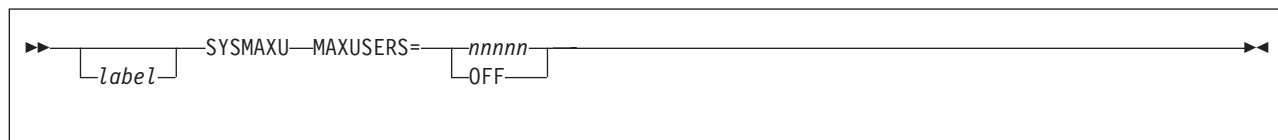
## Migration Aids

You can achieve the function of the SYSJRL macroinstruction by using the following statements in a system configuration file:

- FEATURES—See “FEATURES Statement” on page 136.
- JOURNALING—See “JOURNALING Statement” on page 160.

---

## **SYSMAXU (Optional)**



### **Purpose**

Code the SYSMAXU macroinstruction to specify the user logon limit. The SYSMAXU macro is optional. If you code it, you must place it after the first five required macros in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SYSEND macro. If the SYSMAX macro is not specified, the system does not set any limit.

### **Parameters**

*nnnnn*

is the maximum number of users that can be logged on. The range is from 1 to 99999.

**OFF**

indicates that an unlimited number of users can log on.

### **Usage Notes**

1. If this macro is not specified, the system does not set any limits.
2. The operator or a virtual machine, if the user directory for the virtual machine was created with the IGNMAXU operand of the OPTION control statement, can log on or be logged on by the AUTOLOG or XAUTOLOG command even if the number of logged on users is equal to or greater than the maximum user limit.

### **Examples**

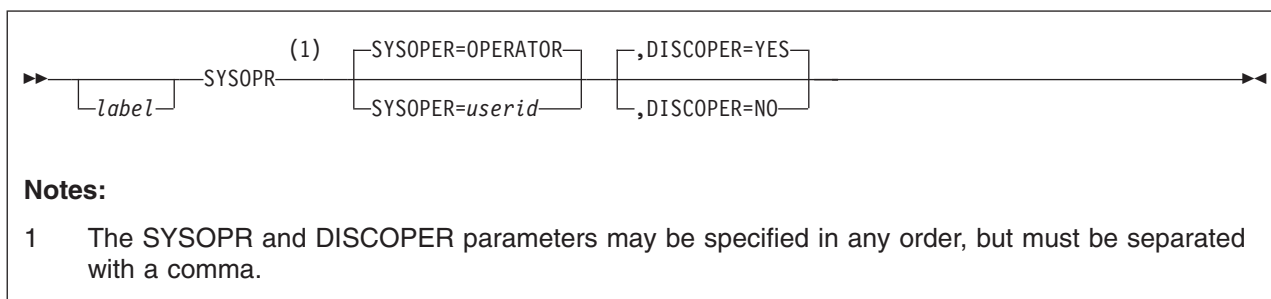
To set the limit to 9999, code the SYSMAXU macro as follows:

```
SYSMAXU MAXUSERS=9999
```

### **Migration Aids**

You can achieve the function of the SYSMAXU macroinstruction by using the FEATURES statement in a system configuration file. See “FEATURES Statement” on page 136.

## SYSOPR (Required)



## Purpose

Code the SYSOPR macroinstruction to specify the primary system operator's user ID and disconnect status. SYSOPR is required and must appear as one of the first five macroinstructions in HCPSYS ASSEMBLE.

## Parameters

### SYSOPER=OPERATOR

#### SYSOPER=*userid*

is the primary system operator's user ID. The variable *userid* is an alphanumeric character string up to 8 characters long. If you do not code the SYSOPER operand, the user ID is automatically OPERATOR.

### DISCOPER=YES

#### DISCOPER=NO

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 DISCOPER=YES, 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 DISCOPER=NO, CP does not force the primary system operator to be disconnected. If you do not code the DISCOPER operand, YES is assumed.

Specifying DISCOPER=NO has implications for system security. For more information, see usage note 3.

## Usage Notes

1. CP logs on a system operator with the user ID you specify on the SYSOPER operand, regardless of whether that user ID has an entry in the user directory.
2. If you do not code a SYSOPR macroinstruction in HCPSYS ASSEMBLE, at system initialization CP logs on a virtual machine with the user ID OPERATOR.
3. If you specify DISCOPER=NO, 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 DISCOPER=NO, make sure that the primary system console and all alternate system consoles are in a secure environment and are continuously monitored by authorized personnel.

## SYSOPR

4. 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 about the ALTERNATE\_OPERATORS statement, see “ALTERNATE\_OPERATORS Statement” on page 58.

For more information on defining operator privilege class, see the “PRIV\_CLASSES Statement” on page 184 or “SYSFCN (Optional)” on page 673.

## Examples

1. To specify OPERATOR as the user ID of the primary system operator and enable the disconnect function, code the SYSOPR macro as one of the following:

```
SYSOPR
SYSOPR DISCOPER=YES
SYSOPR SYSOPER=OPERATOR
SYSOPR SYSOPER=OPERATOR,DISCOPER=YES
```
2. To specify OP as the user ID for the primary system operator and disable the disconnect function, code the SYSOPR macro as follows:

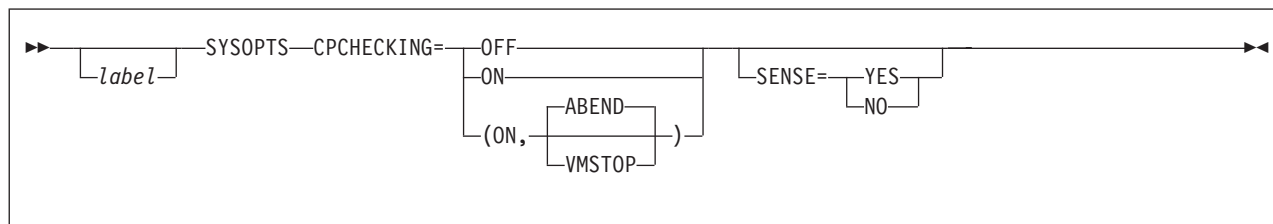
```
SYSOPR SYSOPER=OP,DISCOPER=NO
```

## Migration Aids

You can achieve the function of the SYSOPR macroinstruction by using the SYSTEM\_USERIDS statement in a system configuration file. See “SYSTEM\_USERIDS Statement” on page 232.



## SYSOPTS (Optional)



### Purpose

Code the SYSOPTS macroinstruction to specify initialization options for CP. You can specify whether CP should attempt to dynamically allocate device blocks for devices not already defined in the system configuration file or HCPRIO.

The SYSOPTS macroinstruction is optional; if you code it, you must place it in HCPSYS ASSEMBLE before the SYSEND macroinstruction.

### Parameters

#### **CPCHECKING=OFF**

Indicates that CP internal checking is to be deactivated.

#### **CPCHECKING=ON**

tells CP to activate CP internal checking to confirm the assertions established in CP code and take the appropriate action based on whether ABEND or VMSTOP is specified.

#### **CPCHECKING=(ON,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.

#### **CPCHECKING=(ON,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 a Diagnose X'8' instruction 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 (via a Diagnose X'8' **MESSAGE** \* command) providing information about the cause of the stop.

#### **SENSE=YES**

tells CP to sense devices dynamically and build device definition blocks for devices not already defined in the system configuration file or HCPRIO.

#### **SENSE=NO**

tells CP not to sense devices dynamically and tells CP to build device definition blocks for only those devices explicitly defined in the system configuration file using RDEVICE statements or in HCPRIO using RDEVICE macroinstructions.

### Usage Notes

1. If you do not specify the SYSOPTS macro, or if you specify the SYSOPTS macro without specifying the SENSE option, CP's default action for devices that you did not explicitly define in the system configuration file or in HCPRIO ASSEMBLE is SENSE=YES.

## SYSOPTS

2. If you specify SENSE=NO, you must explicitly define all the devices required at IPL time in either the system configuration file or in HCPRIO ASSEMBLE. If you do not define the minimum number of required CP-owned volumes in the system configuration file or in HCPRIO ASSEMBLE, CP will end the IPL by loading a disabled wait state.
3. Even if you specify SENSE=NO, you can use the SET RDEVICE command to define new devices after you initialize CP. For more information about the SET RDEVICE command, see the *z/VM: CP Commands and Utilities Reference*.

## Examples

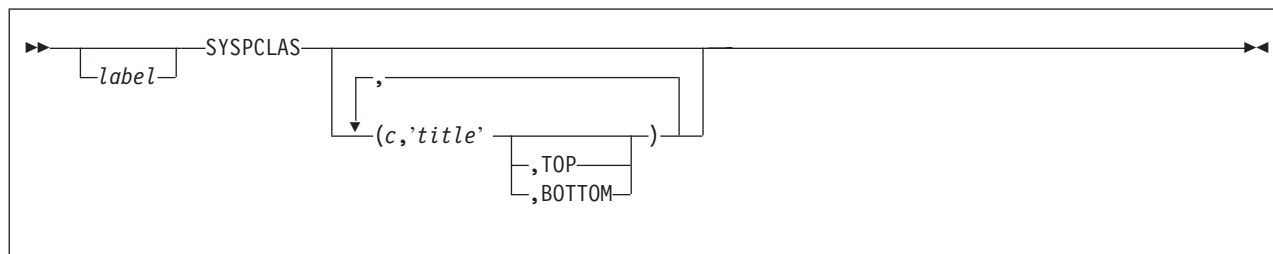
To have CP build device blocks for devices that have not been defined in the system configuration file or in HCPRIO ASSEMBLE, code the SYSOPTS macro as follows:

```
SYSOPTS SENSE=YES
```

## Migration Aids

There is no equivalent statement in a system configuration file for the SYSOPTS macroinstruction.

## SYSPCLAS (Optional)



### Purpose

Code the SYSPCLAS macro 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, 0 through 9). The SYSPCLAS macroinstruction is optional; if you code it, you must place it after the first five required macroinstructions in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SYSEND macroinstruction.

You can specify the SYSPCLAS macro only once. If you omit the SYSPCLAS macro, or you code the macro with no operands, CP does not print any classification titles. If more than one print class is to have a classification title, the operand should be repeated for each class. Each operand is separated by a comma.

### Parameters

*c* is one alphanumeric character specifying the spool file class.

*title*

is a classification title for this class. It may be up to 46 characters long. It must be enclosed in single quotation marks.

**TOP**

specifies that this title is to be printed both on the separator page and at the top of each page of output.

**BOTTOM**

specifies that this title is to be printed both on the separator page and at the bottom of each page of output.

### Usage Notes

- The title line is inserted 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 that is printing the file. There are several implications to this:
  - The titles are inserted as the file is being created. If the file is later changed to a different class, the titles in the file are not changed. 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. This can produce blank pages within the output listing.

## SYSPCLAS

- If the operating system running in the virtual machine that is printing the file does not use Skip to Channel 1, the title lines are never printed.
2. If you specify the same class more than once, CP prints only the first title you specify.

## Examples

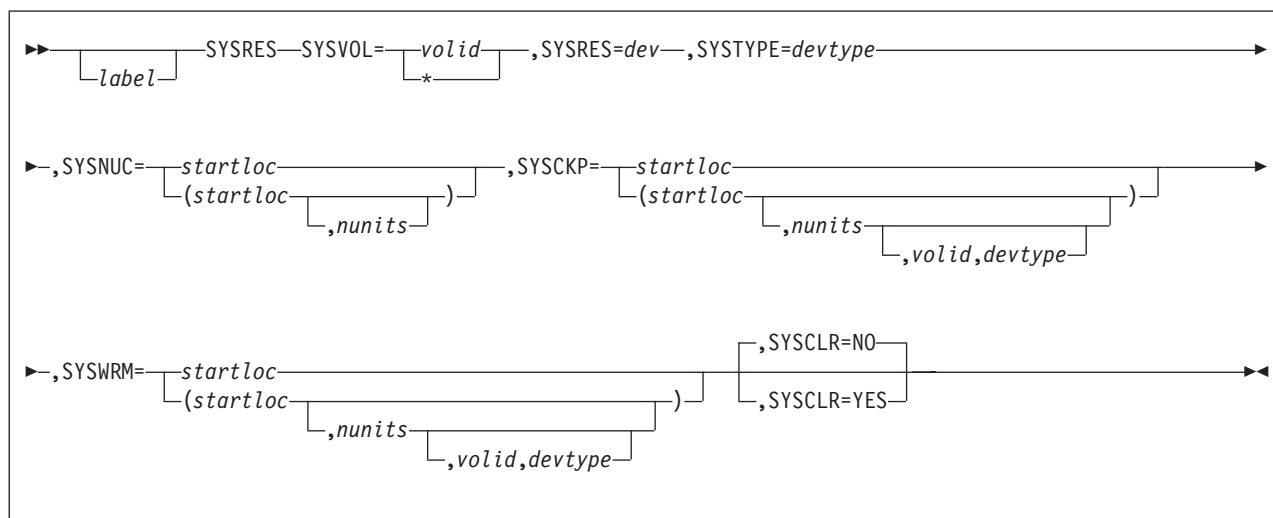
1. If you do not want to have CP print classification titles, omit the SYSPCLAS operand or code it as follows:  
SYSPCLAS
2. If you want CP to print the title “SPECIAL PRIORITY—TOP SECRET” on the separator page of all class R output, code the SYSPCLAS macro as follows:  
SYSPCLAS (R, 'SPECIAL PRIORITY -- TOP SECRET')
3. If you want CP to print the title “SPECIAL PRIORITY—TOP SECRET” on the separator page and at the top of each page of all class N output, code the SYSPCLAS macro as follows:  
SYSPCLAS (N, 'SPECIAL PRIORITY -- TOP SECRET',TOP)
4. If you want CP to (a) print the title “NO PRIORITY” on the separator page of all class B output, (b) print the title “INTERNAL PRIORITY” on the separator page and at the bottom of each page of all class N output, and (c) print the title “SPECIAL PRIORITY—TOP SECRET” on the separator page and at the top of each page of all class Q output, code the SYSPCLAS macro as follows:  
SYSPCLAS (B, 'NO PRIORITY',), \*  
(N, 'INTERNAL PRIORITY',BOTTOM), \*  
(Q, 'SPECIAL PRIORITY -- TOP SECRET',TOP)

**Note:** If you need more than one line to code the SYSPCLAS macro, you *must* code a continuation character in column 72 *and* start the next line in column 16.

## Migration Aids

You can achieve the function of the SYSPCLAS macroinstruction by using the PRINTER\_TITLE statement in a system configuration file. See “PRINTER\_TITLE Statement” on page 182.

## SYSRES (Required)



## Purpose

Code the SYSRES macroinstruction to describe the layout of the CP system residence disk. The SYSRES macroinstruction is required and must appear as one of the first five macroinstructions in HCPSYS ASSEMBLE.

**Note:** For information on how to calculate the values for the SYSCKP and SYSWRM operands, refer to Chapter 20, “Allocating DASD Space,” on page 603.

## Parameters

**SYSVOL=valid**

**SYSVOL=\***

is not used; code any 1-character to 6-character string.

**SYSRES=dev**

is not used; code any 1-character to 4-character string.

**SYSTYPE=devtype**

is not used; code any 4-character string.

**SYSNUC=startloc**

**SYSNUC=(startloc, nunits)**

is not used; code any 1-character to 6-character string.

**SYSCKP=startloc**

**SYSCKP=(startloc, nunits, valid, devtype)**

specifies the real starting location and, optionally, the maximum number of units the checkpoint process uses to preserve system restart data.

**Note:** This definition is overridden by the SYSTEM\_RESIDENCE statement in the system configuration file.

The variable *startloc* is either the cylinder or page number that designates the real starting location where the CP checkpoint information is to be saved. For CKD device types, this value is a nonzero decimal cylinder number ranging from 1 to 65535. For FBA devices, this value is a 1-digit to 6-digit 4 KB page number that is greater than 3 (pages 0-3 are reserved).

The variable *nunits* is a number designating the maximum number of cylinders or 4 KB pages to be allocated for the checkpoint information area. For CKD device types, 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. If you do not specify the *nunits* option, the default depends on the device type:

Device Type	Cylinders	Pages
3380 (emulated)	1	
3390 (real or emulated)	1	
FBA (emulated)		100

For information on calculating the size of the checkpoint area, refer to “Checkpoint Data” on page 607.

The variable *valid* defines the volume label of the device to be used as the SYSCKP device. If not specified, it defaults to the system resident volume.

The variable *devtype* is the device type of the checkpoint volume and must be specified when the volume label is explicitly coded:

- For a 3390 in native mode, specify 3390.
- For a 3390 in 3380 track compatibility mode, specify 3380.
- For a RAMAC<sup>®</sup> or 2105 emulating a 3380, specify 3380.
- For a RAMAC or 2105 emulating a 3390, specify 3390.
- For a 2105 emulating a 9336, specify 9336 or the generic FB-512.

**SYSWRM=***startloc*

**SYSWRM=**(*startloc,nunits,valid,devtype*)

is the real starting location and, optionally, the maximum number of units to be used for saving warm start data.

**Note:** This definition is overridden by the SYSTEM\_RESIDENCE statement in the system configuration file.

The variable *startloc* is either 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 65535. For FBA devices, this value is a 1-digit to 6-digit 4 KB page number that is greater than 3 (pages 0-3 are reserved).

The variable *nunits* is a number designating the maximum number of cylinders or 4 KB pages to be allocated for the warm start information area. For CKD device types, 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. If you do not specify the *nunits* option, the default depends on the device type:

Device Type	Cylinders	Pages
3380 (emulated)	1	
3390 (real or emulated)	1	
FBA (emulated)		75

For information on calculating the size of the warm start area, refer to “Warm Start Data” on page 606.

The variable *valid* defines the volume label of the device to be used as the SYSWRM device. If not specified, it defaults to the system resident volume.

The variable *devtype* is the device type of the warm start volume and must be specified when the volume ID is explicitly coded:

- For a 3390 in native mode, specify 3390.
- For a 3390 in 3380 track compatibility mode, specify 3380.
- For a RAMAC or 2105 emulating a 3380, specify 3380.
- For a RAMAC or 2105 emulating a 3390, specify 3390.
- For a 2105 emulating a 9336, specify 9336 or the generic FB-512.

#### **SYSCLR=YES**

#### **SYSCLR=NO**

specifies automatic clearing of all previously written data and directory areas that are on TDISK DASD space. This prevents other users from accidentally accessing these areas. If YES is specified, TDISK DASD space is cleared to binary zeros at CP initialization and when CP attaches a CP-owned volume that contains TDISK allocations. TDISK space is also cleared when a user detaches a TDISK minidisk. Specifying SYSCLR=YES provides additional security for your VM/ESA installation. If you specify NO and the TDISK is a CKD device, the system clears only cylinder 0. If you specify NO and the TDISK is an FBA device, the system clears only the first two pages. The SYSCLR option is not required. The default is NO.

**Note:** This setting is overridden by the CLEAR\_TDISK setting on the FEATURES statement in the system configuration file.

## Usage Notes

1. The SYSVOL=, SYSRES=, SYSTYPE=, and SYSNUC= parameters are not used because CP is IPLed from a module.
2. The SYSCKP and SYSWRM areas must not overlap.
3. When preparing a system residence device, be sure to format CP paging, spooling, dumping, directory, warm start, and checkpoint areas in 4 KB format. You can use the Device Support Facilities program (ICKDSF), or the CMS utility CPFMTXA. However, ICKDSF is the preferred method of CP volume maintenance and is invoked by CPFMTXA.
4. You must specify the system residence volume identification for the checkpoint area and the warm start area on the SYSCPVOL macroinstruction. For more information about the SYSCPVOL macroinstruction, see “SYSCPVOL (Required)” on page 664.
5. You can query the current setting of the SYSCLR option by entering the QUERY TDISKCLR command.

## Examples

1. The following example shows how to code the SYSRES instruction so that:
  - The warm start area begins at cylinder 204 and occupies one cylinder on the 3390 DASD identified by SYS001.
  - The checkpoint area begins at page 101 and occupies 300 pages the 9336 DASD identified by SYS002.
  - CP clears:
    - All TDISK space during system IPL

## SYSRES

- All TDISK space after IPL whenever it attaches a DASD volume to the system for use as temporary disk space because of an ATTACH command
- TDISK minidisks whenever they are detached from a guest virtual machine because of a DETACH command.

```
SYSRES SYSVOL=xxxxxx,SYSRES=xxxx,SYSTYPE=xxxx,SYSNUC=xxxxxx, *  
        SYSWRM=(204,1,SYSO01,3390),SYSCKP=(100,300,SYSO02,9336), *  
        SYSCLR=YES
```

**Note:** If you need more than one line to code the SYSRES macro, you *must* code a continuation character in column 72 *and* start the next line in column 16.

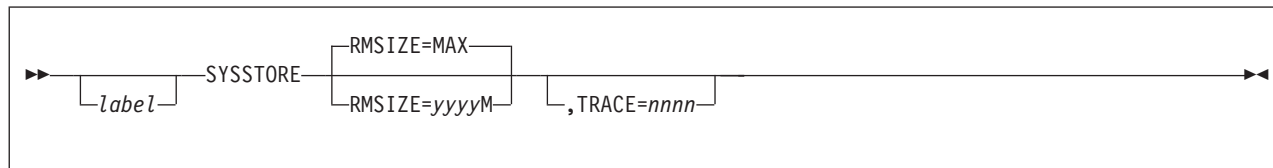
## Migration Aids

You can achieve the function of the SYSRES macroinstruction by using the following statements in a system configuration file:

- FEATURES—See “FEATURES Statement” on page 136.
- SYSTEM\_RESIDENCE—See “SYSTEM\_RESIDENCE Statement” on page 230.



## SYSSTORE (Required)



### Purpose

Code the SYSSTORE macroinstruction to configure the use of real storage. This macroinstruction is required and must appear as one of the first five macroinstructions in HCPSYS ASSEMBLE.

### Parameters

#### **RMSIZE=MAX**

tells CP to use the maximum available real storage that you have installed.

Because CP no longer creates static storage management tables, you need not specify the exact amount of storage to be used.

#### **RMSIZE=yyyyM**

specifies the amount of real storage available for CP. If the actual amount of real machine storage is greater than RMSIZE, the storage available to CP will be the RMSIZE value. If the actual amount of real machine storage is less than RMSIZE, the system displays a message to the operator at initialization time stating the amount of storage available.

The variable *yyyy* is a one- to four-digit decimal number that states the number of megabytes (M) of real storage. The value of *yyyy* can range from 1 to 2048. One megabyte is 1,048,576 bytes.

#### **Notes:**

1. CP no longer creates storage management tables according to the RMSIZE operand you specify. Instead, it builds the necessary storage management at IPL time based on the maximum available amount of storage. Therefore, if you specify an RMSIZE value greater than the actual storage available, you will not waste storage.
2. The maximum values you specify for the remaining operands are determined by the actual amount of real storage your system has, not necessarily by the value you specify on the RMSIZE operand.

#### **TRACE=nnnn**

specifies the decimal number of 4 KB pages CP allocates to the internal trace table for each processor in the configuration. For example, if you run z/VM on a uniprocessor and you specify TRACE=4, CP allocates 4 pages as the internal trace table. If you run z/VM on a dyadic processor and you specify TRACE=4, CP allocates 8 pages as the internal trace table (4 pages for each processor).

If you do not specify the TRACE operand, CP allocates to the internal trace table 1 page for each 64 pages (256 KB) of real storage or 100 pages, whichever is smaller.

The minimum value you can specify is TRACE=3; the maximum value must be less than or equal to 1/4 of RMSIZE. For example, if you specify RMSIZE=16m (4096 pages), then you must then specify TRACE as less than or equal to 448.

## SYSSTORE

### Examples

1. To specify 16 MB of real storage and allow CP default values for the internal trace table, code the SYSSTORE macro as follows:

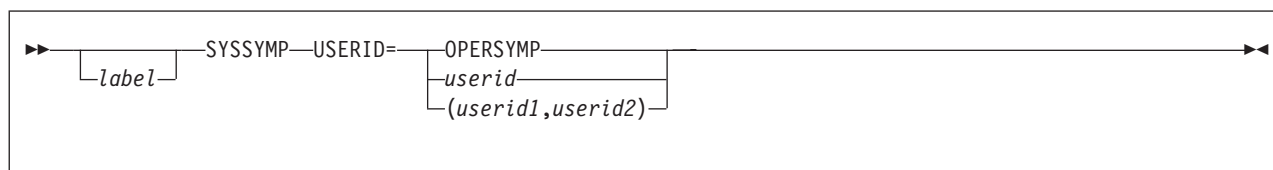
```
SYSSTORE RMSIZE=16M
```

CP assigns 64 pages per processor for the internal trace table (1 page for each 256 KB of real storage, or 4 pages for each MB).

### Migration Aids

You can achieve the function of the SYSSTORE macroinstruction by using the STORAGE statement in a system configuration file. See “STORAGE Statement” on page 222.

## SYSSYMP (Optional)



### Purpose

Code the SYSSYMP macroinstruction to specify the user IDs of one or two virtual machines for which the CP \*SYMPTOM system service is to accumulate symptom records. When the virtual machines identified to CP connect to the CP \*SYMPTOM system service, symptom records are sent to them.

In addition, these virtual machines are automatically logged on during CP initialization.

The SYSSYMP macroinstruction is optional; if you code it, you must place it after the first five required macroinstructions in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SYSEND macroinstruction. If this macro is not coded, CP processes the virtual machine having the user ID OPERSYMP as the symptom record recording virtual machine.

**Note:** Any virtual machine specified by the SYSSYMP macro must contain an IUCV directory control statement that authorizes the application running in the virtual machine to connect to the CP \*SYMPTOM system service. Failure to supply this required authority results in the failure of the connection request from the virtual machine. If the RETRIEVE utility is used to collect symptom records and this authority is not coded in the directory, RETRIEVE will fail.

### Parameters

**USERID=OPERSYMP**

**USERID=userid**

**USERID=(userid1,userid2)**

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 up to 8 characters each. OPERSYMP is the default user ID.

### Usage Notes

If the user ID specified on the SYSSYMP macroinstruction has an entry in the user directory at system initialization, the virtual machine designated by that user ID is logged on to the system.

### Examples

- To specify OPERSYMP as the user ID of the symptom record recording virtual machine, you can do one of the following:
  - Omit the SYSSYMP macro
  - Code the SYSSYMP macro as follows:
 

```
SYSSYMP USERID=OPERSYMP
```

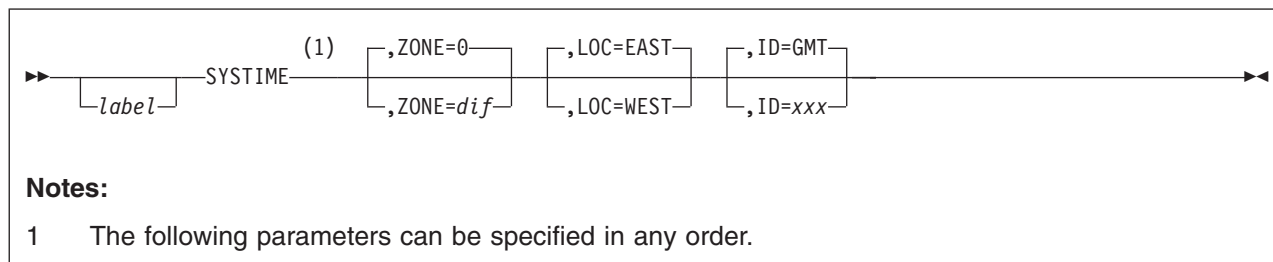
## SYSSYMP

2. To specify DISKSYMP as the user ID of the symptom record recording virtual machine, code the SYSSYMP macro as follows:  
`SYSSYMP USERID=DISKSYMP`
3. To specify OPERSYMP and DISKSYMP as the user IDs of the symptom record recording virtual machines, code the SYSSYMP macro as follows:  
`SYSSYMP USERID=(OPERSYMP,DISKSYMP)`

## Migration Aids

You can achieve the function of the SYSSYMP macroinstruction by using the `SYSTEM_USERIDS` statement in a system configuration file. See “`SYSTEM_USERIDS` Statement” on page 232.

## SYSTIME (Required)



## Purpose

Code the SYSTIME macroinstruction to generate information that CP needs to set the hardware time of day (TOD) clock. The value stored in the TOD clock always represents Coordinated Universal Time (UTC); the SYSTIME macroinstruction lets you correct the value to local time. The SYSTIME macroinstruction is required and must appear as one of the first five macroinstructions in HCPSYS ASSEMBLE.

## Parameters

### ZONE=0

### ZONE=*dif*

is the time zone differential from Coordinated Universal Time (UTC). If you do not specify ZONE, CP uses a differential value of 0 hours from UTC.

The value *dif* is the time zone differential specified in hours, minutes, and seconds (h,m,s). The form for *dif* can be:

(h,m,s)  
 (h,m)  
 (h,,s)  
 (,m,s)  
 (,,s)  
 (,m)  
 h

Note that h,m, and s are positional; if not specified, they must be replaced by commas. If you code only hours, you do not have to use parentheses. For instance, you can specify ZONE=5, or ZONE=(7,30,30), or ZONE=(6,,30).

The total value of *dif* cannot exceed 13 hours.

### LOC=EAST

### LOC=WEST

specifies whether the time zone differential is to be taken as EAST or WEST of Coordinated Universal Time. EAST is the default. If the value of ZONE is 0, the macro definition ignores the setting of LOC.

### ID=GMT

### ID=*xxx*

is the name of the time zone. The default is GMT (Greenwich Mean Time). The value *xxx* is a 3-character alphanumeric string.

## SYSTIME

### Examples

1. If your installation is in the Eastern Standard Time zone, code the SYSTIME macro as follows:

```
SYSTIME ZONE=5,LOC=WEST,ID=EST
```

2. If your installation is in the Pacific Standard Time zone, code the SYSTIME macro as follows:

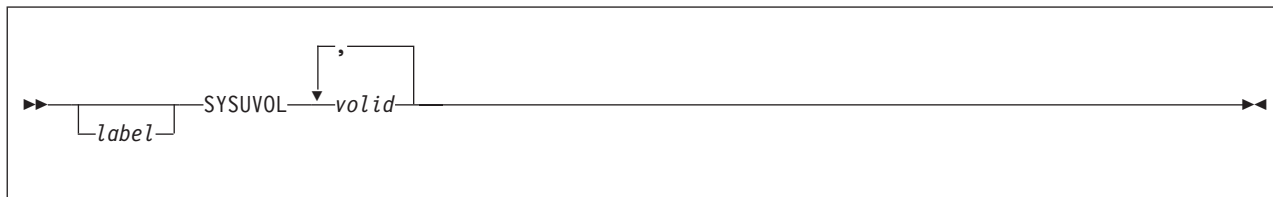
```
SYSTIME ZONE=8,LOC=WEST,ID=PST
```

### Migration Aids

You can achieve the function of the SYSTIME macroinstruction by using the following statements in a system configuration file:

- TIMEZONE\_BOUNDARY—See “TIMEZONE\_BOUNDARY Statement” on page 237.
- TIMEZONE\_DEFINITION—See “TIMEZONE\_DEFINITION Statement” on page 239.

## SYSUVOL (Optional)



### Purpose

Code the SYSUVOL macroinstruction 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 but does not use them for paging, spooling, directory, or temporary disk space. A volume must be attached to the system before users can link to minidisks it contains.

The SYSUVOL macroinstruction is optional; if you code it, you must place it after the first five required macroinstructions in HCPSYS ASSEMBLE (SYSCPVOL, SYSOPR, SYSRES, SYSSTORE, and SYSTIME) but before the SYSEND or CSESYS macroinstruction.

### Parameters

*valid*

is the user volume identification. The variable *valid* is a 1- to 6-character identifier. Each character can be a letter, number, or hyphen.

### Usage Notes

1. If you do not specify user volumes in the SYSUVOL macroinstruction, a volume can nevertheless be attached to a virtual machine or system. Use the CP ATTACH command to do so.
2. If a volume is specified in the SYSUVOL macroinstruction but is not mounted when CP is loaded, CP considers that volume as unavailable. CP notifies the operator that the volume is not mounted and continues processing, if possible. The system operator can mount and attach (ATTACH command) the volume to the system when it is needed. This provides extra space for future growth.
3. You can also define user volumes by specifying a generic volume identifier. For more information, see “SYSINCL (Optional)” on page 680.
4. When more than one DASD volume is found at system initialization (IPL) to have the same volume serial (*valid*), and that volume serial is in the CP or user volume list, the volume attached to the system is the one at the lowest device number. This rule does not apply to duplicates of the system residence (IPLed) volume.

### Examples

To specify UDSK1, UDSK2, and UDSK3 as user volumes, code the SYSUVOL macro as follows:

```
SYSUVOL UDSK1,UDSK2,UDSK3
```

### Migration Aids

You can achieve the function of the SYSUVOL macroinstruction by using the USER\_VOLUME\_LIST statement in a system configuration file. See

## SYSUVOL

"USER\_VOLUME\_LIST Statement" on page 255.



---

## Appendix C. Defining I/O Devices Using HCPRIO

This appendix describes how to define your I/O devices to CP using the RDEVICE macroinstruction in the real I/O configuration file (HCPRIO ASSEMBLE).

**Note:** Now that z/VM uses device sensing to dynamically determine the device characteristics, most devices no longer require an RDEVICE macroinstruction to be coded in HCPRIO. However, some older devices don't support device sensing and need to be explicitly defined in either the system configuration file or HCPRIO. To determine if a given device can be defined in the system configuration file refer to Chapter 5, "Defining I/O Devices," on page 45. If the device is not listed in any of the configuration guide tables for the system configuration file, then you must define this device in HCPRIO. Devices that are defined in HCPRIO are considered static definitions and cannot be changed with either the RDEVICE statement in the system configuration file or the SET RDEVICE command.

No additional device support is being provided in HCPRIO. New device support will only be provided in the SYSTEM CONFIGURATION file.

---

### Coding HCPRIO ASSEMBLE

To define your installation's real I/O configuration in the real I/O configuration file, HCPRIO ASSEMBLE, you must code macroinstructions in the file. HCPRIO ASSEMBLE has two macroinstructions:

#### RDEVICE

Use this macroinstruction to define each real I/O device or group of I/O devices in your installation's I/O configuration.

#### RIOGEN

Use this macroinstruction to end the HCPRIO assembly and to define your installation's primary and alternate system consoles.

**Note:** On 3090™ processor complexes, you must define the real I/O configuration to the hardware. This means that *rdev* will be device specific on these processors. While the HCPRIO macroinstruction will assemble correctly with any real device number between X'0000' and X'FFFF', the input/output configuration program (IOCP) for each processor has different restrictions. See the *3090 Input/Output Configuration Program User's Guide and Reference* book.

The following rules apply to the HCPRIO ASSEMBLE file.

1. You may code the RDEVICE macroinstructions in any order. z/VM does not support multiple RDEVICE macroinstruction statements for a single physical device.
2. You can specify a maximum of 4096 devices for the real system.
3. The RDEVICE macroinstruction definition ignores any labels that you specify on RDEVICE macroinstructions. The macro definition appends the device number to the characters RDEV to generate its own label. For example, the macro definition generates the label RDEV0234 for device number 0234.
4. You must code the RIOGEN macroinstruction last in HCPRIO ASSEMBLE.

## Defining I/O Devices

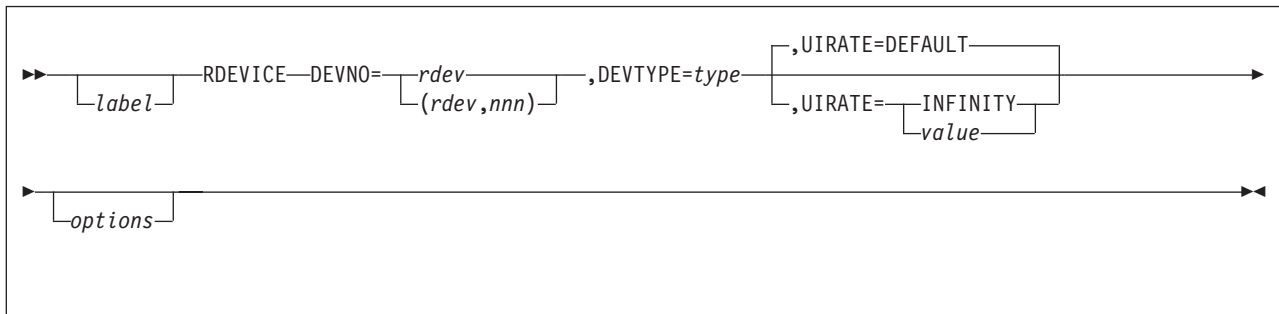
5. You must not code START, CSECT, or END statements in the HCPRIO ASSEMBLE—the macroinstruction expansions generate these records. If you code these statements, an MNOTE is issued.
6. Device initialization is started in the order listed in the HCPRIO, but each device is initialized asynchronously. This results in device initializations completing in a different order from that listed in the HCPRIO. If duplicate DASD volume labels are in the HCPRIO, the volume with the lowest real device number is attached to the system and brought online.
7. You must be very careful when coding statements before the first RDEVICE macroinstruction because many statements that may not generate code do generate implicit CSECT definitions that will cause problems later. SPACE, TITLE, EJECT, and COMMENT cards are always specifically allowed before the first RDEVICE macro.

---

## RDEVICE Macroinstructions

You must code an RDEVICE macroinstruction for each real I/O device or group of devices in the real I/O configuration. During system initialization, CP brings online all devices you define in HCPRIO for which there are real counterparts. CP brings DASDs online based on the lowest physical address.

The general format of the RDEVICE macroinstruction is as follows:



**DEVNO=*rdev***

**DEVNO=(*rdev*,*nnn*)**

DEVNO=*rdev* specifies the real I/O device number you assign a device. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev*,*nnnn*) specifies the real I/O device numbers you assign a range of devices. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF. The variable *nnnn* is a decimal number that tells CP to create a group of real device specifications with consecutive device numbers. For example, if you specify DEVNO=(0100,8), real device numbers 0100, 0101, 0102, 0103, 0104, 0105, 0106, and 0107 are defined. The default value for *nnnn* is 1.

**DEVTYPE=*type***

DEVTYPE=*type* specifies the type of device.

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=*value***

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

*options*

*options* are device-specific options you specify for the device.

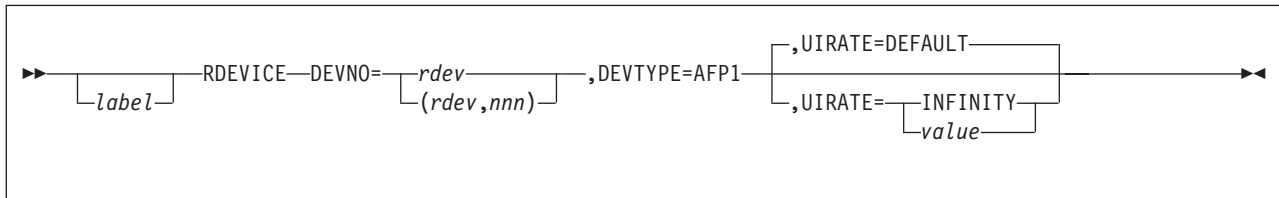
### RDEVICE—General Usage Notes

1. Appendix D, “Configuration Guide for Printers,” on page 769 lists according to device:
  - The DEVTYPE operand you must specify
  - Other required operands you must specify
  - Optional operands you can specify
  - The place in this chapter where coding of the RDEVICE macroinstruction for that device is described.
2. “Unsupported Devices” on page 763 describes how to code the RDEVICE macroinstruction for any unsupported devices.
3. Remember to code the RIOGEN macroinstruction at the end of your HCPRIO ASSEMBLE file. See “RIOGEN” on page 766 for details on the RIOGEN macroinstruction.

---

### Dedicated AFP\* Printers

## Common Control Unit (CCU) Printers

**Purpose**

The following printers are supported as advanced function printers using advanced function printing licensed programs:

3825                      3827                      3828                      3835                      3900

**Note:** Printers defined as advanced function printers that follow the CCU specifications can be used only as dedicated devices. For information about printers that follow CCU specifications, see the *AFP Printer Information* book. The printers listed above, that are useable only when dedicated, are Group 3 printers.

**Parameters**

**DEVNO=rdev**

**DEVNO=(rdev,nnn)**

`DEVNO=rdev` specifies a real I/O device number. It must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

`DEVNO=(rdev,nnn)` specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default value for *nnn* is 1.

**DEVTYPE=AFP1**

specifies an advanced function printer (AFP) that follows the CCU specification. Note that printers defined in this manner can only be used as dedicated devices.

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=value**

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no `UIRATE` parameter specified.

**INFINITY** turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the `UIRATE` value to **INFINITY** for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a Hot I/O condition is detected.

For more information about advanced function printers, see the *PSF/VM System Programming Guide*.

### Usage Notes

1. When the RDEVICE macro is used with DEVTYPE=AFP1 to define an advanced function printer, only the DEVNO and DEVTYPE operand may be specified.

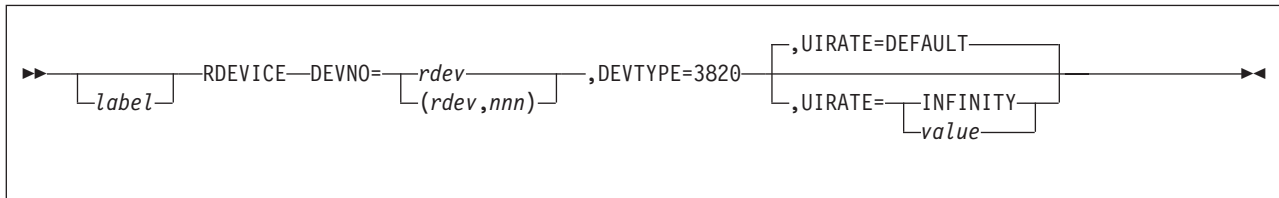
### Examples

1. To specify an advanced function printer at device number 0110, code the RDEVICE macro as follows:  

```
RDEVICE DEVNO=0110,DEVTYPE=AFP1
```
2. To specify three advanced function printers at device numbers 0111, 0112, and 0113, code the RDEVICE macro as follows:  

```
RDEVICE DEVNO=(0111,3),DEVTYPE=AFP1
```

## 3820 Printer

**Purpose**

The dedicated 3820 printers are defined using this RDEVICE format.

**Parameters**

**DEVNO=*rdev***

**DEVNO=(*rdev*,*nnn*)**

DEVNO=*rdev* specifies a real I/O device number. It must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev*,*nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default value for *nnn* is 1.

**DEVTYPE=3820**

specifies that a 3820 advanced function printer is being defined. Note that printers defined in this manner can be used only as dedicated devices.

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=*value***

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

**Usage Notes**

When the RDEVICE macro is used with DEVTYPE=3820 to define a 3820 printer, only the DEVNO and DEVTYPE operand may be specified.

**Examples**

- To specify a 3820 printer at device number 0110, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=0110,DEVTYPE=3820
```

2. To specify three 3820 printers at device numbers 0111, 0112, and 0113, code the RDEVICE macro as follows:

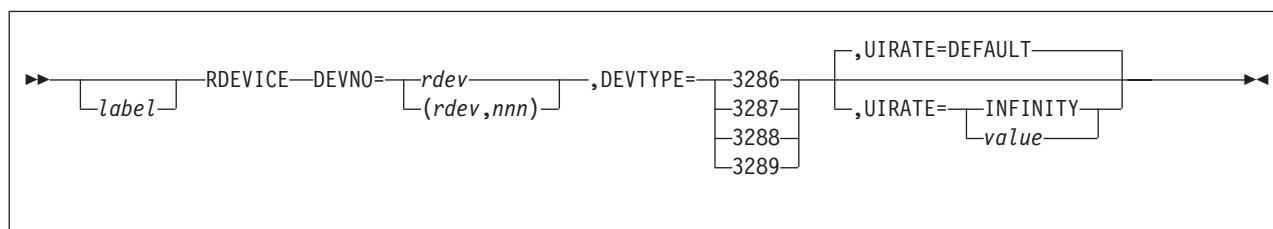
```
RDEVICE DEVNO=(0111,3),DEVTYPE=3820
```

---

**Printers Attached to a Display Control Unit**



## Display Printers



### Purpose

The following display printers are supported (display printers are printers that allow key-operator-initiated copy only):

3262	3287	3812 <sup>1 2</sup>	4224 <sup>1</sup>	5210
3268	3288	3816 <sup>1 2</sup>	4234 <sup>1</sup>	6262
3286	3289	4214	4245	

### Notes:

1. Supported as advanced function printers using Print Services Facility/Virtual Machine (PSF/VM), Program Number 5684-141.
2. You can download fonts using PSF/VM.

### Parameters

**DEVNO=*rdev***

**DEVNO=(*rdev*,*nnn*)**

DEVNO=*rdev* specifies a real I/O device number. It must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev*,*nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default value for *nnn* is 1.

**DEVTYPE=3286**

**DEVTYPE=3287**

**DEVTYPE=3288**

**DEVTYPE=3289**

specifies the type of device (3286, 3287, 3288, or 3289).

Other display printers supported are the 3262, 3268, 3812, 3816, 4214, 4224, 4234, 4245, 5210, and 6262. For these printers, however, DEVTYPE must be coded as 3287.

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=*value***

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

**INFINITY** turns hot I/O recovery off for the device.

## Display Printers

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

### Examples

1. To specify three 3286 display printers, at device numbers 0171, 0172, and 0173, code the RDEVICE macro as follows:  

```
RDEVICE DEVNO=(0171,3),DEVTYPE=3286
```
2. To specify a 3262 or 3268 display printer at device number 0174, code the RDEVICE macro as follows:  

```
RDEVICE DEVNO=0174,DEVTYPE=3287
```
3. To specify a 3288 display line printer at device number 0180, code the RDEVICE macro as follows:  

```
RDEVICE DEVNO=0180,DEVTYPE=3288
```
4. To specify 3289 display line printers at device numbers 0181, 0182, and 0183, code the RDEVICE macro as follows:  

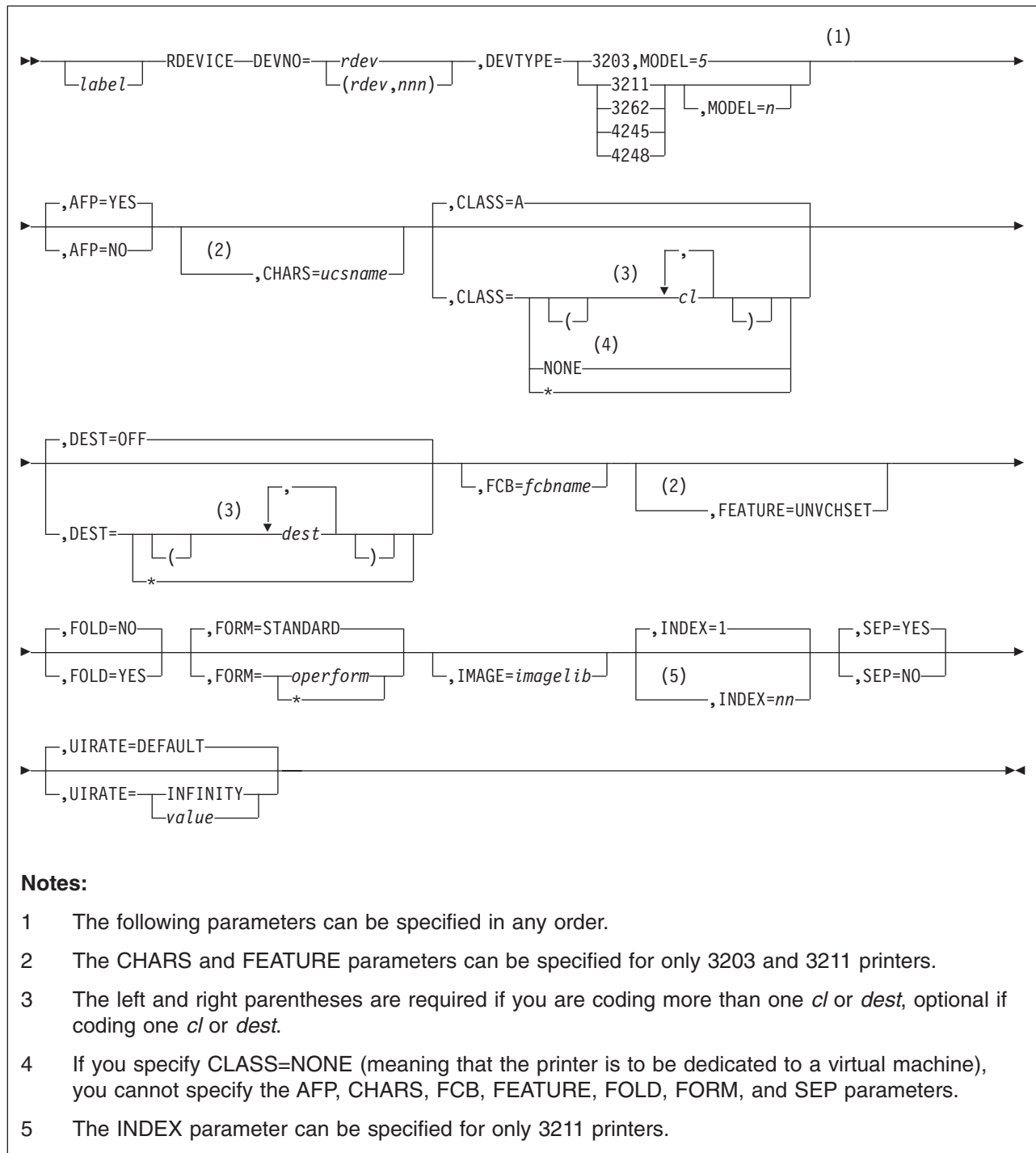
```
RDEVICE DEVNO=(0181,3),DEVTYPE=3289
```
5. To specify a 4234 display line printer at device number 0184, code the RDEVICE macro as follows:  

```
RDEVICE DEVNO=0184,DEVTYPE=3287
```

---

## CP System-Managed Impact Printers

## Impact Printers



### Purpose

The CP system-managed 3203, 3211, 3262, 4245, 4248, and 6262 impact printers are defined using this RDEVICE format.

### Parameters

DEVNO=*rdev*

**DEVNO=(*rdev,nnn*)**

DEVNO=*rdev* specifies a real I/O device number. It must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev,nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default value for *nnn* is 1.

**DEVTYPE=*devtype***

specifies the device type and model.

**Notes:**

1. The default is MODEL=5 for the 3262, and MODEL=1 for the 4245 and the 4248.
2. Although the 4248 Model 2 is supported, it must be specified as MODEL=1.
3. The 3262 Model 5 is supported as a 3262 or in 4248-compatibility mode. Specify it as a 4248 when it is used in 4248-compatibility mode.
4. The 4245 Model 1 is supported as a 4245 or in 3262-compatibility mode. Specify it as a 3262 MODEL=5 when it is used in 3262-compatibility mode.
5. The 4248 Models 1 and 2 are supported as 4248s or in 3211-compatibility mode. Specify it as a 3211 when it is used in 3211-compatibility mode.
6. The 6262 Model 14 must be specified as a 4248 MODEL=1.
7. The 6262 Model 22 must be specified as a 4248 MODEL=1.

**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=*ucsname***

For a printer with the UNVCHSET feature, CHARS=*ucsname* specifies the 1- to 4-character suffix of the name of the default universal character set (UCS) buffer image to be used. The variable *ucsname* must correspond to one of the UCS images stored in the image library. For example, if you specify:

```
RDEVICE DEVNO=0003,DEVTYPE=3203,CHARS=AN
```

The image library must contain a member named 3203AN.

**Class=*cl*****Class=A****Class=\*****Class=NONE**

specifies the output spooling class(es) the printer can print. (To change the spooling classes without recoding and reassembling the HCPRIO file, use the CP START command.)

You can specify CLASS in one of the following ways:

- If you omit the CLASS parameter, class A is assumed by default.
- CLASS=*cl*,... to list up to eight output spooling classes, separated by commas. The spooling class, *cl*, is one alphanumeric character. If you code only one spooling class, you do not have to use parentheses. For example, you can specify multiple classes as CLASS=(A,B,C) and a single class as CLASS=A or CLASS=(A).
- CLASS=NONE specifies that CP will not use the printer for spooling.
- CLASS=\* specifies that the printer processes files regardless of class.

**Note:** If you specify CLASS=NONE (meaning that the printer is to be dedicated to a virtual machine), you cannot specify the AFP, CHARS, FCB, FEATURE, FOLD, FORM, and SEP parameters.

**DEST=***dest*

**DEST=OFF**

**DEST=\***

specifies the output destination value(s) the printer can print. (To change the spooling destination values without having to recode and reassemble the HCPRIO file, use the CP START command.)

You can specify DEST in one of the following ways:

- By default—if you omit the DEST parameter, DEST OFF is assumed.
- DEST=*dest*,...—to list up to four output destination values, separated by commas. Each value, *dest*, is one to eight alphanumeric characters in length. If you code more than one destination, you must use parentheses. For example, you can specify multiple destinations as:

```
DEST=(FLOOR3,SOUTH3RD,PERCIVAL)
```

When you code only one destination, parentheses are not necessary. For example:

```
DEST=FL00R3
```

- DEST=\*—specifies that the printer processes files regardless of destination.

**FCB=***fcname*

specifies the 1- to 4-character suffix of the name of the default forms control buffer (FCB) image to be used after a cold start or force start. This must correspond to one of the forms control buffer images added to an image library. For example, if you specify:

```
RDEVICE DEVNO=0008,DEVTYPE=4248,FCB=FCB8
```

the image library must contain a member named 4248FCB8.

**FEATURE=UNVCHSET**

specifies that the printer is a universal character set printer. FEATURE=UNVCHSET can be specified only for 3203 and 3211 printers.

**FOLD=NO**

**FOLD=Yes**

specifies whether lowercase should be folded (translated) into uppercase. The default is NO.

**FORM=***operform*

**FORM=\***

**FORM=STANDARD**

is the current spooling form number that the printer can process. This form implicitly determines 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=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. The default value is FORM=STANDARD.
- FORM=*operform*, where *operform* specifies a 1- to 8-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.

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.

**IMAGE=imagelib**

specifies the image library to be used after a cold start. The default is `IMAGnnnn`, where *nnnn* is the device type number.

**INDEX=nn****INDEX=1**

specifies the position at which to start printing, where *nn* can be from 1 to 31, 1 being the default. This parameter is only for devices that emulate the 3211.

**SEP=YES****SEP=NO**

specifies whether a separator is desired for output files. The default is YES.

**UIRATE=DEFAULT****UIRATE=INFINITY****UIRATE=value**

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no `UIRATE` parameter specified.

**INFINITY** turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the `UIRATE` value to **INFINITY** for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

**Examples**

1. To specify a 3203 printer at device number 0110 that prints only class A spool files (the default value), code the `RDEVICE` macro as follows:  

```
RDEVICE DEVNO=0110,DEVTYPE=3203,MODEL=5
```
2. To specify three 4245 printers at device numbers 0111, 0112, and 0113 that are not to be used to print spool files, code the `RDEVICE` macro as follows:  

```
RDEVICE DEVNO=(0111,3),DEVTYPE=4245,CLASS=NONE
```
3. To specify a 3262 printer at device number 0110 that prints only class A spool files (the default value), code the `RDEVICE` macro as follows:  

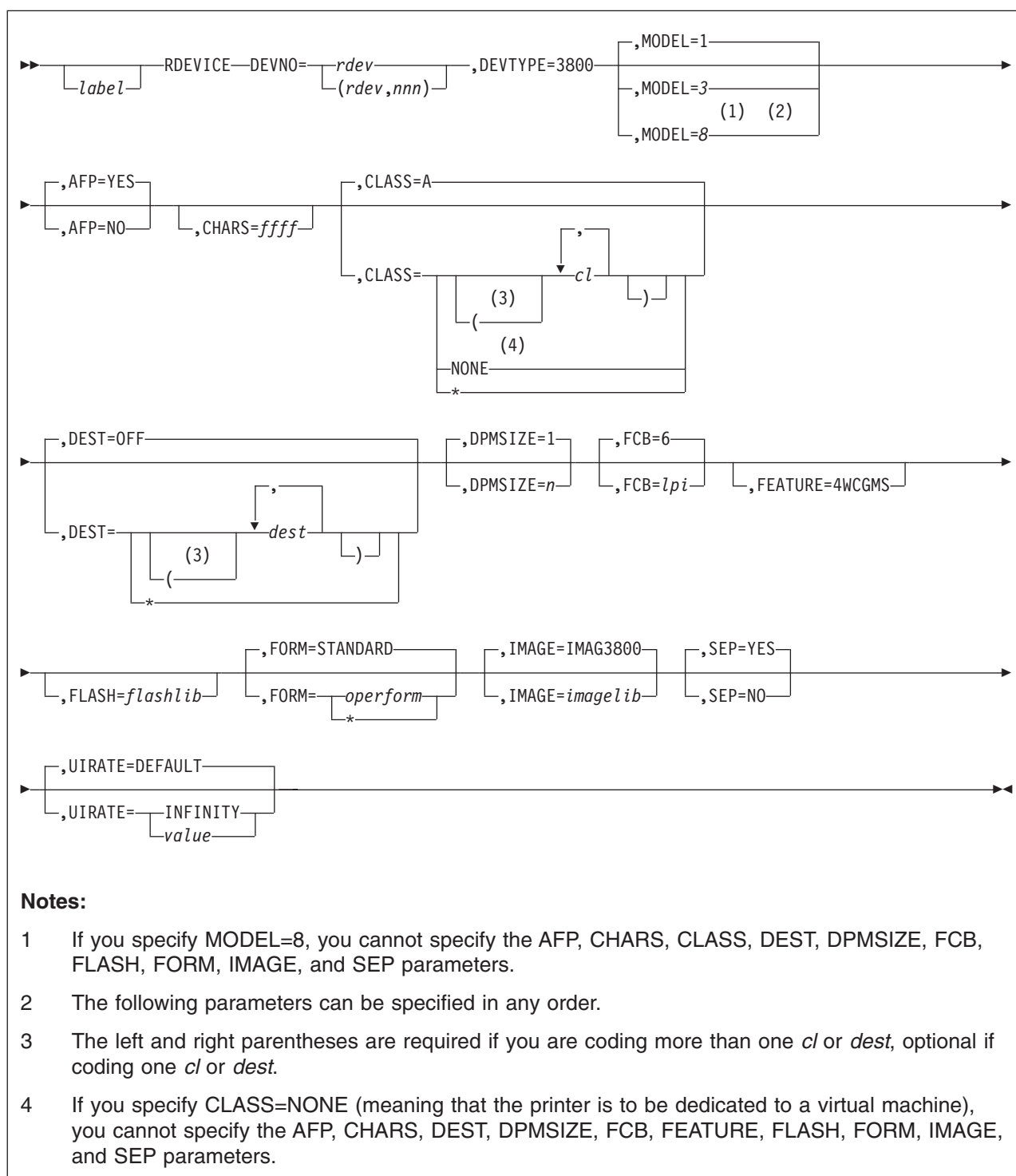
```
RDEVICE DEVNO=0110,DEVTYPE=3262,MODEL=5
```

---

**CP System-Managed 3800 Printers**



## 3800 Printer

**Purpose**

Use this RDEVICE format to define the CP system-managed 3800 printers.

### Parameters

**DEVNO=*rdev***

specifies a real I/O device number. The variable *rdev* must be a 1- to 4-digit hexadecimal number from X'0000' to X'FFFF'.

**DEVNO=(*rdev,nnn*)**

specifies the real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default is 1.

**DEVTYPE=3800**

specifies the device type.

**MODEL=1****MODEL=3****MODEL=8**

specifies the model number of the 3800 printer. The default is 1.

**Notes:**

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. If you are using them as advanced function printers, code AFP=YES.
2. The 3800 Model 8 is supported using an Advanced Function Printing licensed program; code AFP=YES.
3. If you specify MODEL=8, you cannot specify the following parameters: AFP, CHARS, CLASS, DEST, DPMSIZE, FCB, FLASH, FORM, IMAGE, and SEP.

**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.

**Note:** If you specify MODEL=8 or CLASS=NONE, you cannot specify AFP.

**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.

**Note:** If you specify MODEL=8 or CLASS=NONE, you cannot specify CHARS.

**CLASS=*cl*****CLASS=NONE****CLASS=\***

specifies the output spooling class or classes the 3800 Model 1, Model 3, or Model 6 printer can print. (To change the spooling classes without having to recode and reassemble the HCPRIO file, use the CP START UR command.)

You can specify CLASS in one of the following ways:

- If you omit CLASS, the default is class A.
- CLASS=*cl*,... to list up to eight output spooling classes, separated by commas. The spooling class, *cl*, is one alphanumeric character. If you code only one spooling class, you do not have to use parentheses. For example, you can specify multiple classes as CLASS=(A,B,C) and a single class as CLASS=A or CLASS=(A).

- CLASS=NONE specifies that CP will not use the printer for spooling.
- CLASS=\* specifies that the 3800 processes files regardless of class.

**Notes:**

1. If you specify MODEL=8, you cannot specify CLASS.
2. If you specify CLASS=NONE (meaning that the 3800 is to be dedicated to a virtual machine), you cannot specify the following parameters: AFP, CHARS, DEST, DPMSIZE, FCB, FEATURE, FLASH, FORM, IMAGE, and SEP.

**DEST=OFF****DEST=***dest***DEST=\***

specifies the output destination value or values the 3800 Model 1, Model 3, or Model 6 printer can print. (To change the spooling destination values without having to recode and reassemble the HCPRIO file, use the CP START UR command.)

You can specify DEST in one of the following ways:

- If you omit DEST, the default is OFF. A 3800 with DEST set to OFF will only process files with DEST specifically set to, or defaulted to, OFF.
- DEST=*dest*,... to list up to four output destinations, separated by commas. The destination value, *dest*, is a 1- to 8-character alphanumeric string. If you code only one destination, you do not have to use parentheses. For example, you can specify multiple destinations as DEST=(FLOOR3,SOUTH3RD,PERCIVAL) and a single destination as DEST=FLOOR3.
- DEST=\* specifies that the 3800 processes files regardless of destination.

**Note:** If you specify MODEL=8 or CLASS=NONE, you cannot specify DEST.

**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, 9.

**Notes:**

1. After the 3800 prints a file, CP places the file on the delayed purge queue. CP puts all files on the delayed purge queue in system hold status. 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.
2. If you specify MODEL=8 or CLASS=NONE, you cannot specify DPMSIZE.
3. 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 CP START UR command. The default value for *lpi* is 6.

**Note:** If you specify MODEL=8 or CLASS=NONE, you cannot specify FCB.

**FEATURE=4WCGMS**

specifies that the 3800 has the 4-writable-character generation module feature. For a 3800 model 1, the default is 2WCGMS. For a 3800 model 3 and model 8, the default is 4WCGMS.

**Note:** If you specify CLASS=NONE, you cannot specify FEATURE.

**FLASH=flashlib**

specifies the flash overlay for use with this device.

**Note:** If you specify MODEL=8 or CLASS=NONE, you cannot specify FLASH.

**FORM=STANDARD**

**FORM=operform**

**FORM=\***

is the current spooling form number that the printer can process. This form implicitly determines the default operator form for the real printer when the operator starts the device after a cold start without specifying a form.

You can specify FORM in one of the following ways:

- 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. The default is STANDARD.
- FORM=operform to specify a 1- to 8-character operator form number for the files the printer can process.
- FORM=\* specifies that the printer can process files regardless of form number.

**Note:** If you specify MODEL=8 or CLASS=NONE, you cannot specify FORM.

**IMAGE=IMAG3800**

**IMAGE=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 CP START UR command. The default is IMAG3800.

**Note:** If you specify MODEL=8 or CLASS=NONE, you cannot specify IMAGE.

**SEP=YES**

**SEP=NO**

tells CP whether the printer should use a separator page between output files. The default is YES.

**Note:** If you specify MODEL=8 or CLASS=NONE, you cannot specify SEP.

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=value**

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

### Examples

1. To specify a Model 3 3800 printer at device number 0500 that is to be dedicated to a virtual machine, use the following RDEVICE macroinstruction:

```
RDEVICE DEVNO=0500,DEVTYPE=3800,MODEL=3,CLASS=NONE
```

2. To specify three Model 8 3800 printers at device numbers 0501, 0502, and 0503 that are to be dedicated to virtual machines, use the following RDEVICE macroinstruction:

```
RDEVICE DEVNO=(0501,3),DEVTYPE=3800,MODEL=8
```

3. To specify a Model 1 3800 printer at device number 0505 that has all of the following characteristics:

- Is to be used to print spool files
- Prints any class spool files
- Has the 4-writable-character generation module (WCGM) feature
- Has a delayed purge queue length of 5

use the following RDEVICE macroinstruction:

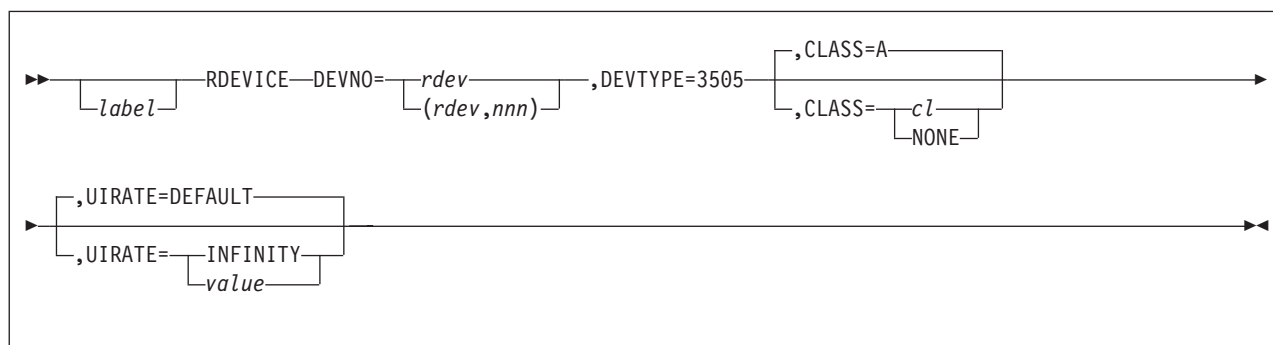
```
RDEVICE DEVNO=0505,DEVTYPE=3800,MODEL=1,CLASS=*,      *
        FEATURE=4WCGMS,DPMSIZE=5
```

**Note:** If you need more than one line to code the RDEVICE macroinstruction, you must code a continuation character in column 72 and start the next line in column 16.

---

## Unit Record Devices

## 3505 Card Reader

**Parameters****DEVNO**=*rdev***DEVNO**=(*rdev*,*nnn*)

**DEVNO**=*rdev* specifies a real I/O device number. It must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

**DEVNO**=(*rdev*,*nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default value for *nnn* is 1.

**DEVTYPE**=3505

specifies the type of device (3505).

**CLASS**=*cl***CLASS**=A**CLASS**=NONE

specifies the spooling class the card reader assigns to spool files it creates. (To change the spooling class the card reader assigns without recoding and reassembling the HCPRI0 file, use the CP START command.)

You can specify CLASS in one of the following ways:

- If you omit the CLASS operand, class A is assumed by default.
- CLASS=*cl*, where the spooling class, *cl*, is one alphanumeric character.
- CLASS=NONE to specify that CP will not use the card reader for spooling.

**UIRATE**=DEFAULT**UIRATE**=INFINITY**UIRATE**=*value*

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

## Unit Record Devices

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

### Examples

1. To specify three 3505 card readers at device numbers 00C1, 00C2, and 00C3 that are not to be used to read spool files, code the RDEVICE macro as follows:

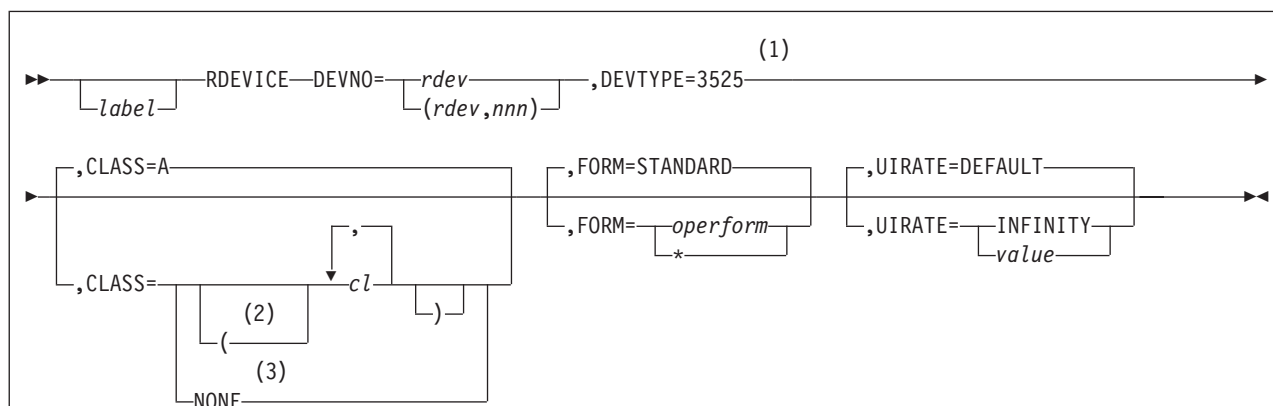
```
RDEVICE DEVNO=(00C1,3),DEVTYPE=3505,CLASS=NONE
```

2. To specify a 3505 card reader at device number 00C4 that reads information in as class R spool files, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=00C4,DEVTYPE=3505,CLASS=R
```



## 3525 Card Punch

**Notes:**

- 1 The following parameters may be specified in any order.
- 2 The left and right parentheses are required if you are coding more than one *cl* or optional if coding one *cl*.
- 3 If you specify CLASS=NONE (meaning that the punch is to be dedicated to a virtual machine), you cannot specify the FORM parameter.

**Parameters****DEVNO=rdev****DEVNO=(rdev,nnn)**

DEVNO=rdev specifies a real I/O device number. It must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(rdev,nnn) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default value for *nnn* is 1.

**DEVTYPE=3525**

specifies the type of device (3525). Note that the 3525 is supported only for punching; the printing features are not supported.

**CLASS=(cl)****CLASS=A****CLASS=\*****CLASS=NONE**

specifies the output spooling classes. (To change these spooling classes without recoding and reassembling the HCPRI0 file, use the CP START command.)

You can specify CLASS in one of the following ways:

- If you omit the CLASS operand, class A is assumed by default.
- CLASS=cl to list up to eight output spooling classes, separated by commas. The spooling class, *cl*, is one alphanumeric character. If you code only one spooling class, you do not have to use parentheses. For example, you can specify multiple classes as CLASS=(A,B,C) and a single class as CLASS=A or CLASS=(A).

## Unit Record Devices

- CLASS=NONE specifies that CP will not use the card punch for spooling.
- CLASS=\* specifies that the punch can process files regardless of class.

**Note:** If you specify CLASS=NONE (meaning that the punch is to be dedicated to a virtual machine), you cannot specify the FORM operand.

**FORM=***operform*

**FORM=\***

**FORM=STANDARD**

is the current spooling form number that the punch can process. This form implicitly determines the default operator form for the real punch when the operator starts the device after a cold start without specifying a form. Specify this operand in one of the following ways:

- FORM=*operform*, where *operform* specifies a 1- to 8-character operator form number for the files the punch can process.
- FORM=\*, where the asterisk specifies that the punch can process files regardless of form number.
- Do not specify FORM. The default value is FORM=STANDARD.

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=***value*

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

### Examples

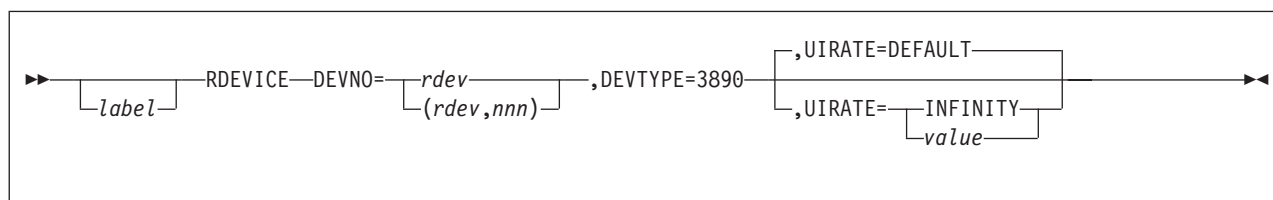
1. To specify a 3525 punch at device number 00D0 that punches only class A spool files (the default value), code the RDEVICE macro as follows:  

```
RDEVICE DEVNO=00D0,DEVTYPE=3525
```
2. To specify three 3525 punches at device numbers 00D1, 00D2, and 00D3 that are not to be used to punch spool files, code the RDEVICE macro as follows:  

```
RDEVICE DEVNO=(00D1,3),DEVTYPE=3525,CLASS=NONE
```
3. To specify a 3525 punch at device number 00D4 that punches class E, F, and G spool files, code the RDEVICE macro as follows:  

```
RDEVICE DEVNO=00D4,DEVTYPE=3525,CLASS=(E,F,G)
```

## 3890 Document Processor



### Purpose

The 3890™ Document Processor is supported as a dedicated-only device.

### Parameters

**DEVNO=*rdev***

**DEVNO=(*rdev,nnn*)**

DEVNO=*rdev* specifies a real I/O device number. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev,nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default value for *nnn* is 1.

**DEVTYPE=3890**

specifies the type of device (3890).

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=*value***

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

### Examples

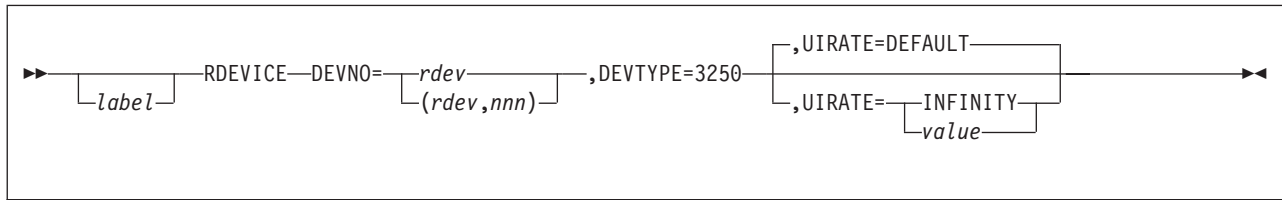
To specify a 3890 document processor at device number 100,

```
RDEVICE DEVNO=(100),DEVTYPE=3890
```

---

## Display Terminals and Adapters

## 3250 Display



### Purpose

The 3250 is supported as a dedicated-only device.

### Parameters

**DEVNO=*rdev***

**DEVNO=(*rdev,nnn*)**

DEVNO=*rdev* specifies a real I/O device number. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev,nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default value for *nnn* is 1.

**DEVTYPE=3250**

specifies the type of device (3250).

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=*value***

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

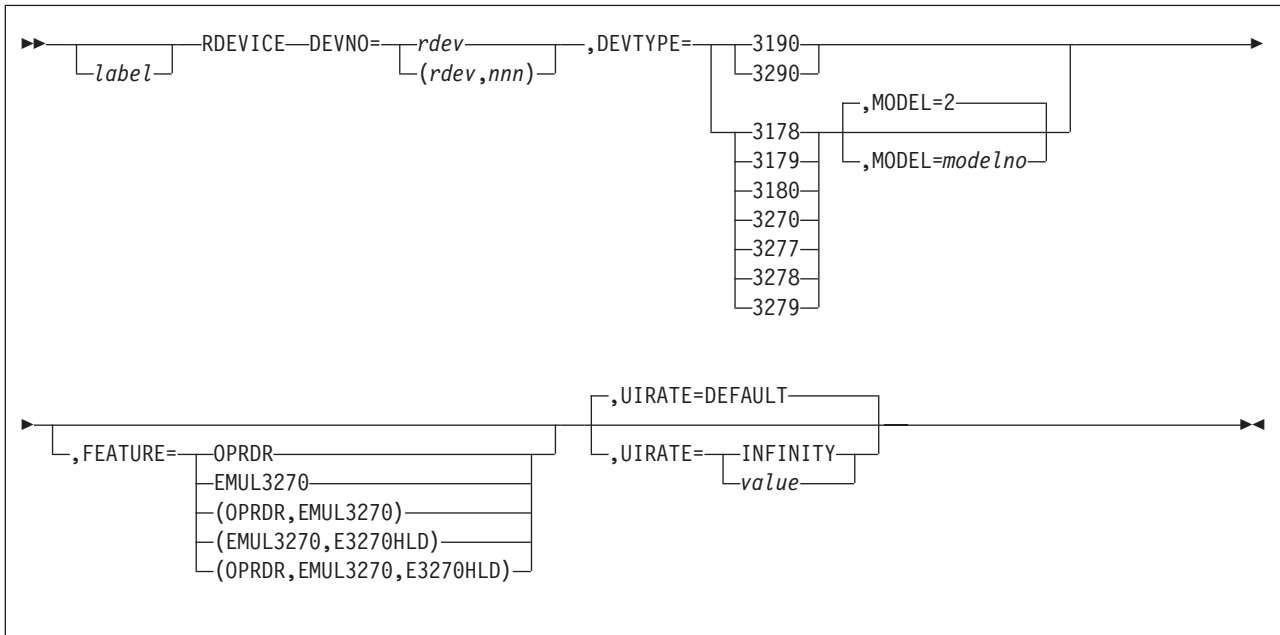
**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great length of time. Hot I/O detection protects CP from broken hardware that floods the system with unsolicited interrupts. If you turn Hot I/O detection off, you may experience performance degradation or a system abend.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

### Examples

1. To specify a 3250 display at device number 0006, code the RDEVICE macro as follows: RDEVICE DEVNO=0006,DEVTYPE=3250
2. To specify three 3250 displays at device numbers 0007, 0008, and 0009, code the RDEVICE macro as follows: RDEVICE DEVNO=(007,3),DEVTYPE=3250

## 3270-Family Displays



### Parameters

#### DEVNO=*rdev*

is the real I/O device number. The variable *rdev* must be a 1- to 4-digit hexadecimal number from X'0000' to X'FFFF'.

#### DEVNO=(*rdev,nnn*)

are the real I/O device numbers for a range of devices. The variable *rdev* must be a 1- to 4-digit hexadecimal number from X'0000' to X'FFFF'. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. If omitted, the default for *nnn* is 1.

#### DEVTYPE=*devtype*

is the type of device. Valid device types are: 3178, 3179, 3180, 3190, 3270, 3277, 3278, 3279, or 3290. However, CP only recognizes two categories of display devices, 3277 and all others. The 3277 must be specified as a 3277; other device types may be specified as any of the other valid device types. For example, a 3191 display could be specified as a 3270, a 3190, or any other valid device number, except 3277.

#### Notes:

1. The 3270-PC is supported in control-unit terminal mode and distributed-function terminal (DTF) mode when defined as a 3270.
2. The 3101 must be connected to a 7171 Control Unit. You must also specify FEATURE=EMUL3270.
3. If you want to see what other displays are supported, see the *z/VM: General Information* book, which lists all displays supported for this release.

#### MODEL=*modelno* (do not specify for a 3190 or 3290)

specifies the model number which determines the display screen format. If omitted, the default is 2. Note that CP determines the real size of the screen when the terminal is powered-on or enabled. Table 35 on page 735 shows the options. (Also see Usage Note 2 on page 736.)

Table 35. Model Options Depending on Screen Size

Model ( <i>modelno</i> )	Number of Characters	Screen Size	Valid for DEVTYPE=
2	1920	24 x 80	3178, 3179, 3180, 3191, 3192, 3193, 3194, 3270, 3277, 3278, 3279
2A	1600	20 x 80	3270, 3278
2C	1600	20 x 80	3270, 3279
3	2560	32 x 80	3178, 3179, 3180, 3192, 3193, 3270, 3278, 3279
4	3440	43 x 80	3178, 3180, 3192, 3193, 3270, 3278
5	3564	27 x 132	3178, 3180, 3270, 3278

The model number is set to 2 if the terminal is attached to a 7171 Device Attachment Control Unit or ASCII Subsystem (EMUL3270 is specified).

### FEATURE=

specifies one to three display features. The features can be coded in any order. If you specify more than one feature, you must separate the features with a comma and surround them by parenthesis. The features are:

#### OPRDR

specifies an operator identification card reader on a 327X display. If you specify FEATURE=OPRDR, the virtual machine operator can gain access to the system (log on) only by inserting a magnetically encoded card. Using the badge reader is optional for each user ID and not required. It cannot be used instead of a correct password but may be used after a correct password is supplied for an additional security measure. If you do not want access authorization by a card reader, do not code FEATURE=OPRDR.

#### EMUL3270

specifies that the device is a TTY ASCII display terminal connected to the system through a 7171 ASCII DACU or ASCII Subsystem (an emulated 3270).

#### E3270HLD

specifies that the display terminal telecommunications connection to the 7171 ASCII DACU or ASCII Subsystem is to remain after logging off or disconnecting. E3270HLD is valid only if EMUL3270 is also coded.

### UIRATE=

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

### Usage Notes

1. If you specify DEVTYPE=3270, device type 3278 is generated in the HCPRI0 file. All device types except for the 3277 are equivalent to device type 3278.
2. The MODEL=*modelno* is optional. If not specified, MODEL=2 is the default. Otherwise, the model specified is generated. The actual value of MODEL is determined by CP during display initialization and is set accordingly. The MODEL parameter is used when CP cannot determine the real device characteristics.

Note that for the RDEVICE macro, the model number refers to the display screen format as shown in Table 35 on page 735. The model number, as specified on the RDEVICE statement, may or may not correspond to the model number on the display itself. MODEL defaults to 2 if the screen size does not match 2A, 2C, 3, 4, or 5.

For example, a 3192 that has the same screen size as a 3278 Model 2 can be generated that way or it can be generated as a 3190 (with no model specified). A 3192 that is the same size as a 3278 Model 3 should be generated that way or as a 3190 (with no model specified).

A 3192 that is the same size as a 3278 Model 4 should be generated that way or as a 3190 (with no model specified).

All of the above is true for 3191, 3192, 3193, and 3194.

The RDEVICE macroinstruction only accepts models types of 2, 3, 4, 5, 2A, or 2C. Other model types that are supported for various terminal types include:

```
3178 C1,C2,C3,C4
3179 1,G1,G2
3180 1
3279 2B,2X,3A,...
3290 220,230,T30
```

These terminals and models are supported but can only be coded as model types 2, 3, 4, 5, 2A, or 2C. However, when they are logged on to, CP determines the real size of the screen and uses it.

3. You must ensure that the device numbers specified by the DEVNO parameter correspond to the device numbers required for that category of display terminal in the controller.
4. Refer to the *7171 ASCII Device Attachment Control Unit Description and Planning Guide* to determine which type of device (DEVTYPE) to code for a TTY ASCII terminal attached to a 7171 DACU or ASCII Subsystem.

### Examples

1. To specify a 3277 display at device number 0140, code the RDEVICE macroinstruction as follows:

```
RDEVICE DEVNO=0140,DEVTYPE=3277
```

2. To specify Model 2A 3278 displays at device numbers 0141, 0142, and 0143 that do use the operator identification card reader, code one of the following RDEVICE macroinstructions:

```
RDEVICE DEVNO=(0141,3),DEVTYPE=3278,MODEL=2A,FEATURE=OPRDR
RDEVICE DEVNO=(141,3),DEVTYPE=3278, FEATURE=OPRDR
```

3. To specify a 3279 Model 3 display device number 0144 that does not use the operator identification card reader, code the RDEVICE macroinstruction as follows:

```
RDEVICE DEVNO=0144,DEVTYPE=3279,MODEL=3
```



4. To specify a 3180 Model 1 display with a screen format of 43 by 80, code the RDEVICE macroinstruction as follows:

```
RDEVICE DEVNO=0145,DEVTYPE=3278,MODEL=4
```

5. To specify a 3179 Model 1 display, code the RDEVICE macroinstruction as follows:

```
RDEVICE DEVNO=0146,DEVTYPE=3279,MODEL=2
```

6. To specify a 3290 display at device number 0145, code the RDEVICE macroinstruction as follows:

```
RDEVICE DEVNO=0145,DEVTYPE=3290
```

7. To specify a series of 3278 or 3279 displays that are a variety of models, beginning at device number 2C0, code the RDEVICE macroinstruction as follows:

```
RDEVICE DEVNO=(2C0,8),DEVTYPE=3270
```

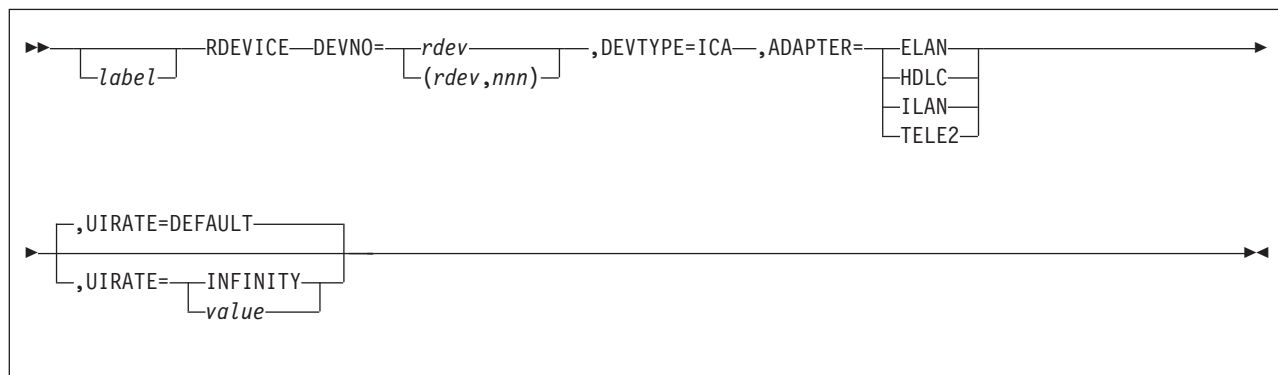
8. To specify a TTY device attached to a 7171 DACU or ASCII Subsystem at device number FE0 that emulates a 3278 and to which the telecommunication line is always held, code the RDEVICE macroinstruction as follows:

```
RDEVICE DEVNO=FE0,DEVTYPE=3278,FEATURE=(EMUL3270,E3270HLD)
```

9. To specify a TTY device attached to a 7171 DACU or ASCII Subsystem at device number 220 that emulates a 3277 and for which the user controls the holding of the telecommunication line, code the RDEVICE macroinstruction as follows:

```
RDEVICE DEVNO=220,DEVTYPE=3277,FEATURE=EMUL3270
```

## Teleprocessing Integrated Adapters



### Parameters

**DEVNO=*rdev***

**DEVNO=(*rdev,nnn*)**

specifies a real I/O device number or numbers. The variable *rdev* must be a 1- to 4-digit hexadecimal number in the range X'0000' through X'FFFF'. If you are specifying a range of devices using DEVNO=(*rdev,nnn*), *rdev* is the first real I/O device in the range and *nnn* is a decimal number indicating the total number of consecutive devices. The default for *nnn* is 1.

**DEVTYPE=ICA**

tells CP that the device is an integrated communication adapter.

**ADAPTER=**

tells CP what teleprocessing protocol being defined. The valid protocols and their meanings are:

**ELAN**

802.3 local area network

**HDLC**

High-level data link control (X.25 network)

**ILAN**

IBM Token Ring Local Area Network

**TELE2**

U.S. Telegraph Terminal Control—type 2.

**UIRATE=**

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

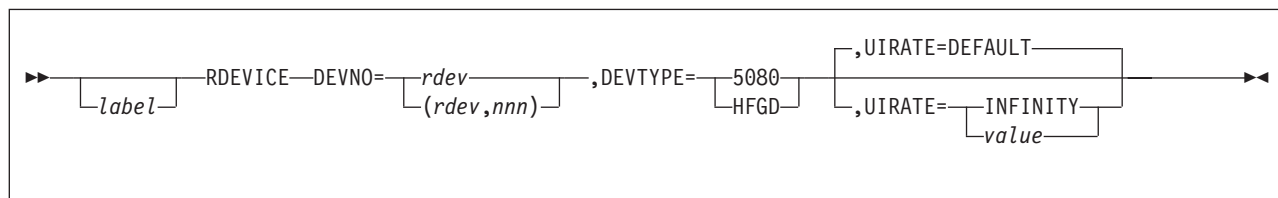
*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

### Examples

To specify an integrated adapter attachment to an IBM Token Ring Local Area Network, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=(220,12),DEVTYPE=ICA,ADAPTER=ILAN
```

## 5080 and 6090 Graphics Display



### Purpose

The 5080 is supported as a dedicated-only device.

### Parameters

**DEVNO=*rdev***

**DEVNO=(*rdev,nnn*)**

DEVNO=*rdev* specifies a real I/O device number. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev,nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default is 1.

**DEVTYPE=5080**

**DEVTYPE=HFGD**

specifies the type of device. HFGD (high function graphics device) is interchangeable with 5080. It is listed here for compatibility with VM/SP HPO. To specify 6090, code 5080.

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=*value***

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

### Examples

1. To specify a 5080 display at device number 0006, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=0006,DEVTYPE=5080 (or DEVTYPE=HFGD)
```

2. To specify three 5080 displays at device numbers 0007, 0008, and 0009, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=(0007,3),DEVTYPE=5080 (or DEVTYPE=HFGD)
```

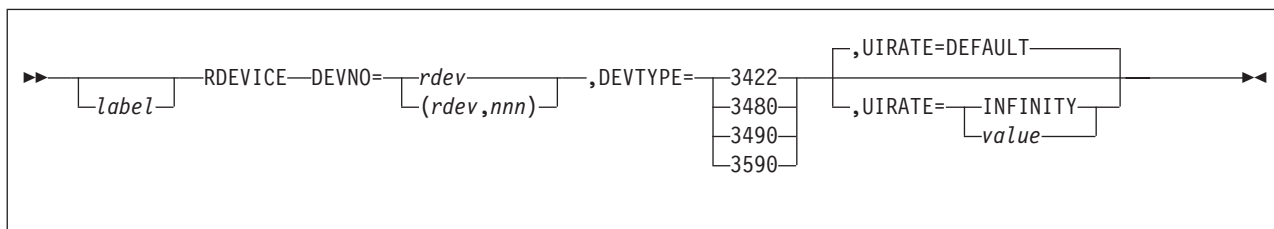
3. To specify a 6090 display at device numbers 0010, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=0010,DEVTYPE=5080
```

---

## Tape Units

## 3422, 3480, 3490 and 3590 Tape Drives



### Parameters

**DEVNO=*rdev***

**DEVNO=(*rdev*,*nnn*)**

DEVNO=*rdev* specifies a real I/O device number. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev*,*nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default is 1.

**DEVTYPE=*devtype***

specifies the type of device (3422, 3480 or 3490).

For 3480s and 3490s, specify the *devtype* based on the model of your tape device:

Device	Model	Specify Type
3480	A11, B11, A22, B22	3480
3490	A01, A02, B02, B04, D31, D32	3480
3490	A10, A20, B20, B40, C10, C11, C22, D41, D42	3490

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=*value***

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

### Examples

- To specify a 3422 tape drive at device member 0422, code the RDEVICE macro as follows:

## Tape Units

```
RDEVICE DEVNO=0422,DEVTYPE=3422
```

2. To specify a 3480 tape drive at device number 0450, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=0450,DEVTYPE=3480
```

3. To specify a 3490 tape drive, Model A01, A02, B02, B04, D31, or D32, at device number 0462, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=0462,DEVTYPE=3490
```

4. To specify a 3490 tape drive Model A10, A20, B20, B40, D41, or D42, at device number 0480, code the RDEVICE macro as follows:

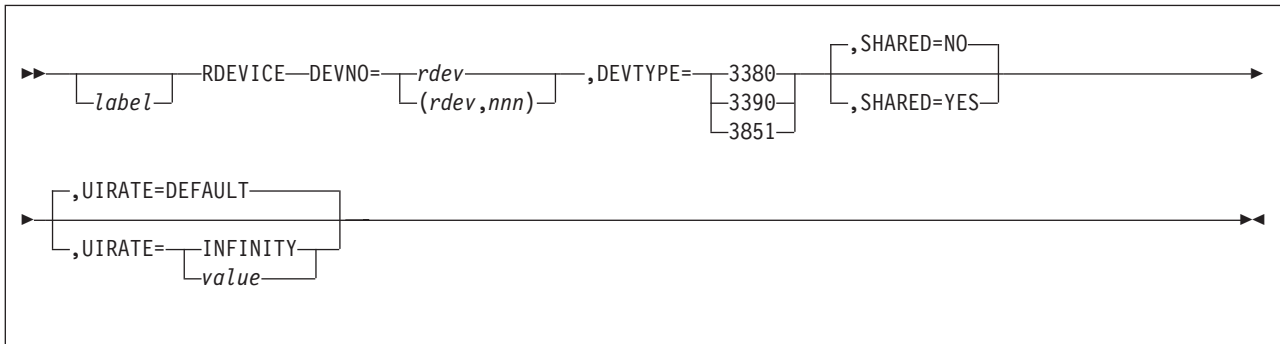
```
RDEVICE DEVNO=0480,DEVTYPE=3490
```



---

**DASD**

## DASD Types



### Purpose

This section describes how to code the RDEVICE macro for the following devices:

3380 DASD

3390 DASD

3850 Mass Storage System

See also “FBA DASD” on page 749.

### Notes:

1. A 3380 attached to a 3880-11 or 3880-13 requires no special treatment.
2. 3995 Models 131, 132 (Optical Library Dataserver), look like 3088's and must be defined as such. For a description of how to code RDEVICE for a 3088 see “CTCA, 3088 Multisystem Channel Communication Unit, and 3737 Remote Channel-to-Channel Unit” on page 760.
3. 3995 Models 151, 152 (Optical Library Dataserver), look like 3390's and must be defined as such.

### Parameters

**DEVNO=rdev**

**DEVNO=(rdev,nnn)**

DEVNO=rdev specifies a real I/O device number. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(rdev,nnn) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default value for *nnn* is 1.

**DEVTYPE=devtype**

specifies the device type plus model for 3330. Code this parameter as follows:

For a	Specify
3380	DEVTYPE=3380 (The use of the CUTYPE keyword is ignored.)
3390	DEVTYPE=3390 (The use of the CUTYPE keyword is ignored.)

For a	Specify
3390 in 3380 track compatibility mode	DEVTYPE=3380 or 3390 except when device is used for the SYSRES device. If the IPL device is operating in 3380 track compatibility mode, you must specify DEVTYPE=3380 and on the SYSRES macro SYSTYPE=3380.
3850	DEVTYPE=3851.

**SHARED=YES****SHARED=NO**

indicates whether the device is to be shared concurrently between multiple real systems. The default is SHARED=NO. For more information on sharing DASDs, see Chapter 21, "DASD Sharing," on page 619.

**Note:** VM/ESA supports the 3850 Mass Storage System only when it is dedicated to a virtual machine. Because hardware that is supported as dedicated-only cannot be shared concurrently between multiple real systems, do not code the SHARED operand for the 3851 Mass Storage Facility nor for the DASD attached to the 3850 Mass Storage System. Devices that contain CMS minidisks that will be shared in a CSE or VM/ISF complex should be specified as SHARED=YES (see "Spool File Directory Statements" on page 341).

**UIRATE=DEFAULT****UIRATE=INFINITY****UIRATE=value**

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

**Usage Notes**

1. If you generate a 3390 DASD attached to a 3990 Models 3 or 6 Storage Control, and that device is physically attached to the system, the operator may request I/O to an offline device.
2. The following commands may be entered to an offline device:
  - SET CACHE
  - SET DASDFW
  - DISCARD
  - QUERY PINNED
  - QUERY RSAW
  - QUERY CACHE

## DASD

- QUERY DASDFW.

**Note:** The only way to ensure that the operating system never requests I/O to a device is not to generate it, or to ensure that it is physically inaccessible.

### Restrictions

1. DASDs on ESCON channels must be formatted without filler records. This is because you must use ECKD channel programs to control DASD on an ESCON channel and the ECKD command, LOCATE RECORD, may fail if there are filler records. Also, system performance will be seriously degraded if you issue non-ECKD channel programs to an ESCON-attached DASD.
2. Any minidisks being used by the shared file system (SFS) that are participating in virtual machine data-space mappings that are on ESCON channels must be formatted without filler records because virtual machine data-space mapped minidisks use CP paging routines to perform I/O.

There are two cases depending on the ownership of the ESCON attached volume:

- a. CP owned:

Cylinder 0 and all CP-owned extents must have been formatted for CP. CP will do I/O operations to the volume based on how cylinder 0 was formatted; that is, with or without filler records. If the volume is a 3380 formatted with filler records, paging forces CKD usage for the packs and CKD usage will cause command rejects.

- b. Non CP-owned:

The format of the volume is not known because the mapped minidisk is formatted however the user or application wanted. CP will always use ECKD channel programs when doing mapped minidisk paging. This may cause the ECKD command, LOCATE RECORD, to fail if there are filler records.

3. The 3990 Models 3 and 6 Storage Control are track caching control units. Therefore, it is not recommended that DASD attached to 3990 Models 3 and 6 Storage Control units be used for paging, or if they are, the cache should be turned off for any paging devices attached to them.
4. If you issue QUIESCE/RESUME while running VM/ESA with a 3990 Storage Control attached, you will receive a message indicating that these commands are not supported.

### Examples

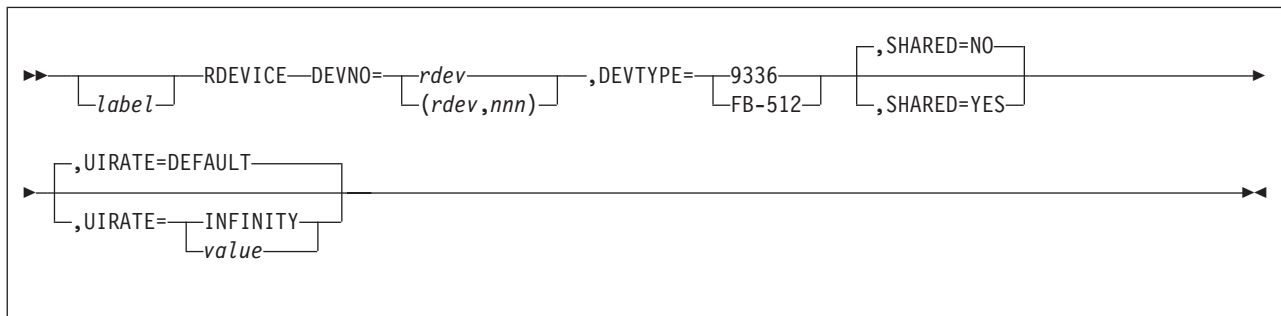
1. To specify a Model AE4 or BE4 3380 DASD at device number 0A70, code the RDEVICE macro as follows:  

```
RDEVICE DEVNO=0A70,DEVTYPE=3380
```
2. To specify a 3390 DASD at device number 0A80, code the RDEVICE macro as follows:  

```
RDEVICE DEVNO=0A80,DEVTYPE=3390
```
3. To specify the 3851 Mass Storage Facility of the 3850 Mass Storage System at device number 0A90, code the RDEVICE macro as follows:  

```
RDEVICE DEVNO=0A90,DEVTYPE=3851
```

## FBA DASD



### Purpose

This section describes how to code the RDEVICE macro for FBA DASD.

### Parameters

**DEVNO=*rdev***

**DEVNO=(*rdev,nnn*)**

DEVNO=*rdev* specifies a real I/O device number. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev,nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device specifications with consecutive device numbers. The default is 1.

**DEVTYPE=9336**

**DEVTYPE=FB-512**

specifies the type of device. If you specify 9336, this indicates the specific FBA device type. If you specify FB-512, this implies that the device is one of the currently supported FBA device types.

**SHARED=YES**

**SHARED=NO**

indicates whether the device is to be shared concurrently between multiple real systems. The default is SHARED=NO.

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=*value***

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

## DASD

### Usage Notes

1. During device initialization time, the device model information will be obtained dynamically from the hardware. If the MODEL= parameter is specified, it will be ignored.
2. If execution of the Read Device Characteristics (RDC) CCW is unsuccessful during DASD initialization time, the device will be brought online as dedicated supported.

### Examples

1. To specify an FBA DASD generically at device number 220, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=220,DEVTYPE=FB-512
```

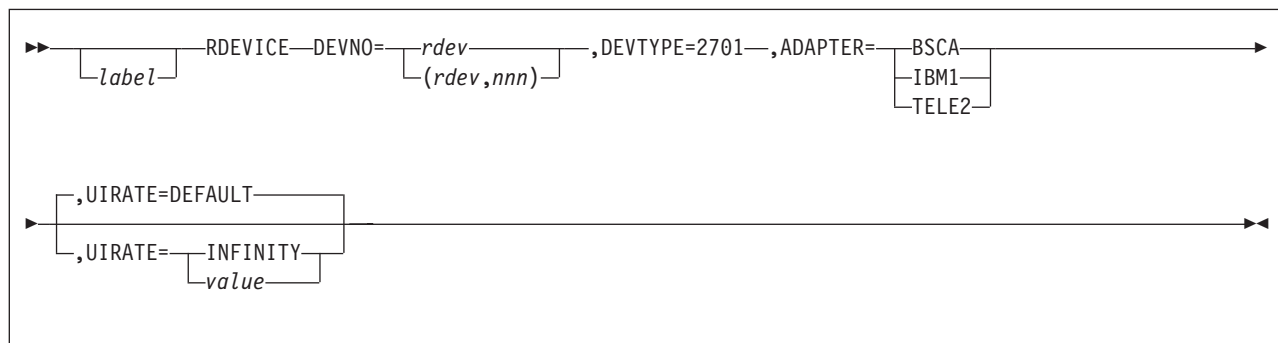
2. To specify a 9336 DASD at device numbers 0336, 0337, and 0338, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=(336,3),DEVTYPE=9336,SHARED=YES
```

---

## Communication Devices

## 2701 Data Adapter



### Parameters

**DEVNO=*rdev***

**DEVNO=(*rdev*,*nnn*)**

DEVNO=*rdev* specifies a real I/O device number. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev*,*nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default is 1.

**DEVTYPE=2701**

specifies the type of device.

**ADAPTER=BSCA**

**ADAPTER=IBM1**

**ADAPTER=TELE2**

specifies the terminal control adapter that connects to the device it controls.

The BSCA adapter is supported as a dedicated-only device.

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=*value***

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

### Examples

- To specify a 2701 line adapter at device number 00F0, code the RDEVICE macro as follows:

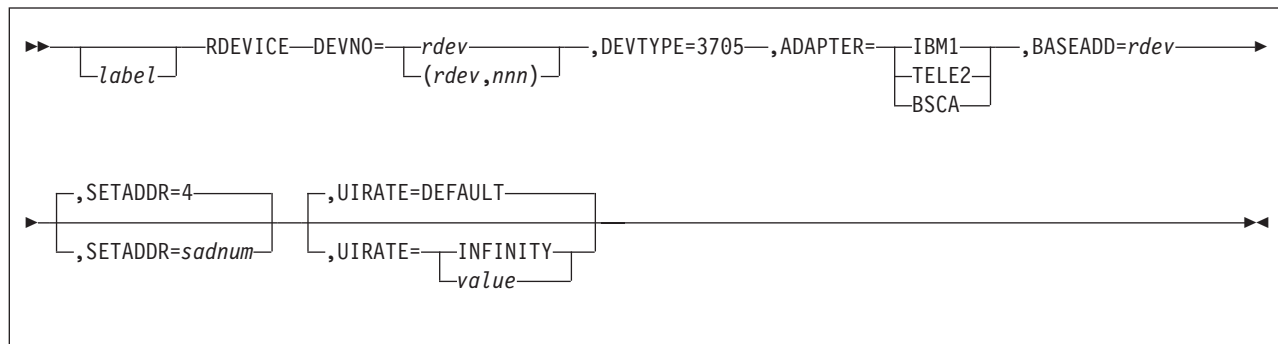


```
RDEVICE DEVNO=00F0,DEVTYPE=2701,ADAPTER=BSCA
```

2. To specify three 2701 line adapters at device numbers 00F1, 00F2, and 00F3, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=(00F1,3),DEVTYPE=2701,ADAPTER=BSCA
```

## 3705, 3725, and 3745 Line Adapters, 2741 Terminal, and 3151, 3161, 3162, 3163, and 3167 Display Terminals



### Purpose

**Note:** To define 3705, 3720, 3725, and 3745 communication controllers, see “37xx Communications Controllers and 3x74 Controllers” on page 757.

### Parameters

**DEVNO**=*rdev*

**DEVNO**=(*rdev,nnn*)

DEVNO=*rdev* specifies a real I/O device number. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev,nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default is 1.

**DEVTYPE**=*devtype*

specifies the type of device. For 3705, 3725, or 3745 line adapters, specify DEVTYPE=3705. For a 2741, 3151, 3161, 3162, 3163, or 3167, specify DEVTYPE=3705.

**ADAPTER**=IBM1

**ADAPTER**=TELE2

**ADAPTER**=BSCA

is the terminal control or transmission adapter used to connect a telecommunication I/O device to its control unit. Specify one of the following values for adapter:

- IBM1 specifies that an IBM Terminal Adapter Type I attaches a 1050 or 2741 to a 3705, 3725, or 3745 (generated as a 3705).
- TELE2 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 3705, 3725, or 3745 (generated as a 3705).
- BSCA specifies an IBM Binary Synchronous Terminal Control Type II for a 3705, 3725, or 3745 (generated as a 3705).

The BSCA adapter is supported as dedicated-only.

**BASEADD**=*rdev*

is the one- to four-digit hexadecimal device number of the communications controller to which the line adapter attaches.

**SETADDR=*sadnum***

is the set address (SAD) command to be entered for a telecommunication line attached to a control unit. The default is 4.

*sadnum*

	<b>Command</b>
<b>0</b>	SADZERO
<b>1</b>	SADONE
<b>2</b>	SADTWO
<b>3</b>	SADTHREE
<b>4</b>	(No SAD command is entered)

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=*value***

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

**Usage Notes**

VM/ESA supports native ASCII devices.

**Examples**

1. To specify at device number 0451 an IBM Binary Synchronous Terminal Control Type II that:
  - a. Attaches to the 3705 communication controller at device number 0441
  - b. Enters the SADZERO command,
 code the RDEVICE macro as follows:
 

```
RDEVICE DEVNO=0451,DEVTYPE=3705,ADAPTER=BSCA,      *
      BASEADD=0441,SETADDR=0
```
2. To specify at device number 0452 an IBM Terminal Adapter Type I that:
  - a. Attaches a 1050 to the 3725 communication controller at device number 0442
  - b. Enters the SADONE command,
 code the RDEVICE macro as follows:
 

```
RDEVICE DEVNO=0452,DEVTYPE=3705,ADAPTER=IBM1,      *
      BASEADD=0442,SETADDR=1
```
3. To specify a 2741 at device number 0460, code the RDEVICE macro as follows:
 

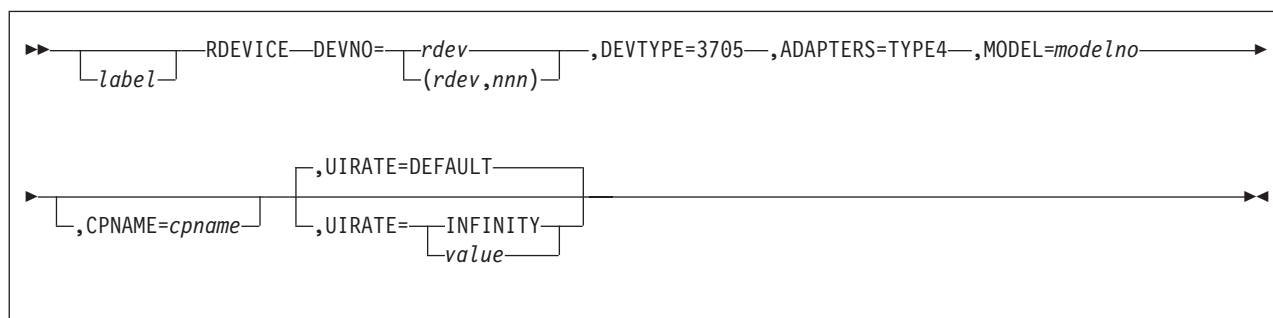
```
RDEVICE DEVNO=0460,DEVTYPE=3705,ADAPTER=IBM1,BASEADD=0422
```
4. To specify a 3151 at device number 0462, code the RDEVICE macro as follows:
 

```
RDEVICE DEVNO=0462,DEVTYPE=3705,ADAPTER=TELE2,BASEADD=0422
```

## Communication Devices

**Note:** If you need more than one line to code the RDEVICE macro, you *must* code a continuation character in column 72 *and* start the next line in column 16.

## 37xx Communications Controllers and 3x74 Controllers



### Purpose

The 370x communication controllers are supported as dedicated-only devices or by SNA attachment through VM/VTAM or VCNA. The 3720, 3725, and 3745, if specified as dedicated-only devices, must be specified as 3705s.

The 3x74 controllers are supported as follows:

#### 3174-1L, -11L

Through non-SNA channel attachment or through VM/VTAM or VCNA

#### 3174-1R, 51R, 81R, 11R, 61R, 91R

Through remote SNA attachment through VM/VTAM or VCNA or dedicated-only for remote non-SNA attachment

#### 3174-2R, 3R, 52R, 53R, 82R, 12R, 13R, 62R, 63R, 92R

Through remote SNA attachment through VM/VTAM or VCNA

#### 3274-1A, 21A, 31A, 31A, 41A

Through SNA channel attachment through VM/VTAM or VCNA

#### 3274 1B, 21B, 1D, 21D, 31D, 41D

Full support for non-SNA attachment

#### 3274-1C, 21C, 31C, 51C

Dedicated-only for non-SNA attachment

#### 3274-1C, 21C, 31C, 41C, 51C, 61C

Through remote SNA attachment through VM/VTAM or VCNA

### Parameters

**DEVNO=rdev**

**DEVNO=(rdev,nnn)**

DEVNO=rdev specifies a real I/O device number. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(rdev,nnn) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default is 1.

#### DEVTYPE=3705

specifies the type of device. For a 3174, 3274, 3705, 3720, 3725, or 3745, specify DEVTYPE=3705.

#### ADAPTER=TYPE4

specifies the transmission adapter used to connect a communication controller to the channel. For a 3174, 3274, 3705, 3720, 3725, or 3745, specify ADAPTER=TYPE4.

## Communication Devices

### **MODEL=***modelno*

specifies the model code for the device or device group you are defining. The model number can be any model number that is valid for the device or device group as follows:

```
A1 A2
B1 B2 B3 B4
C1 C2 C3 C4 C5 C6
D1 D2 D3 D4 D5 D6 D7 D8
E1 E2 E3 E4 E5 E6 E7 E8
F1 F2 F3 F4 F5 F6 F7 F8
G1 G2 G3 G4 G5 G6 G7 G8
H1 H2 H3 H4 H5 H6 H7 H8
J1 J2 J3 J4
K1 K2 K3 K4
L1 L2 L3 L4
1 2 3 4 5 6 7 8
```

### **CPNAME=***cpname*

is the 1- to 8-character name of an emulation program. Note that CP will not automatically load this (see “Setting Up a Virtual Machine for Communication Controller Support for Emulator Program (EP)” on page 321).

### **UIRATE=DEFAULT**

### **UIRATE=INFINITY**

### **UIRATE=***value*

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

**INFINITY** turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to **INFINITY** for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

## Usage Notes

1. z/VM supports the 3174 Subsystem Control Unit Model 1L for local channel attachment. For bisynchronous communications attachment, z/VM supports the Models 1R, 51R, and 81R as dedicated-only through the VM/Pass-Through Facility (PVM) Release 3 or through RSCS networking.
2. To define 3705, 3725, and 3745 line adapters, see “3705, 3725, and 3745 Line Adapters, 2741 Terminal, and 3151, 3161, 3162, 3163, and 3167 Display Terminals” on page 754.

## Examples

1. To specify three Model B1 3705 communications controllers at device numbers 0462, 0463, and 0464 that have emulation programs named X6000, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=(0462,3),DEVTYPE=3705,MODEL=B1,      *
        ADAPTER=TYPE4,CPNAME=X6000
```

**Note:** If you need more than one line to code the RDEVICE macro, you *must* code a continuation character in column 72 *and* start the next line in column 16.

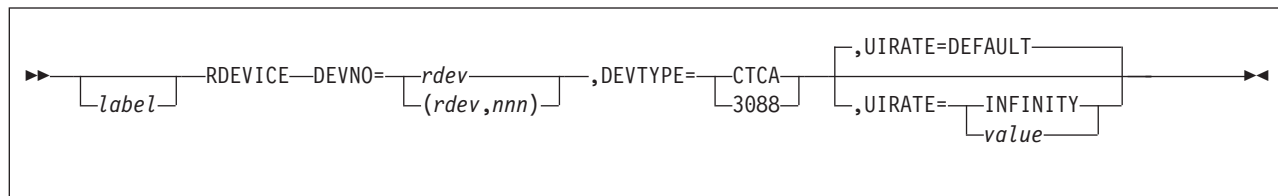
2. To specify a 3725 as a Model D1 3705 communications controller at device number 0465, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=0465,DEVTYPE=3705,MODEL=D1,ADAPTER=TYPE4
```

3. To specify a Model 41A 3274 controller at device number 0E0, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=0E0,DEVTYPE=3705,ADAPTER=TYPE4,MODEL=E8
```

## CTCA, 3088 Multisystem Channel Communication Unit, and 3737 Remote Channel-to-Channel Unit



### Purpose

The CTCA, 3088 MCCU, and 3737 are supported as dedicated-only devices.

### Parameters

**DEVNO=*rdev***

**DEVNO=(*rdev,nnn*)**

DEVNO=*rdev* specifies a real I/O device number. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev,nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default value is 1.

**DEVTYPE=CTCA**

**DEVTYPE=3088**

specifies the type of device. Specify the 3737 Remote Channel-to-Channel Unit Model 2 as a CTCA.

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=*value***

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

### Examples

1. To specify a 3088 multisystem channel communication unit logical channel adapter at device number 0100, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=0100,DEVTYPE=3088
```

2. To specify CTCAs at device numbers 0101, 0102, and 0103, code the RDEVICE macro as follows:

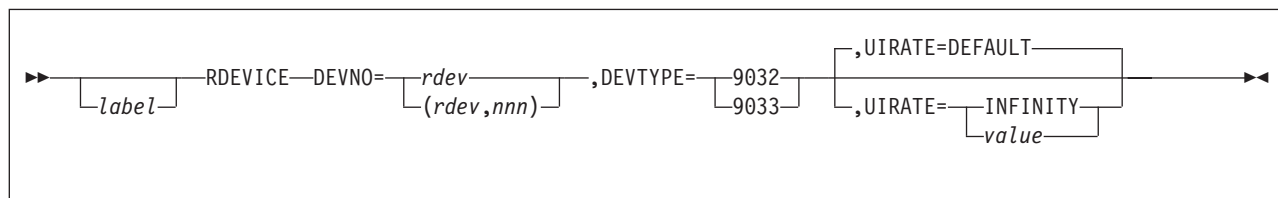
```
RDEVICE DEVNO=(0101,3),DEVTYPE=CTCA
```



---

## Dynamic Switching Devices

## 9032 and 9033 ES Connection Directors (ESCDs)



### Parameters

**DEVNO=*rdev***

**DEVNO=(*rdev*,*nnn*)**

DEVNO=*rdev* specifies a real I/O device number. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev*,*nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default is 1.

**DEVTYPE=9032**

**DEVTYPE=9033**

specifies the type of device (9032 or 9033).

**UIRATE=DEFAULT**

**UIRATE=INFINITY**

**UIRATE=*value***

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

INFINITY turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

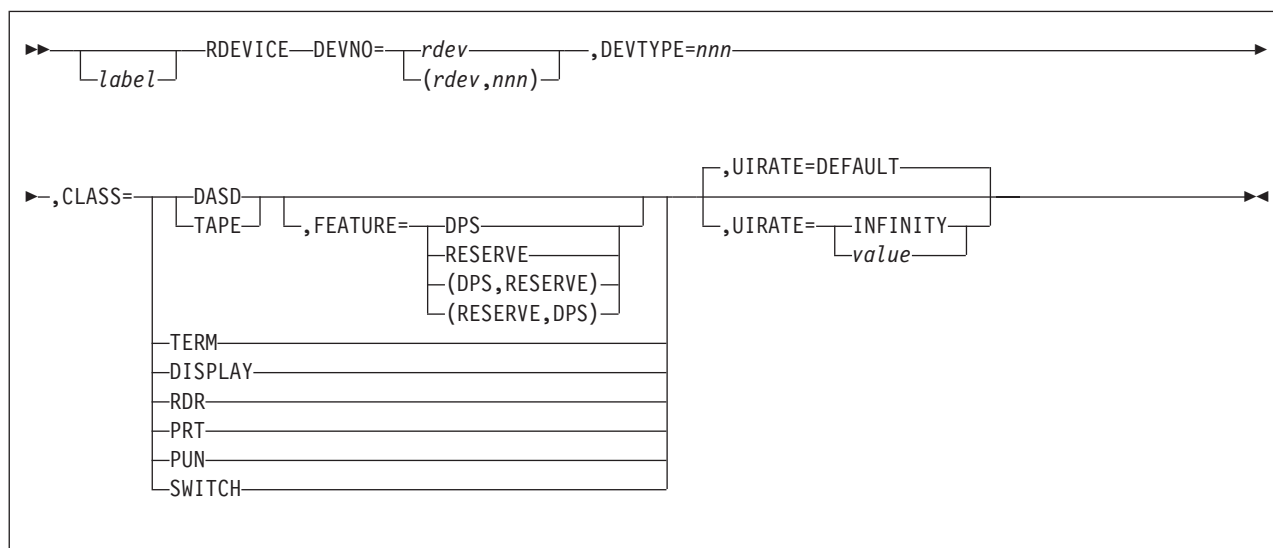
*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

### Examples

To specify a 9032 ES Connection Director at device number 100, code the RDEVICE macro as follows:

```
RDEVICE DEVNO=0100,DEVTYPE=9032
```

## Unsupported Devices



### Purpose

**Note:** When you generate a device as an unsupported device, you must dedicate the device to a virtual machine. Dedicate a device to a virtual machine by coding the DEDICATE control statement in the virtual machine's directory entry or by entering the CP ATTACH command.

### Parameters

**DEVNO**=*rdev*

**DEVNO**=(*rdev, nnn*)

DEVNO=*rdev* specifies a real I/O device number. The variable *rdev* must be a one- to four-digit hexadecimal number in the range 0 through FFFF.

DEVNO=(*rdev, nnn*) specifies real I/O device numbers for a range of devices. The variable *nnn* is a decimal repetition count and creates a group of real device control blocks with consecutive device numbers. The default is 1.

**DEVTYPE**=*nnnn*

where *nnnn* is an unsupported device type.

**CLASS**=DASD

**CLASS**=TAPE

**CLASS**=TERM

**CLASS**=DISPLAY

**CLASS**=RDR

**CLASS**=PRT

**CLASS**=PUN

**CLASS**=SWITCH

is the device class of the unsupported device; valid classes for unsupported devices include:

#### Class Value

##### Class For Unsupported Device

**DASD** Direct access storage devices

**TAPE** Tape devices

**TERM** Terminals

## Unsupported Devices

### DISPLAY

Display mode devices

**RDR** Unit record input devices

**PRT** Unit record printer devices

**PUN** Unit record punch devices

### SWITCH

Dynamic switching devices.

### FEATURE=DPS

### FEATURE=RESERVE

### FEATURE=(DPS,RESERVE)

### FEATURE=(RESERVE,DPS)

specifies the optional features for an unsupported DASD or tape device.

- DPS should be specified whenever a device is defined as an unsupported device and it contains support for 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 DPS is specified, VM's control program places VM's path group identifier (PGID) in the device's control unit for each path to the control unit. This 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 IBM's MVS/XA™ and MVS/ESA™ has an "alternate-PGID" capability to deal with the VM PGID on these channel paths.

When DPS is not specified, VM takes no action with regard to path groups or PGIDs. It is the responsibility of the guest to which the unsupported 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 operationally 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(s).

DPS should not be specified if the guest operating system to which the device will be dedicated or attached does not contain "alternate PGID" support.

The DPS parameter does not apply to devices that are defined to VM as supported devices. In all other cases the DPS parameter, if specified, generates an MNOTE and processing for that particular RDEVICE macro is terminated.

- RESERVE should be specified whenever a device is defined as an unsupported device and the device contains support for the reserve/release function: CCW command codes X'B4', Device Reserve (RES), X'94', Device Release (REL), and X'14', Unconditional Reserve (UR).

When RESERVE is specified for an unsupported device, VM's control program 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 System Clear, System Reset, or IPL commands).

If RESERVE is not specified for an unsupported device 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.

The RESERVE parameter does not apply to devices that are defined to VM as supported devices. In these cases the RESERVE parameter, if specified, generates an MNOTE and processing for that particular RDEVICE macro is terminated.

You can code DPS and RESERVE in any order. A single feature does not need to be enclosed in parentheses.

### **UIRATE=DEFAULT**

### **UIRATE=INFINITY**

### **UIRATE=*value***

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

**INFINITY** turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

## **Examples**

To specify a type 9000 unsupported DASD, code the RDEVICE macroinstruction as follows:

```
RDEVICE DEVNO=0FFF,DEVTYPE=9000,CLASS=DASD
```

To specify an unsupported DASD, that supports dynamic path selection, pick a device type value that does not correspond to any IBM device type; for example, device type 8000 and code the RDEVICE macroinstruction as follows:

```
RDEVICE DEVNO=0FE0,DEVTYPE=8000,CLASS=DASD, *  
        FEATURE=DPS
```

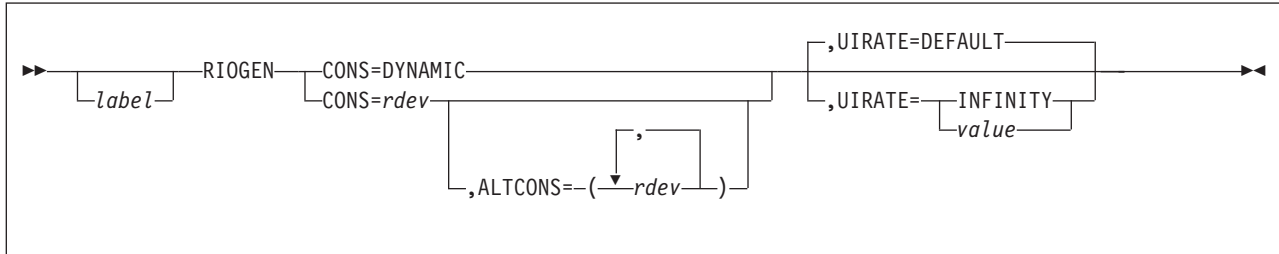
To specify an unsupported DASD, that supports both dynamic path selection and reserve/release, pick a device type value that does not correspond to any IBM device type; for example device type 7000 and code the RDEVICE macroinstruction as follows:

```
RDEVICE DEVNO=0FD0,DEVTYPE=7000,CLASS=DASD, *  
        FEATURE=(DPS,RESERVE)
```

To specify an unsupported tape that supports dynamic path selection, select a device type value that does not correspond to any IBM device type; for example device type 9100 and code the RDEVICE macroinstruction as follows:

```
RDEVICE DEVNO=0FE0,DEVTYPE=9100,CLASS=TAPE,FEATURE=DPS
```

---

**RIOGEN**


## Purpose

Use the RIOGEN macroinstruction to complete the real I/O definition and define the primary and alternate system consoles. It must appear as the last macroinstruction in the HCPRIO file.

## Parameters

### **CONS=DYNAMIC**

tells CP that the console addresses are defined in the system configuration file.

### **CONS=rdev**

specifies the real hexadecimal device number of the CP primary system console. You must specify the real device number in an RDEVICE macroinstruction. This device must be a locally attached 3270-type supported display (see “3270-Family Displays” on page 734 for a list of 3270-type displays supported).

### **ALTCONS=(rdev)**

### **ALTCONS=(rdev,...,rdev)**

specifies the device number or a list of device numbers to be used as alternate system consoles. The maximum number of primary and alternate consoles is 32. If you specify a single device number, you can omit the parentheses.

Although you do not need to specify alternate device numbers in an RDEVICE macroinstruction, you receive an MNOTE if you decide not to do so. An alternate device can be any locally attached 3270-type device. If the primary system console is not operational at system initialization, CP tries to access the first alternate console. If the first alternate console is not operational, CP tries to start the next alternate console, and so on. If CP finds an operational console, it uses it as the primary system operator’s console. If CP cannot find an operational alternate console (or if you specified no alternate console), CP enters a disabled wait state with a wait state code of X'1010' in the PSW.

### **UIRATE=DEFAULT**

### **UIRATE=INFINITY**

### **UIRATE=value**

This optional parameter allows the detection of a hot I/O occurrence to be customized for a device.

**DEFAULT** results in 16 unsolicited interrupts per second before a hot I/O condition is detected. This value is the same rate used for those devices that have no UIRATE parameter specified.

**INFINITY** turns hot I/O recovery off for the device.

**ATTENTION:** It is not recommended that you set the UIRATE value to INFINITY for any great 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.

*value* can be a decimal number from 1 to 62500 which specifies the number of consecutive unsolicited interrupts allowed per second before a hot I/O condition is detected.

## Usage Notes

1. For practical purposes, alternate system consoles should be close to primary system consoles. Because the system operator may have to respond to messages on an alternate console, the most convenient location for an alternate console is near the primary console.
2. 3250 displays are not valid for the RIOGEN macroinstruction.
3. During a software re-IPL caused by either a system incident or the 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 onto a console other than the primary console, CP disconnects the operator virtual machine when the system is reinitialized. This is a security measure to reduce the risk of CP logging the operator virtual machine on to an unattended console.

## Examples

The following example defines a primary system console (001F) with an alternate console (0050).

```
RIOGEN CONS=001F,ALTCONS=0050
```

You can indicate that you are defining console addresses in the system configuration file by coding:

```
RIOGEN CONS=DYNAMIC
```





## Appendix D. Configuration Guide for Printers

Table 36 gives the following information for each printer:

- The printer type and the page in Appendix C, “Defining I/O Devices Using HCPRI0” that describes in detail how to code the RDEVICE macroinstruction for the printer (including examples)
- The DEVTYPE operand you must specify
- The DEVNO operand you must specify
- Optional operands you can specify.

In the tables, the less than (<) and greater than (>) signs indicate that you have a choice of values or operands, one of which you must choose. The “logical or bar” (|) means *or*. For example, <YESINO> means you must choose YES or NO as a value. Never include the <, >, or | in the macroinstruction. When an option appears in **bold, underlined** letters, that option is the default.

Before coding the RDEVICE macroinstruction for a device, make sure you consult any pertinent notes that appear at end of the table.

### Important Note

Many of the printers discussed in this appendix can be sensed by CP or defined in the system configuration file using the RDEVICE statement. For more information about sensing devices, see Chapter 5, “Defining I/O Devices,” on page 45. For more information about the RDEVICE statement, see page 188.

Table 36. Configuration Guide for Printers

Device Type	DEVTYPE	Required Operands	Optional Operands
Advanced Function Printers: see page 708	AFP1	DEVNO=<rdevl (rdev,nnnn)>	
3203 printer: see page 716	3203	DEVNO=<rdevl (rdev,nnnn)>	MODEL=5 CLASS=<cl(cl,...) NONE * <b><u>IA</u></b> > CHARS=<ucsname  <b><u>PCAN</u></b> > DEST=<dest1,dest2,dest3,dest4  <b><u>OFF</u></b>  *> AFP=< <b><u>YESINO</u></b> > FCB=<fcbname  <b><u>FCB1</u></b> > FEATURE=UNVCHSET FORM=<operforml * <b><u>STANDARD</u></b> > SEP=< <b><u>YESINO</u></b> > IMAGE=<imagelib  <b><u>IMAG3203</u></b> > FOLD=< <b><u>YESINO</u></b> >
Devices that emulate a 3211 printer: see page 716	3211	DEVNO=<rdevl (rdev,nnnn)>	CLASS=<cl(cl,...) NONE * <b><u>IA</u></b> > CHARS=<ucsname  <b><u>A11</u></b> > DEST=<dest1,dest2,dest3,dest4  <b><u>OFF</u></b>  *> AFP=< <b><u>YESINO</u></b> > FCB=<fcbname  <b><u>FCB1</u></b> > FEATURE=UNVCHSET FORM=<operforml * <b><u>STANDARD</u></b> > SEP=< <b><u>YESINO</u></b> > IMAGE=<imagelib  <b><u>IMAG3211</u></b> > FOLD=< <b><u>YESINO</u></b> > INDEX=<nnl  <b><u>1</u></b> >

## Configuration Guide for Printers

Table 36. Configuration Guide for Printers (continued)

Device Type	DEVTYPE	Required Operands	Optional Operands
3216 display printer: see page 708	AFP1	DEVNO=<rdevl (rdev,nnnn)>	
3262 display printer: see page 713	3287	DEVNO=<rdevl (rdev,nnn)>	
3262 impact printer: see page 716	3262 <sup>Note 1</sup>	DEVNO=<rdevl (rdev,nnnn)>	MODEL=5 CLASS=<cl(cl,...) NONE *IA> CHARS=<ucsname P48> DEST=<dest1,dest2,dest3,dest4 OFF *> AFP=<YESINO> FCB=<fcbname FCB1> FORM=<operform * STANDARD> SEP=<YESINO> IMAGE=<imagelib IMAG3262> FOLD=<YESINO>
3286 display printer: see page 713	3286	DEVNO=<rdevl (rdev,nnn)>	
3268 display printer: see page 713	3287	DEVNO=<rdevl (rdev,nnn)>	
3287 display printer: see page 713	3287	DEVNO=<rdevl (rdev,nnn)>	
3288 display line printer: see page 713	3288	DEVNO=<rdevl (rdev,nnn)>	
3289 display line printer: see page 713	3289	DEVNO=<rdevl (rdev,nnn)>	
3800 printer: see pages 721 and 708	3800 <sup>Note 2</sup>	DEVNO=<rdevl (rdev,nnnn)>	MODEL=<1 3 8> CLASS=<cl(cl,...) NONE *IA> CHARS=ffff DEST=<dest1,dest2,dest3,dest4 OFF *> AFP=<YESINO> DPMSIZE=<n 1> FCB=< pi 6> FEATURE=4WCGMS FLASH=flashlib FORM=<operform * STANDARD> IMAGE=<imagelib IMAG3800> SEP=<YESINO>
3812 printer: see page 708	AFP1	DEVNO=<rdevl (rdev,nnnn)>	
3816 printer: see page 708	AFP1	DEVNO=<rdevl (rdev,nnnn)>	
3820 printer: see page 708	AFP1	DEVNO=<rdevl (rdev,nnnn)>	
3827 printer: see page 708	AFP1	DEVNO=<rdevl (rdev,nnnn)>	
3835 printer: see page 708	AFP1	DEVNO=<rdevl (rdev,nnnn)>	
4214 display printer: see page 713	3287	DEVNO=<rdevl (rdev,nnn)>	
4224 display printer: see page 713	3287	DEVNO=<rdevl (rdev,nnn)>	

Table 36. Configuration Guide for Printers (continued)

Device Type	DEVTYPE	Required Operands	Optional Operands
4234 display printer: see page 713	3287	DEVNO=<rdevl (rdev,nnn)>	
4245 display printer: see page 713	3287	DEVNO=<rdevl (rdev,nnn)>	
4245 impact printer: see page 716	4245 <sup>Note 3</sup>	DEVNO=<rdevl (rdev,nnnn)>	MODEL=<1 12 20> CLASS=<cl(cl,...) NONE * A> DEST=<dest1,dest2,dest3,dest4 OFF *> AFP=<YES NO> FCB=<fcname FCB1> FORM=<operforml * STANDARD> SEP=<YES NO> IMAGE=<imagelib IMAG4245> FOLD=<YES NO>
4248 printer: see page 716	4248 <sup>Note 4</sup>	DEVNO=<rdevl (rdev,nnnn)>	MODEL=1 CLASS=<cl(cl,...) NONE * A> DEST=<dest1,dest2,dest3,dest4 OFF *> AFP=<YES NO> FCB=<fcname FCB1> FORM=<operforml * STANDARD> SEP=<YES NO> IMAGE=<imagelib IMAG4248> FOLD=<YES NO>
5210 display printer: see page 713	3287	DEVNO=<rdevl (rdev,nnn)>	
6262 display printer: see page 713	3287	DEVNO=<rdevl (rdev,nnn)>	
6262 impact printer: see page 716	4248	DEVNO=<rdevl (rdev,nnnn)>	MODEL=1 CLASS=<cl(cl,...) NONE * A> DEST=<dest1,dest2,dest3,dest4 OFF *> AFP=<YES NO> FCB=fcname FORM=<operforml * STANDARD> SEP=<YES NO> IMAGE=imagelib FOLD=<YES NO>

**Notes:**

1. If a 3262 is in 4248-compatibility mode, specify DEVTYPE=4248.
2. If a 3800-3 or 3800-6 is run in 3800-1 compatibility mode, specify MODEL=1. If a 3800-3, 3800-6, or 3800-8 is used as an advanced function printer, specify DEVTYPE=AFP1.
3. If the 4245-1 is in 3262-compatibility mode, specify DEVTYPE=3262.
4. If a 4248-1 or 4248-2 is in 3211-compatibility mode, specify DEVTYPE=3211. When specifying a 4248-2 as a 4248, specify MODEL=1.



---

## Appendix E. 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
CLASAIFF	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



TYPAIF0 EQU X'00' AIF - AIF Adapter Type 0

---

## Device Features by Class and Type

DEVFEAT BIT DEFINITIONS		FEATURES OF DEVICES BY CLASS/TYPE
FTR0PRDR	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

## Device Class and Type Codes

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 F. Stand-Alone Dump Formats

The following sections describe the tape, DASD, and printer formats of stand-alone dumps.

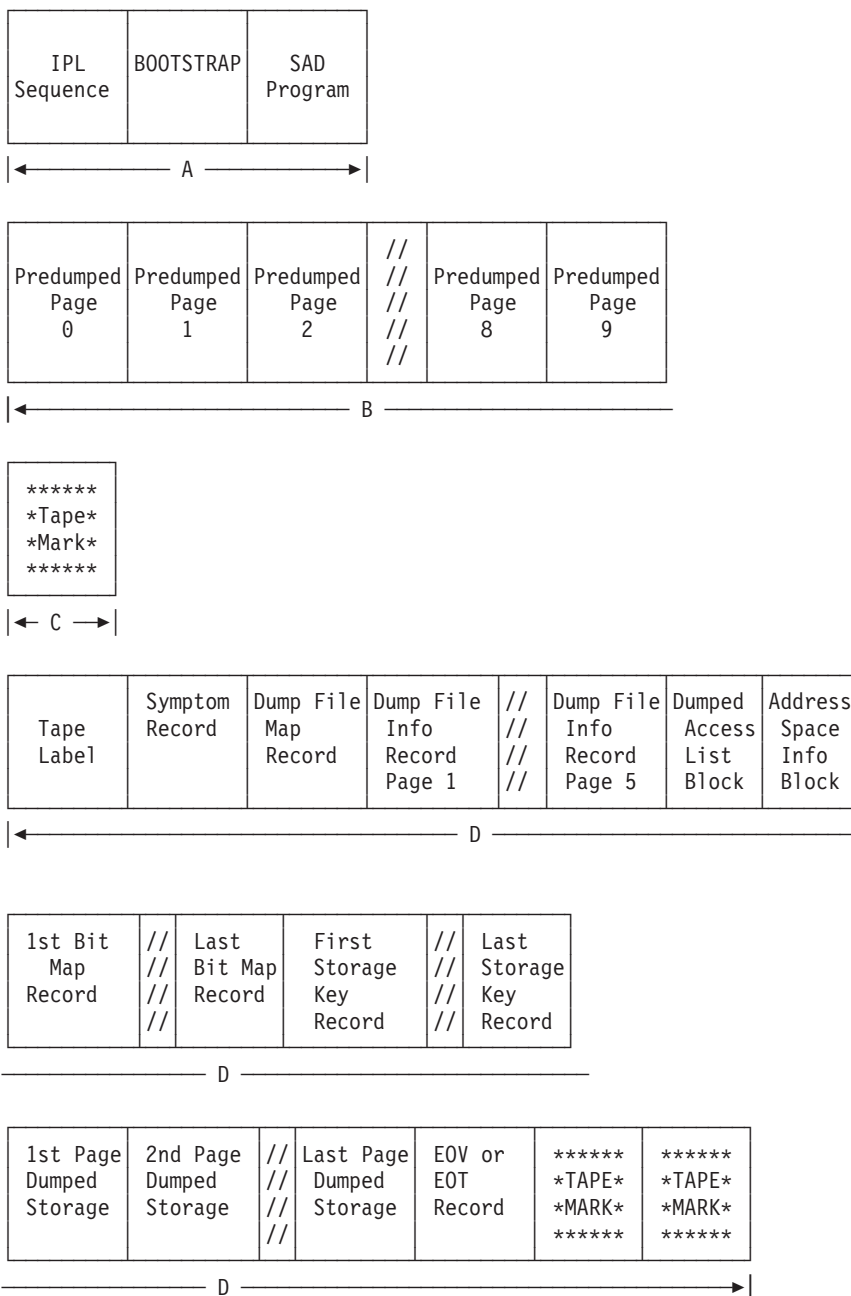
---

### Tape Format

A tape used with the stand-alone dump facility has the format shown in Figure 42 on page 778.

- If the IPL tape and the dump device are not the same, the IPL tape includes sections A, B, and C.
- If the dump device is a tape, but not the same tape as the IPL tape, the dump output tape includes sections C and D.
- If the IPL device and the dump device are the same, the tape includes sections A, B, C, and D.

## Stand-Alone Dump Formats

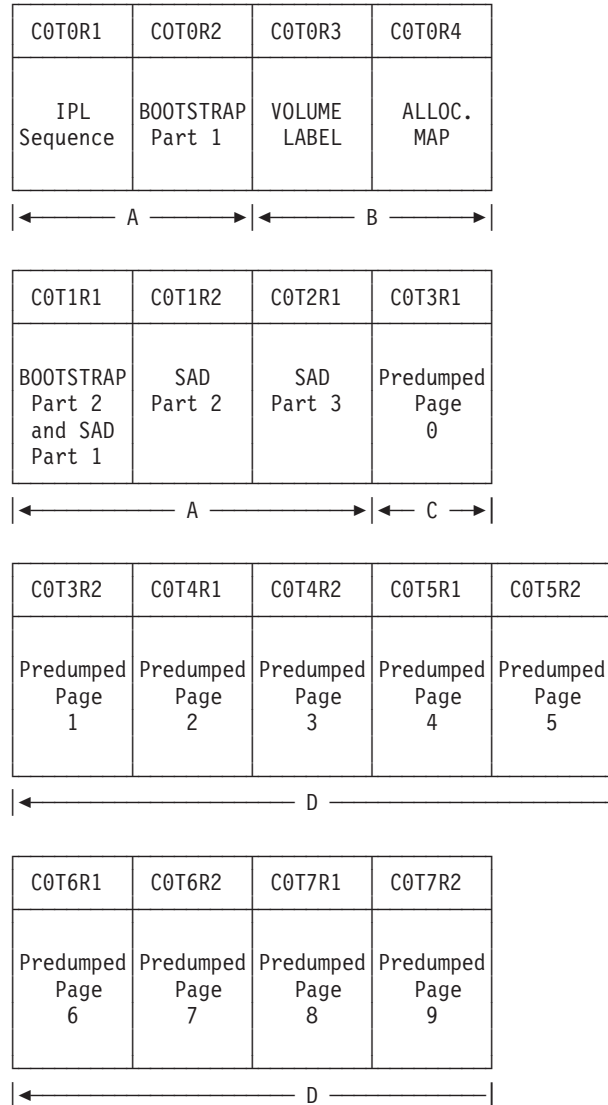


- A**      Written by the stand-alone dump utility on the IPL tape at generation time.
- B**      Written by BOOTSTRAP on the IPL tape.
- C**      Written by BOOTSTRAP if the IPL tape is the same as the dump tape.  
          Written by the stand-alone dump program if the IPL tape is not the same as  
          the dump tape.
- D**      Written by the stand-alone dump program on the dump tape.

Figure 42. Stand-Alone Dump Facility Tape Format

## DASD Format

When you use a DASD device to IPL the stand-alone dump program, the system uses cylinder 0 to hold the program. Cylinder 0 must be CP-formatted and allocated as permanent space. The stand-alone dump facility has the format shown in Figure 43.



- A** Written by the stand-alone dump utility on the IPL DASD at generation time.
- B** Written by the FORMAT/ALLOCATE program.
- C** Written by the BOOTSTRAP Part 1 on the IPL DASD.
- D** Written by the BOOTSTRAP Part 2 on the IPL DASD.

CnTnRn identifies cylinder, track, and record numbers.

Figure 43. Stand-Alone Dump Facility DASD Format

### Printer Format

Dumps to printer devices are printed as follows:

- CP formats the following data fields for each processor, beginning with the processor where the stand-alone dump program was IPLed:
  - The CPU address
  - General purpose registers
  - Control registers
  - Access registers (only if multiple address spaces [MAS] is installed)
  - Floating point registers 0, 2, 4, and 6
  - The TOD clock (only for the processor where the stand-alone dump program was IPLed)
  - CPU timer values
  - The TOD clock comparator
  - The prefix register (when valid)
  - The stored-status PSW (only for non-IPL processors and only if there is a valid prefix register)
  - Old/new PSWs for the IPL processor when there is a valid prefix register:
    - External interrupt
    - SVC
    - Program check
    - Machine check
    - I/O interrupt.
  - The print storage start at absolute page 0.
- Lines of duplicate data evoke a suppression message after the first line of the data is printed.
- A full page (4096 bytes) of all zeros has one line of zeros printed with the key, followed by a line-suppressed message.
- The dump page is interpreted on the right-hand side of the printout.

---

## Notices

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

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

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, New York 10594-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
Mail Station P300  
2455 South Road  
Poughkeepsie, New York 12601-5400  
U.S.A.  
Attention: Information Request

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

---

## Programming Interface Information

This book documents intended Programming Interfaces that allow the customer to write programs to obtain services of z/VM.

---

## Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, or other countries, or both:

Advanced Function Printing  
AFP  
DFSMS/VM  
DirMaint  
ECKD  
Enterprise Storage Server  
ESCON  
eServer  
FICON  
HiperSockets  
IBM  
IBMLink  
Language Environment  
MVS  
MVS/ESA  
MVS/XA  
OpenExtensions  
OS/390  
Performance Toolkit for VM  
Print Services Facility  
RACF  
RAMAC  
RETAIN  
S/360  
S/370  
SAA  
SQL/DS  
System/370  
System/390  
Systems Application Architecture  
TotalStorage  
VM/ESA  
VTAM  
z/Architecture  
z/OS  
z/VM  
zSeries  
3090  
3890

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

Other company, product, and service names may be trademarks or service marks of others.





---

## Glossary

For a list of z/VM terms and their definitions, see the *z/VM: Glossary* book.

The glossary is also available through the online HELP Facility. For example, to display the definition of “cms”, enter:

```
help glossary cms
```

You will enter the glossary HELP file and the definition of “cms” will be displayed as the current line. While you are in the glossary HELP file, you can also search for other terms.

If you are unfamiliar with the HELP Facility, you can enter:

```
help
```

to display the main HELP menu, or enter:

```
help cms help
```

for information about the HELP command.

For more information about the HELP Facility, see the *z/VM: CMS User's Guide*.



---

## Bibliography

This bibliography lists the books in the z/VM product library. For abstracts of these books and information about current editions and available media, see *z/VM: General Information*.

---

### Where to Get z/VM Books

z/VM books are available from the following sources:

- IBM Publications Center at [www.ibm.com/shop/publications/order/](http://www.ibm.com/shop/publications/order/)
- z/VM Internet Library at [www.ibm.com/eserver/zseries/zvm/library/](http://www.ibm.com/eserver/zseries/zvm/library/)
- IBM eServer zSeries Online Library: z/VM Collection CD-ROM, SK2T-2067

---

### z/VM Base Library

The following books describe the facilities included in the z/VM base product.

#### System Overview

- z/VM: General Information*, GC24-6095
- z/VM: Glossary*, GC24-6097
- z/VM: License Information*, GC24-6102
- z/VM: Migration Guide*, GC24-6103

#### Installation and Service

- z/VM: Guide for Automated Installation and Service*, GC24-6099
- z/VM: Service Guide*, GC24-6117
- z/VM: VMSES/E Introduction and Reference*, GC24-6130

#### Planning and Administration

- z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6074
- z/VM: CMS Planning and Administration*, SC24-6078
- z/VM: Connectivity*, SC24-6080
- z/VM: CP Planning and Administration*, SC24-6083
- z/VM: Getting Started with Linux on zSeries*, SC24-6096
- z/VM: Group Control System*, SC24-6098
- z/VM: I/O Configuration*, SC24-6100
- z/VM: Performance*, SC24-6109

*z/VM: Running Guest Operating Systems*, SC24-6115

*z/VM: Saved Segments Planning and Administration*, SC24-6116

*z/VM: Secure Configuration Guide*, SC24-6138

*z/VM: TCP/IP Planning and Customization*, SC24-6125

*eServer zSeries 900: Planning for the Open Systems Adapter-2 Feature*, GA22-7477

*eServer zSeries: Open Systems Adapter-Express Customer's Guide and Reference*, SA22-7935

*eServer zSeries: Open Systems Adapter-Express Integrated Console Controller User's Guide*, SA22-7990

*z/OS and z/VM: Hardware Configuration Manager User's Guide*, SC33-7989

#### Customization

*z/VM: CP Exit Customization*, SC24-6082

#### Operation

- z/VM: System Operation*, SC24-6121
- z/VM: Virtual Machine Operation*, SC24-6128

#### Application Programming

- z/VM: CMS Application Development Guide*, SC24-6069
- z/VM: CMS Application Development Guide for Assembler*, SC24-6070
- z/VM: CMS Application Multitasking*, SC24-6071
- z/VM: CMS Callable Services Reference*, SC24-6072
- z/VM: CMS Macros and Functions Reference*, SC24-6075
- z/VM: CP Programming Services*, SC24-6084
- z/VM: CPI Communications User's Guide*, SC24-6085
- z/VM: Enterprise Systems Architecture/Extended Configuration Principles of Operation*, SC24-6094
- z/VM: Language Environment User's Guide*, SC24-6101
- z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6104

*z/VM: OpenExtensions Callable Services Reference*, SC24-6105

*z/VM: OpenExtensions Commands Reference*, SC24-6106

*z/VM: OpenExtensions POSIX Conformance Document*, GC24-6107

*z/VM: OpenExtensions User's Guide*, SC24-6108

*z/VM: Program Management Binder for CMS*, SC24-6110

*z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6112

*z/VM: REXX/VM Reference*, SC24-6113

*z/VM: REXX/VM User's Guide*, SC24-6114

*z/VM: Systems Management Application Programming*, SC24-6122

*z/VM: TCP/IP Programmer's Reference*, SC24-6126

*Common Programming Interface Communications Reference*, SC26-4399

*Common Programming Interface Resource Recovery Reference*, SC31-6821

*OS/390: DFSMS Program Management*, SC27-0806

*z/OS: Language Environment Concepts Guide*, SA22-7567

*z/OS: Language Environment Debugging Guide*, GA22-7560

*z/OS: Language Environment Programming Guide*, SA22-7561

*z/OS: Language Environment Programming Reference*, SA22-7562

*z/OS: Language Environment Run-Time Messages*, SA22-7566

*z/OS: Language Environment Writing ILC Applications*, SA22-7563

## End Use

*z/VM: CMS Commands and Utilities Reference*, SC24-6073

*z/VM: CMS Pipelines Reference*, SC24-6076

*z/VM: CMS Pipelines User's Guide*, SC24-6077

*z/VM: CMS Primer*, SC24-6137

*z/VM: CMS User's Guide*, SC24-6079

*z/VM: CP Commands and Utilities Reference*, SC24-6081

*z/VM: Quick Reference*, SC24-6111

*z/VM: TCP/IP User's Guide*, SC24-6127

*z/VM: XEDIT Commands and Macros Reference*, SC24-6131

*z/VM: XEDIT User's Guide*, SC24-6132

*CMS/TSO Pipelines Author's Edition*, SL26-0018

## Diagnosis

*z/VM: Diagnosis Guide*, GC24-6092

*z/VM: Dump Viewing Facility*, GC24-6093

*z/VM: System Messages and Codes - AVS, Dump Viewing Facility, GCS, TSAF, and VMSES/E*, GC24-6120

*z/VM: System Messages and Codes - CMS and REXX/VM*, GC24-6118

*z/VM: System Messages and Codes - CP*, GC24-6119

*z/VM: TCP/IP Diagnosis Guide*, GC24-6123

*z/VM: TCP/IP Messages and Codes*, GC24-6124

*z/VM: VM Dump Tool*, GC24-6129

*z/OS and z/VM: Hardware Configuration Definition Messages*, SC33-7986

---

## Books for z/VM Optional Features

The following books describe the optional features of z/VM.

## Data Facility Storage Management Subsystem for VM

*z/VM: DFSMS/VM Customization*, SC24-6086

*z/VM: DFSMS/VM Diagnosis Guide*, GC24-6087

*z/VM: DFSMS/VM Messages and Codes*, GC24-6088

*z/VM: DFSMS/VM Planning Guide*, SC24-6089

*z/VM: DFSMS/VM Removable Media Services*, SC24-6090

*z/VM: DFSMS/VM Storage Administration*, SC24-6091

## Directory Maintenance Facility

*z/VM: Directory Maintenance Facility Commands Reference*, SC24-6133

*z/VM: Directory Maintenance Facility Messages*, GC24-6134

*z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6135

## Performance Toolkit for VM™

*z/VM: Performance Toolkit, SC24-6136*

### Resource Access Control Facility

*External Security Interface (RACROUTE)*

*Macro Reference for MVS and VM,*

*GC28-1366*

*Resource Access Control Facility: Auditor's Guide, SC28-1342*

*Resource Access Control Facility: Command Language Reference, SC28-0733*

*Resource Access Control Facility: Diagnosis Guide, GY28-1016*

*Resource Access Control Facility: General Information, GC28-0722*

*Resource Access Control Facility: General User's Guide, SC28-1341*

*Resource Access Control Facility: Macros and Interfaces, SC28-1345*

*Resource Access Control Facility: Messages and Codes, SC38-1014*

*Resource Access Control Facility: Migration and Planning, GC23-3054*

*Resource Access Control Facility: Security Administrator's Guide, SC28-1340*

*Resource Access Control Facility: System Programmer's Guide, SC28-1343*



---

# Index

## Special characters

- \*ACCOUNT system service 233, 494, 661
- \*BLOCKIO system service 494
- \*CONFIG system service 494
- \*CRM system service 494
- \*IDENT system service 496
- \*LOGREC system service 314, 494, 669
- \*MONITOR system service 494
- \*MSG system service 494
- \*MSGALL system service 494
- \*RPI system service 494
- \*SIGNAL system service 494
- \*SPL system service 494
- \*SYMPTOM system service 318, 494
  - specifying virtual machines for 699
- \*VSWITCH system service 494
- + operand and – operand
  - TIMEZONE\_DEFINITION statement 239

## Numerics

- 2250 display
  - restriction against using 124
- 2701 data adapter
  - coding an RDEVICE macro 752
- 2741 terminal
  - coding an RDEVICE macro 754
- 3088 MCCU
  - coding an RDEVICE macro 760
  - defining in system configuration file 50
  - requirement for CSE 335
  - virtual, defining for virtual machine 557
  - virtual, defining for virtual network adapters 529
- 3101 display
  - coding an RDEVICE macro 734
- 3151 display terminal
  - coding an RDEVICE macro 754
- 3161 display terminal
  - coding an RDEVICE macro 754
- 3162 display terminal
  - coding an RDEVICE macro 754
- 3163 display terminal
  - coding an RDEVICE macro 754
- 3167 display terminal
  - coding an RDEVICE macro 754
- 3172 controller
  - defining 205
  - defining in system configuration file 50
- 3174 controller
  - coding an RDEVICE macro 757
- 3178 display
  - coding an RDEVICE macro 734
  - defining 197
- 3179 display
  - coding an RDEVICE macro 734
  - defining 197
- 3180 display
  - coding an RDEVICE macro 734
  - defining 197
- 3190 display
  - defining 197
- 3191 display
  - coding an RDEVICE macro 734
- 3192 display
  - coding an RDEVICE macro 734
- 3193 display
  - coding an RDEVICE macro 734
- 3194 display
  - coding an RDEVICE macro 734
- 3203 printer
  - coding an RDEVICE macro 716
  - defining 201
  - defining in system configuration file 47
  - FCB image
    - adding 418
    - example for coding new 419
    - provided 413
  - UCS buffer image
    - adding 414
    - example of new 416
    - provided 412
- 3211 printer
  - coding an RDEVICE macro 716
  - defining 201
  - defining in system configuration file 47
  - FCB image
    - adding 418
    - example for coding new 419
    - provided 413
  - UCS buffer image
    - adding 414
    - example of new 417
    - provided 412
- 3250 display
  - coding an RDEVICE macro 733, 734
  - restriction against using 124
- 3262 printer
  - coding an RDEVICE macro 713, 716
  - defining 201
  - FCB image
    - adding 414, 418
    - provided 413
  - UCS buffer image
    - example of new 417
    - provided 413
- 3268 display printer
  - coding an RDEVICE macro 713
  - defining in system configuration file 47
- 3270 console mode
  - defining 464
- 3270 display
  - defining 197, 198
- 3270\_DISPLAY operand
  - RDEVICE statement 210

3270\_PRINTER operand  
  RDEVICE statement 210

3270-PC  
  coding an RDEVICE macro 734

3274 controller  
  coding an RDEVICE macro 757

3277 display  
  coding an RDEVICE macro 734  
  defining 197, 198  
  exit for installation-supplied code 312

3277 operand  
  RDEVICE statement 198

3277\_APL operand  
  TRANSLATE\_TABLE statement 244

3277\_TEXT operand  
  TRANSLATE\_TABLE statement 245

3278 display  
  coding an RDEVICE macro 734  
  defining 197

3278\_APL operand  
  TRANSLATE\_TABLE statement 245

3278\_TEXT operand  
  TRANSLATE\_TABLE statement 245

3279 display  
  coding an RDEVICE macro 734  
  defining 197

3286 printer  
  coding an RDEVICE macro 713

3287 printer  
  coding an RDEVICE macro 713  
  defining in system configuration file 47

3288 printer  
  coding an RDEVICE macro 713

3289 printer  
  coding an RDEVICE macro 713  
  defining in system configuration file 47

3290 display  
  coding an RDEVICE macro 734  
  defining 197

3350 DASD  
  coding an RDEVICE macro 746  
  in 3330 compatibility mode 746

3370 DASD  
  coding an RDEVICE macro 749

3375 DASD  
  coding an RDEVICE macro 746

3380 DASD  
  capacity information 605  
  coding an RDEVICE macro 746  
  defining 195  
  defining in system configuration file 46  
  using as a cached storage device 628

3390 DASD  
  capacity information 605  
  coding an RDEVICE macro 746  
  defining 195  
  defining as a full-pack minidisk 625  
  defining in system configuration file 46  
  migration considerations 617  
  switching operating modes 615  
  using as a cached storage device 628

3420 tape unit  
  defining 207

3422 tape unit  
  coding an RDEVICE macro 743  
  defining 207

3423 optical media attachment  
  defining in system configuration file 50

3430 tape unit  
  coding an RDEVICE macro 743

3480 tape unit  
  coding an RDEVICE macro 743  
  defining in system configuration file 47

3490 tape drive  
  coding an RDEVICE macro 743  
  defining in system configuration file 47

3495 tape library dataserver  
  controlled by a virtual machine 566  
  defining in system configuration file 47

3505 card reader  
  coding an RDEVICE macro 727  
  defining 192  
  defining in system configuration file 48

3525 card punch  
  coding an RDEVICE macro 729  
  defining 190  
  defining in system configuration file 48

3590 tape unit  
  coding an RDEVICE macro 743  
  defining in system configuration file 47

3705 communication controller  
  coding an RDEVICE macro 754, 757  
  defining 193  
  defining in system configuration file 49

3705 operand  
  RDEVICE statement 194

3720 communication controller  
  coding an RDEVICE macro 757  
  defining 193  
  defining in system configuration file 49

3725 communication controller  
  coding an RDEVICE macro 754, 757  
  defining 193  
  defining in system configuration file 49

3737 remote channel-to-channel unit  
  coding an RDEVICE macro 760  
  defining 205  
  defining in system configuration file 50

3745 communication controller  
  coding an RDEVICE macro 754, 757  
  defining 193  
  defining in system configuration file 49

3800 operand  
  RDEVICE statement 212

3800 printer  
  coding an RDEVICE macro 721  
  creating a text deck for 411  
  defining 189, 212  
  defining in system configuration file 47  
  image library  
  creating 424  
  description 411



- 3812 printer
    - coding an RDEVICE macro 713
    - defining in system configuration file 48
  - 3816 printer
    - coding an RDEVICE macro 713
    - defining in system configuration file 48
  - 3820 printer
    - coding an RDEVICE macro 710
    - defining 189
    - defining in system configuration file 48
  - 3825 printer
    - coding an RDEVICE macro 708
    - defining 189
    - defining in system configuration file 48
  - 3827 printer
    - coding an RDEVICE macro 708
    - defining 189
    - defining in system configuration file 48
  - 3828 printer
    - coding an RDEVICE macro 708
    - defining 189
    - defining in system configuration file 48
  - 3835 printer
    - coding an RDEVICE macro 708
    - defining 189
    - defining in system configuration file 48
  - 3850 DASD
    - coding an RDEVICE macro 746
  - 3890 document processor
    - coding an RDEVICE macro 731
    - defining in system configuration file 50
  - 3900 printer
    - coding an RDEVICE macro 708
    - defining 189, 212
    - defining in system configuration file 48
  - 4214 display printer
    - coding an RDEVICE macro 713
  - 4224 display printer
    - coding an RDEVICE macro 713
  - 4234 display printer
    - coding an RDEVICE macro 713
  - 4245 printer
    - coding an RDEVICE macro 713, 716
    - defining 201
    - defining in system configuration file 48
    - FCB image
      - adding 418
      - provided 413
  - 4248 printer
    - coding an RDEVICE macro 716
    - defining 201
    - defining in system configuration file 48
    - FCB image
      - adding external 420
      - adding new 418
      - provided 413
    - FCB macro, example for coding 421
  - 4753 network security processor
    - defining in system configuration file 50
  - 5080 graphics display
    - coding an RDEVICE macro 740
  - 5210 display printer
    - coding an RDEVICE macro 713
  - 5550 display
    - coding an RDEVICE macro 734
  - 6090 graphics display
    - coding an RDEVICE macro 740
  - 6262 printer
    - coding an RDEVICE macro 713, 716
    - defining 201
    - defining in system configuration file 48
  - 7171 device attachment control unit
    - coding an RDEVICE macro 734
    - defining in system configuration file 50
  - 802.3 local area network 738
  - 8232 LAN channel station
    - defining 205
    - defining in system configuration file 50
  - 9032 ESCON Director Model 2
    - defining 762
    - sensed by CP 50
  - 9033 ESCON Director Model 1
    - defining 762
    - sensed by CP 50
  - 9034 ESCON Converter Model 1 50
  - 9035 ESCON Converter Model 2 50
  - 9221 integrated ethernet
    - defining in system configuration file 49
  - 9221 integrated token ring
    - defining in system configuration file 49
  - 9336 DASD
    - capacity information 605
    - defining 195
  - 9345 DASD
    - coding an RDEVICE macro 746
- A**
- ABBREVLLENGTH operand
    - DEFINE ALIAS statement 82
    - DEFINE CMD statement 85
    - DEFINE COMMAND statement 85
  - abnormal end (abend)
    - list of consoles to receive messages about 124
  - ACCEPTED operand
    - DEVICES statement 105
  - access
    - disks
      - by CP 70
    - list control parameters 580
    - modes
      - defined 501
      - exclusive 327, 387
      - exclusive, authorized to use 535
      - minidisk definitions 515
      - stable 327, 387
      - stable, authorized to use 536
    - read/write 349
    - to spool files 370
  - account number, defining for a virtual machine 456
  - ACCOUNT operand
    - JOURNALING statement 162

ACCOUNT user directory control statement 456  
ACCOUNT1 operand  
  SYSTEM\_USERIDS statement 233  
ACCOUNT2 operand  
  SYSTEM\_USERIDS statement 233  
accounting  
  adding your own source code 310  
  disassociating a user ID from retrieval of 298  
  exit, CP 310  
  record  
    adding your own 310  
    dedicated device 300  
    format of 299  
    Inter-System Facility for Communications 304  
    journaling 301  
    SET PRIVCLASS 306, 307, 309  
    SNA/CCS 304  
    temporary disk space 301  
    type 04 301  
    type 05 302  
    type 06 302  
    type 07 304  
    type 08 302  
    type 09 304  
    type 0A 306  
    type 0I 303  
    type 1 299  
    type 2 300  
    type 3 301  
    type B 307  
    type C 307  
    type C0 310  
    type D 309  
    user-initiated 310  
    virtual disk space 307  
    virtual machine resource usage 299  
  records  
    specifying user ID to accumulate 232  
  setting up virtual machines for 295  
  system service (\*ACCOUNT) 233  
  virtual machine, defining a user ID for 661  
  virtual machine, logon identification 661  
ACCT option 394  
accumulate  
  accounting records  
    specifying user ID to 232  
  EREP records  
    disassociating a user ID from retrieval of 317  
    specifying user ID to 232  
  symptom records  
    specifying user ID to 232  
  system dump records  
    specifying user ID to 232  
ACIGROUP user directory control statement 458  
ACNT command 299  
activate  
  an override file 438  
  cross system spool 355  
adapter  
  channel-to-channel  
    defining 205  
  adapter (*continued*)  
    data, coding an RDEVICE macro 752  
    line  
      defining 193  
    line, coding an RDEVICE macro 754  
    Local Area Network (LAN)  
      defining 205  
    teleprocessing integrated, coding an RDEVICE  
      macro 738  
add  
  files to an image library 424  
  forms control buffer 418, 420  
  print buffer image 414  
  real devices  
    to system's definition 188  
  your own accounting records and source code 310  
address  
  console  
    alternate 766  
    defining list for operator 180  
    operator 766  
    to receive emergency messages 124  
  spaces  
    defining maximum nonprimary 583  
    defining total size of nonprimary 583  
address space control parameters 580  
ADDRSPACE macro  
  using to create or delete address spaces 583  
administration  
  considerations summary 3  
  CSE 358  
  performance tasks 12  
  real and virtual storage 7  
  system tasks 4  
  task overview 5  
  user tasks 5  
advanced function printer (AFP)  
  coding an RDEVICE macro 708  
  defining 189  
AFP operand  
  RDEVICE statement  
    3800 printers 213  
    advanced function printers 189  
    impact printers 202  
AFTER\_LOGON operand  
  DEFINE CMD statement 85  
  DEFINE COMMAND statement 85  
alias  
  creating  
    new for existing CP command 81  
ALIAS operand  
  XSPOOL\_SYSTEM statement 271  
ALL operand  
  DRAIN statement 118  
  HOT\_IO\_RATE statement 152  
  START statement 220  
allocate  
  device blocks 689  
  space  
    XSPOOL trace tables 273

- allocate (*continued*)
  - storage
    - with STORAGE statement 222
    - with SYSSTORE macro 697
- ALSERV macro
  - adding and deleting entries with 581
- ALTCONS operand
  - RIOGEN macro 766
- alternate
  - parm disk 51
  - system console
    - defining on OPERATOR\_CONSOLES statement 180
    - defining on RIOGEN macro 766
- ALTERNATE\_OPERATORS statement 58
- ANYTIME operand
  - DEFINE CMD statement 85
  - DEFINE COMMAND statement 85
- AOFPARM parameter list 314
- AONPARM parameter list 312
- APAR (Authorized Program Analysis Report)
  - process 400
- APPC/VM path 459
- APPCPASS user directory control statement 459
- architecture, System/390 397
- area, CSE 359
- ART (Access Register Translation)
  - maintains storage protection 391
  - translates addresses 390
- ASCII\_1977 operand
  - TRANSLATE\_TABLE statement 244
- ASCII\_1980 operand
  - TRANSLATE\_TABLE statement 244
- ASCII\_LINE\_DELETE operand
  - CHARACTER\_DEFAULTS statement 67
- ASCII-APL operand
  - TRANSLATE\_TABLE statement 244
- assign
  - commands to types of users 435
  - exit point with entry points 60, 64
  - exit point with external symbols 60, 64
  - new classes 438
- associate
  - classes with users and commands 436
  - exit point with entry points 60, 64
  - exit point with external symbols 60, 64
- ASSOCIATE EXIT statement 51, 60
- ASSOCIATE MESSAGES statement 51, 64
- ASSOCIATE MSGS statement 51, 64
- attach
  - expanded storage to a virtual machine 592
  - retained Expanded Storage 596
- ATTACH command 478, 623, 665
- ATTACH XSTORE command 592, 593
- AUDIT operand
  - DEFINE CMD statement 85
  - DEFINE COMMAND statement 85
  - DEFINE DIAGNOSE statement 90
- auditing and protecting
  - using an external security manager 384

- authorization bypass
  - directory password 385
- AUTO\_WARM\_IPL operand
  - disabling on FEATURES statement 140
  - enabling on FEATURES statement 142
- AUTOLOG command 296, 299
  - journaling 385
  - prompting for password on 144
- AUTOLOG operand
  - SYSTEM\_USERIDS statement 233
- AUTOLOG user directory control statement 461, 578
- AUTOLOG1 service virtual machine 6
- automatic
  - deactivation of restricted passwords 386
- auxiliary storage 7
- avoiding DASD bottlenecks during I/O 337

## B

- backup and recovery, DASD for 337
- BEFORE\_LOGON operand
  - DEFINE CMD statement 85
  - DEFINE COMMAND statement 85
- block, command table entry
  - defining new CP 84
- BOTTOM operand
  - PRINTER\_TITLE statement 182
- BSC\_ADAPTER operand
  - RDEVICE statement 193
- buffer, retrieve
  - specifying
    - default number of 136
    - maximum number of 136
- bypassing directory password authorization 385

## C

- cabling, hardware 341
- cache
  - DASD
    - defining a minidisk for 629
    - defining as a dedicated device 630
    - description 628
  - minidisk
    - devices available for 600
    - performance considerations 602
    - planning for 600
    - special considerations 601
    - turning off 600
    - using 600
    - using Expanded Storage as a 597
- capacity, DASD
  - storage 604
- card punch
  - 3525
    - defining 190
  - coding an RDEVICE macro 729
  - configuration guide 48
  - dedicating to virtual machine 477
  - defining 190
  - defining in system configuration file 48

- card punch (*continued*)
  - spooled, virtual machine 562
- card reader
  - coding an RDEVICE macro 727
  - configuration guide 48
  - dedicating to virtual machine 477
  - defining 192
  - defining 3505 192
  - defining in system configuration file 48
  - generating default user form name for 150
  - spooled, defining for virtual machine 562
- case used in system configuration files 56
- CCDUMP utility 321
- CCLOAD EXEC 323
- CCLOAD utility 321
- CCU (Common Control Unit) printer
  - coding an RDEVICE macro 708
- CCW translation 427
  - device macroinstruction 429
- cease
  - new operations on real DASD 118
  - use of X'15' 483
  - users issuing multiple CP commands 483
- change
  - back to IBM-defined user classes 442
  - command classes in directory 443
  - defaults
    - CSE track location format 259
    - for text of status fields 293
    - logical character delete symbol 67
    - logical escape symbol 67
    - logical line delete symbol 67
    - logical line end symbol 67
    - logical tab symbol 67
  - dynamic I/O 105, 141
  - exit point 172
  - hot I/O rate 152
  - online message 292
  - privilege class of CP functions 673
  - privilege classes 433
  - privilege classes authorizing CP functions 184
- CHANGE command 368
- change CSE track location format default 646
- changing information on minidisks 34
- channel path 12, 210, 764
- channel programs 390, 394, 397
- channel-to-channel link
  - defining addresses 350
- CHAR\_DELETE operand
  - CHARACTER\_DEFAULTS statement 67
- character delete symbol
  - definition 67
- CHARACTER\_DEFAULTS statement 51, 67
- CHARS operand
  - RDEVICE statement 202, 213
- checking syntax of statements in a system configuration file 57
- checkpoint
  - data
    - allocating DASD space for 607
    - SYSTEM\_RESIDENCE statement 230
- choose local time zone at IPL 237
- CHOOSE\_LOGO statement 279, 285
- choosing storage size
  - dynamic paging area 590
- CKD (Count-Key-Data)
  - DASD 335, 603
  - device geometry 603
  - devices
    - defining length of map records 269
    - finding CSE track 269
    - using for cross system link 650
- class
  - associating with users and commands 436
  - command privilege defining for virtual machine 572
  - IBM
    - defining for new CP command 84
  - override file
    - activating 440
    - creating 438
    - example 439
    - verifying 440
  - privilege 392
    - authorizing CP functions, changing 184
    - controlling which users can change 136
    - defining for new CP command 84
  - spool file
    - containing classification titles 182
    - specifying for impact printers 202
- CLASS statement 395
- CLASS user directory control statement 443, 445, 462
- CLASSES operand
  - RDEVICE statement 190, 192, 213
  - RDEVICE statement for impact printers 202
- classification title
  - printed output classes containing 182
  - classification titles, printer output 691
- clear
  - minidisks 398
  - T-disks 398
  - TDISK space, specifying whether CP should 136
- CLEAR option on OVERRIDE utility 442
- CLEAR\_TDISK operand
  - disabling on FEATURES statement 141
  - enabling on FEATURES statement 142
- CMS (Conversational Monitor System)
  - command
    - DIRECTXA 448
  - SFS (Shared File System)
    - DASD sharing using 622
  - shared minidisk expanded storage
    - considerations 592
  - utility
    - CPFMTXA 665
    - GENIMAGE 411
    - IMAGELIB 423
    - IMAGEMOD 424
    - VMFHASM 412, 422
- code
  - distribution, defining for a virtual machine 456
- code, executable
  - loading into CP's system execution space 76

- coding
  - HCPRIO ASSEMBLE 705
- collect
  - setting up VM for 297
- command
  - ACNT 299
  - assigning new classes 438
  - assigning to types of users 435
  - ATTACH 478, 623, 665
  - ATTACH XSTORE 592, 593
  - AUTOLOG 144, 296, 299
  - CHANGE 368
  - conventions, notational xvi
  - CPSYNTAX 57
  - creating
    - new CP 84
    - new version of existing CP 84
  - DEDICATE 6, 630
  - DEFINE CPU 5, 534
  - DEFSEG 527
  - DEFSYS 491
  - DETACH XSTORE 593
  - DISABLE 331
  - DISABLE SNA 332
  - DISCARD 747
  - ENABLE 331
  - FORCE 299, 332
  - INDICATE LOAD 601
  - IPL 491
  - LINK 144, 299, 359, 501, 515, 535, 620
  - LOGON 144, 299, 326
  - MESSAGE 142
  - MODIFY 165
  - MSG 276
  - MSGNOH 276
  - notational conventions xvi
  - ORDER 368
  - preventing from issuing multiple 483
  - privilege class
    - defining for virtual machine 462, 572
    - IBM-defined 433
    - redefining 433
  - PURGE 369
  - PURGE IMG 425
  - PURGE UCR 442
  - QUERY
    - LOGMSG 143
    - RETRIEVE 146
  - QUERY (User ID) 277
  - QUERY CACHE 747
  - QUERY COMMANDS 446
  - QUERY DASDFW 748
  - QUERY FILES 369
  - QUERY FRAMES 589
  - QUERY IMG 424
  - QUERY NAMES 277
  - QUERY PINNED 747
  - QUERY READER/PRINTER/PUNCH 369
  - QUERY RSAW 747
  - QUERY UCR 442
  - QUERY UCR COUNT 442

- command (*continued*)
  - QUERY XSTORE 594
  - QUERY XSTORE MAP 592, 593
  - response suppression 86, 166
  - RETAIN XSTORE 591, 593, 596
  - SET CACHE 747
  - SET CONCEAL OFF 534
  - SET D8ONECMD 381
  - SET DASDFW 747
  - SET DUMP 614
  - SET MAXUSERS 535
  - SET PRIVCLASS 382
  - SET SHARED 601, 620
  - SET SHARED OFF 620
  - SET SHARED ON 626
  - SET SRM 332, 537
  - SHUTDOWN 299
  - SMSG 276, 353
  - SPOOL 665
  - SPXTAPE 665
  - START CSECOM 353
  - supported by CSE 335
  - syntax, explanation of xvi
  - TAG 369
  - TERMINAL LINESIZE 331
  - TRANSFER 369
  - WNG 276
  - XAUTOLOG 299, 332
  - XLINK CHECK 353
  - XLINK FORMAT 340, 652
  - XLINK FORMAT (CP) 350
  - XLINK RESET 359
  - XSPOOL QUERY 354
- command table entry block
  - defining new CP 84
- command, cross system
  - excluding virtual machines' input spool files from 274
  - excluding virtual machines' output spool files from 276
  - specifying systems participating in 271
- commands
  - RECORDING command (CP)
    - specifying EREP virtual machine 316
- comments field on SAPL screen 41
- comments in a system configuration file 55
- communication
  - controller
    - coding an RDEVICE macro 757
    - configuration guide 49
    - defining in system configuration file 49
    - setting up a virtual machine for 321
  - enabling SNA 331
  - virtual machine (CVM)
    - establishing connections between 350
    - preparing 350
    - verifying its functioning 353
- communication controller
  - 3705
    - defining 193

- communication controller (*continued*)
  - 3725
    - defining 193
  - 3745
    - defining 193
- COMMUNICATIONS\_USERID operand
  - XSPool\_SYSTEM statement 271
- COMPONENT operand
  - ASSOCIATE MESSAGES statement 64
  - ASSOCIATE MSGS statement 64
- composite reader file 396
- concurrent copy sessions 566
- concurrent virtual and real reserve/release
  - when to use 626
- configuration
  - guide
    - printers 769
  - guide for printers 769
  - overview 23
  - real storage 222
  - real storage, defining 697
  - specifying information 23
- configuration file
  - logo
    - description 279
    - specifying file name and file type 164
    - summary of statements 279
  - system
    - ignoring errors in 241
    - imbedding files into 155
    - summary of statements 51
  - user defaults
    - statements 279
- configuration guide
  - card punches 48
  - card readers 48
  - communication controllers 49
  - DASD 46
  - display devices 49
  - ESCON devices 49
  - other devices 50
  - printers 47
  - tape drives 47
  - unit record devices 48
- CONS operand
  - RIOGEN macro 766
- console
  - address
    - defining list for emergency messages 124
    - defining list for operator consoles 180
  - alternate, defining on RIOGEN macro 766
  - alternates
    - defining 180, 766
  - changing 464
  - defining extended color 553
  - defining extended highlighting 553
  - emergency message
    - defining 124
  - mode, defining 3270 464
- console (*continued*)
  - operator
    - defining 180, 766
  - primary
    - defining 180, 766
  - primary system, defining on RIOGEN macro 766
- CONSOLE operand
  - FORM\_DEFAULT statement 150
- CONSOLE user directory control statement 463
- continuations in system configuration files 55
- CONTROL operand
  - CPXLOAD statement 77
- controller
  - coding an RDEVICE macro 757
  - communication, defining 193
- convention
  - notational
    - for commands xvi
- Coordinated Universal Time (UTC)
  - SYSTIME macro 701
- copy sessions, concurrent 566
- copy spool descriptor 368
- CP
  - starting communication with manually
    - EREP virtual machine 316
- CP (Control Program)
  - accounting exit 310
  - assigning new classes 438
  - attaching Expanded Storage to 596
  - command
    - ACNT 299
    - ATTACH 478, 623, 665
    - ATTACH XSTORE 592, 593
    - AUTOLOG 144, 296, 299
    - CHANGE 368
    - CPSYNTAX 57
    - DEDICATE 6, 630
    - DEFINE CPU 5, 534
    - defining new 84
    - DEFSEG 527
    - DEFSYS 491
    - DETACH XSTORE 593
    - DISABLE 331
    - DISABLE SNA 332
    - DISCARD 747
    - ENABLE 331
    - FORCE 299, 332
    - INDICATE LOAD 601
    - IPL 491
    - LINK 144, 299, 359, 501, 515, 535, 620
    - LOGON 144, 299, 326
    - MESSAGE 142
    - MSG 276
    - MSGNOH 276
    - ORDER 368
    - preventing from issuing multiple 483
    - PURGE 369
    - PURGE IMG 425
    - PURGE UCR 442
    - QUERY (User ID) 277
    - QUERY CACHE 747

CP (Control Program) (continued)

command (continued)

QUERY COMMANDS 446  
QUERY DASDFW 748  
QUERY FILES 369  
QUERY FRAMES 589  
QUERY IMG 424  
QUERY LOGMSG 143  
QUERY NAMES 277  
QUERY PINNED 747  
QUERY RDR/PRT/PUN 369  
QUERY RETRIEVE 146  
QUERY RSAW 747  
QUERY UCR 442  
QUERY UCR COUNT 442  
QUERY VDISK 145  
QUERY XSTORE 594  
QUERY XSTORE MAP 592, 593  
RETAIN XSTORE 591, 593, 596  
SET CACHE 747  
SET CONCEAL OFF 534  
SET D8ONECMD 381  
SET DASDFW 747  
SET DUMP 614  
SET LOGMSG 141  
SET MAXUSERS 535  
SET PFnn RETRIEVE 145  
SET PRIVCLASS 136, 146, 382  
SET RETRIEVE 146  
SET SHARED 601, 620  
SET SHARED OFF 620  
SET SHARED ON 626  
SET SRM 332, 537  
SET VDISK 145  
SHUTDOWN 299  
SMSG 276, 353  
SPOOL 665  
SPXTAPE 665  
TAG 369  
TERMINAL LINESIZE 331  
TRANSFER 369  
WNG 276  
XAUTOLOG 299, 332  
XLINK CHECK 353  
XLINK FORMAT 340, 350, 652  
XLINK RESET 359

disk access

establishing 70

exits

accounting records 310  
HCPACU module 310  
HCPMSU module 373  
message function 373

functions

changing privilege classes authorizing 184

link feature 358

macros

HCPDROP 311, 374  
HCPENTER 311, 374  
HCPEPILG 311, 374  
HCPEXIT 311, 374

CP (Control Program) (continued)

macros (continued)

HCPPROLG 311, 374  
HCPUSING 311

message commands, altering output 373

module

allocating DASD space for 605  
specifying initialization options for 689  
starting CP extensions at initialization time 663

storage

allocating for XSPPOOL trace tables 273  
configuring 222, 697

symptom record

specifying user ID to accumulate 232

system execution space

loading CP routines into 76

system residence disk 693

system residence volume

defining 73  
describing layout 230

use of devices 105

volume

defining list 73  
identifying for cross system spooling 73

CP LINK command

and CSL 359

journaling 385

CP storage for CSE trace tables, establish number of

pages 658

CP\_ACCESS statement 52, 70

CP\_ADDON\_INITIALIZE\_ROUTINES statement 52, 72

CP\_OWNED statement 52, 73, 158

CP-accessed minidisks 31, 33, 34

CP-owned volume list

checking for valid object directory 447

cross system extensions considerations 337

defining 73

generate a list of 664

HCD 7

HCM 7

sequence of statements 57

system use 7

CPACCESS command 393

CPCACHE command 393

CPCACHE FILES file 33

CPCHECKING operand

disabling on FEATURES statement 141

enabling on FEATURES statement 142

CPFMTXA (CMS utility) 7, 665

CPREAD operand

STATUS statement 293

CPRELEASE command 393

CPSYNTAX command 57

CPU power, allocating to users in user directory 555

CPU user directory control statement 465

CPXLOAD statement 52, 76

create

a class override file 438

default system identifier 228

default user form names 150

- create *(continued)*
  - image library
    - map 424
  - list of operator form numbers 247
  - list of user DASD volumes 255
  - list of user form names 247
  - logo configuration file 279
  - logo screens 282
  - new
    - alias for existing CP command 81
    - CP command 84
    - DIAGNOSE code 90
    - exit point 94
    - version of existing CP command 84
  - stand-alone dump utility 403
  - system configuration file 55
  - system date format 225
  - system identifier 226, 228
  - text deck
    - for impact printers 412
    - for the 3800 printer 411
  - the user directory 451
- cross system
  - link
    - defining 359
    - described 358
    - facility utility program 349, 357
    - generating 349
    - in CSE 333
    - initializing 349
    - initializing minidisks for 349
    - maximum number of systems supported 334, 338
    - maximum systems supported 656
    - synchronizing directories 340
    - verifying 353
  - MESSAGE and QUERY commands
    - define virtual machines that will not participate in 659
    - in CSE 334
  - overview of statements 26
  - spool
    - activating 355
    - and CSECOM task 371
    - and PVM 371
    - commands 368
    - define virtual machines that will not participate in 659
    - described 366
    - enabling 353
    - in CSE 334
    - verifying 354
- cross system command
  - excluding virtual machines' input spool files from 274
  - excluding virtual machines' output spool files from 276
  - specifying systems participating in 271
- cross system link
  - excluding systems 263
  - excluding volumes 266
- cross system link *(continued)*
  - including systems 264
  - including volumes 264, 268, 650
- cross system spooling
  - excluding virtual machines' input spool files from 274
  - excluding virtual machines' output spool files from 276
  - identifying volumes for 73
  - specifying systems participating in 271
- trace tables
  - allocating space for 273
- CRYPTO user directory control statement 468
- CSE (Cross-System Extensions)
  - administering 358
  - area
    - default values for 365
    - description of 359
    - format 361
    - formatting 350
    - location 361
    - map records on 363
    - placing 340
    - protecting 361
    - track location format default, change 259, 646
  - capabilities 333
  - choosing minidisks to share 338
  - commands and cross system spool 368
  - complex, system topography 653
  - components 333
  - controlling CSL 359
  - cross system
    - link 333
    - spool 334
  - defining volumes containing minidisks to be shared 340
  - enabling cross-system extensions 345
  - example secondary pre-PROFILE EXEC 344
  - generating CSL 349
  - INCLUDE statement and 487
  - initializing CSL 349
  - installation planning 334
  - installing
    - cross-system extensions 345
    - PVM 345
    - the preparation package 335
  - joining systems 353
  - macroinstruction
    - CSELDEV 646
    - CSELVOL EXCLUDE 649
    - CSELVOL INCLUDE 650
    - CSESYS 653
    - CSETRACE 658
    - CSEUSER 659
  - MDC (minidisk caching feature) 339
  - MDISK statement 339
  - MESSAGE and QUERY commands 334
  - networking implications 357
  - normal operation 368
  - operation, define DASD volumes included in 650
  - operation, define excluded DASD volumes 649



CSE (Cross-System Extensions) *(continued)*  
 outages, preparing for 372  
 overview 333  
 performance considerations for CMS minidisks 339  
 phasing into production environment 356  
 preparing for 341  
 preparing the CVM 350  
 PVM networks 357  
 requirements 334  
 restrictions 335  
 returning to non-CSE operation 372  
 RSCS networks 357  
 setting up 334  
 SNA networks 358  
 spool file limits 341  
 supported features 335  
 synchronizing directories 340  
 SYSAFFIN statement and 487  
 system definition using SYSTEM CONFIG 346  
 trace tables, establish number of pages for CP storage 658  
 track  
   format 361  
   formatting 350  
   location 361  
   map records on 363  
   users supported 334  
 CSE area, multiple cylinders 350  
 CSECOM task  
   and cross system spool 371  
   starting 353, 357  
 CSELDEV macro 646  
 CSELVOL EXCLUDE macro 359, 649  
 CSELVOL INCLUDE macro 359  
 CSESYS (HCPSYS macroinstruction) 653  
 CSETRACE (HCPSYS macroinstruction) 658  
 CSEUSER macro 371, 659  
 CSEVOL macroinstruction  
   EXCLUDE list 338  
   INCLUDE list 338, 650  
 CTCA (Channel-to-Channel Adapter)  
   coding an RDEVICE macro 760  
   defining 205  
   defining in system configuration file 50  
   requirement for CSE 335  
   virtual, defining for virtual machine 557  
 CTCA operand  
   RDEVICE statement 205  
 CYL operand  
   CSELVOL INCLUDE macro 650  
 CYLINDER operand  
   XLINK\_VOLUME\_INCLUDE statement 268

## D

D8ONECMD setting  
 for server virtual machines 381  
 D8ONECMD user directory control statement 483  
 DASD (Direct Access Storage Device)  
 administration overview 7  
 allocating space on 603

DASD (Direct Access Storage Device) *(continued)*  
 avoiding bottlenecks during I/O 337  
 backup and recovery 337  
 coding an RDEVICE macro  
   3350 746  
   3350 (in 3330 compatibility mode) 746  
   3370 749  
   3375 746  
   3380 746  
   3390 746  
   3850 746  
   9332 749  
   9335 749  
   9336 749  
   9345 746  
   FB-512 749  
   FBA DASD 749  
 configuration guide 46  
 count-key-data (CKD) 335  
 CP-owned  
   defining 73  
 CSE area 361  
 dedicated 8, 622  
 defining 195  
 defining in system configuration file 46  
 defining list of CP-owned volumes 73  
 excluding volumes from cross system link 266  
 extended count-key-data (ECKD) 335  
 including volumes in cross system link 268  
 performance measurement 365  
 planning overview 7  
 preparing volumes 361  
 preventing new operations 118  
 preventing shared devices 105  
 restarting 220  
 shared  
   access to minidisks 338  
   defining in system configuration file 46  
   I/O behavior 365  
   performance 365  
   spool files 337  
 sharing  
   among multiple virtual machines and other systems 624  
   using CMS Shared File System 622  
   using concurrent virtual and real reserve/release 625  
   using MPLF 627  
   using Parallel Access Volumes 632  
   using real reserve/release 622  
   without using virtual reserve/release 622  
 sharing devices 105  
 space  
   checkpoint data 607  
   dumps 614  
   named saved systems 615  
   paging space 610  
   spooling space 613  
   used for shared file pools 9  
   warm start data 606  
 stopping new operations 118

DASD (Direct Access Storage Device) *(continued)*

- storage capacities 604
- storage requirements 603
- unsupported
  - defining 209
- used for minidisks 8
- user owned volumes, generating a list of 703
- volumes excluded from CSE 649
- volumes included in CSE 650

DASD format, stand-alone dump 779

DASD operand

- DRAIN statement 118
- HOT\_IO\_RATE statement 152
- RDEVICE statement 195, 209
- START statement 220

DASDOPT user directory control statement 472

DAT (Dynamic Address Translation)

- maintains storage protection 391
- translates addresses 390

data

- adapter
  - coding an RDEVICE macro 752
  - transfer to CP spool files 532

DATAMOVR virtual machine 380

DATEFORMAT user directory control statement 475

deactivation of restricted password

- automatic 386

DEDICATE command 6, 630

DEDICATE user directory control statement 477, 623

dedicated

- AFP printers 708
- cached DASD 630
- DASD 8, 622
- device 477
- device accounting record 300
- devices 622
- HCD 11
- HCM 11
- unit record devices 11

dedicated advanced function printer

- defining 189

default

- for logical character delete symbol 67
- for logical escape symbol 67
- for logical line delete symbol 67
- for logical line end symbol 67
- for logical tab symbol 67
- number of RETRIEVE buffers 136
- system identifier 228
- text of status fields 293
- user form names 150

DEFAULT operand

- CHOOSE\_LOGO statement 286
- HOT\_IO\_RATE statement 153

DEFAULTS command 396

define

- 3800 printers 212
- 3900 printers 212
- access list size, host 581
- accounting virtual machine logon identification 661
- accounting virtual machine user ID 661

define *(continued)*

- advanced function printers 189
- card punch 190
- card readers 192
- channel-to-channel adapters 205
- communication controllers 193
- configuration of real storage 697
- console mode 3270 464
- consoles for emergency messages 124
- DASD 195
- DASD excluded from cross system link 266
- DASD to be included in cross system link 268
- DASD volumes excluded from CSE 649
- DASD volumes included in CSE 650
- default system identifier 228
- dynamic paging area 590
- file for input area 291
- graphic display devices 197
- host access list size 581
- I/O devices 45, 705
- impact printers 201
- journaling facility 160
- LAN adapters 205
- line adapters 193
- list of CP-accessed disks 70
- list of cp-owned DASD volumes 664
- list of CP-owned DASD volumes 73
- list of installation-added entry points 72
- logo picture file 285
- maximum number of nonprimary address spaces 583
- new
  - alias for existing CP command 81
  - CP command 84
  - DIAGNOSE code 90
  - exit point 94
  - version of existing CP command 84
- online message 292
- primary system operator 687
- privilege classes for a virtual machine 444
- real devices 188
- real storage 697
- size of host access list 581
- SNA/CCS to VSCS 331
- special devices 205
- system console 157, 180, 766
- system date format 225
- system identifier 226, 228
- systems in cross system spooling 271
- systems participating in cross system commands 271
- tape units 207
- text of status fields 293
- time zone 239
- total size of nonprimary address spaces 583
- unsupported devices 209
- user IDs to perform special functions 232
- user owned DASD volumes 680
- users' needs 434
- virtual machines for use during hardware outages 372

define (*continued*)  
 virtual machines that will not participate in cross system spooling and cross system message and query commands 659  
 virtual processor 465  
 volumes excluded from user volume list 251, 671  
 volumes to be included in user volume list 253  
 your system 51, 643  
 DEFINE ALIAS statement 52, 81  
 DEFINE CMD statement 52, 84  
 DEFINE COMMAND statement 52, 84  
 DEFINE CPOWNER command 393  
 DEFINE CPU command 5, 534  
 DEFINE DIAGNOSE statement 52, 90  
 DEFINE EXIT statement 52, 94  
 DEFINE LAN statement 52, 97  
 DEFINE TIMEZONE command 393  
 DEFINE VSWITCH statement 52, 100  
 defining an emulated device 121  
 defining global settings for use during user processing 486  
 defining virtual LAN segment 97  
 defining virtual switch segment 100  
 definition of  
   set of real devices, adding to 188  
 DEFSEG command  
   NAMESAVE statement relationship 527  
 DEFSYS command 491  
 degaussing 398  
 DELAY operand  
   CPXLOAD statement 78  
 delete  
   exit point 172  
   members from an image library 424  
 delete symbol, defining for virtual machine 572  
 describe topography of CSE complex 653  
 DEST operand  
   RDEVICE statement 202, 213  
 DETACH XSTORE command 593  
 determine local time zone 237  
 DEVCLASS operand  
   RDEVICE statement 209  
 device  
   adding real 188  
   block  
     building 105  
     dynamic allocation of 689  
   bringing online 105  
   class definitions 773  
   defining 45, 529, 557, 705  
   defining for virtual machine expanded storage facility 585  
   disabling for dynamic I/O changes 105  
   enabling for dynamic I/O changes 105  
   features by class and type 775  
   initializing  
     after IOMCK 136  
     at IPL 105  
     when added to the system 136  
   limiting I/O rate 105  
   measuring subchannels 105

device (*continued*)  
 model definitions 775  
 preventing  
   DASD sharing 105  
   initialization at IPL 105  
 sensing 46  
 sharing DASD 105  
 stand-alone dump 405  
 supported 45  
 tables for unsupported devices 428  
 taking offline 105  
 type definitions 773  
 unit record 11  
 unsupported 46  
   coding an RDEVICE macro 763  
 device class assemble files 427  
 device number field on SAPL screen 40  
 Device Support Facility  
   allocating DASD space for a directory 608  
   used in switching 3390 operating modes 616  
 DEVICES statement 52, 105  
 DFSMS/VM  
   setting up virtual machines for 327  
 DIAGNOSE code  
   assigning new classes 438  
   creating new 90  
   defining new 90  
   restriction of 394  
   using to initiate accounting records 310  
 X'04' 394  
 X'08' 381, 394, 483, 538  
 X'14' 532  
 X'28' 394  
 X'3C' 394  
 X'4C' 310, 394, 533, 538  
 X'64' 527  
 X'7C' 394  
 X'84' 394, 535  
 X'88' 535  
 X'98' 380, 390, 394, 535, 538  
 X'98' option 395, 397  
 X'A0' 384  
 X'DC' 533  
 X'E4' 382, 501, 515, 534, 535, 536  
 X'F8' 537  
 directory  
   allocated space for 480  
   calculating DASD space for 455  
   changing 443  
   changing a user 447  
   CLASS 443  
   control  
     MDISK 339  
     SPOOLFILE 341  
   CP system 395  
   creating a user 447  
   definition entry location 504  
   entry  
     example for user class restructure 445  
     nonshared 342  
   options, RSCS 395

directory (*continued*)  
   password authorization bypass 385  
   profiles 455  
   Shared File System 622  
   single source  
     bringing online 357  
     defining volumes and cylinders 341  
   space 338  
   update-in-place 396  
   updating a user 447  
 DIRECTORY user directory control statement 350,  
 448, 480  
 DIRECTXA utility 448  
   checking directory for errors 454  
   checking directory for VM/SP directory  
   statements 455  
   checking for VM/SP statements 455  
   creating user directory with 451  
   running the directory creation program 453  
 DIRMAINT (Directory Maintenance) licensed program  
   avoiding manual updates 381  
   DATAMOVR virtual machine 380  
   dual registration 381  
   general description 379  
 disable  
   dynamic I/O changes  
     for specific devices 105  
     on your processor 136  
   I/O throttling 105  
   log on password suppression 136  
 DISABLE CMD statement 52, 110  
 DISABLE command 331  
 DISABLE COMMAND statement 52, 110  
 DISABLE DIAGNOSE statement 52, 112  
 DISABLE EXITS statement 52, 114  
 DISABLE operand  
   ASSOCIATE EXIT statement 61  
   DEFINE ALIAS statement 82  
   DEFINE CMD statement 85  
   DEFINE COMMAND statement 85  
   DEFINE DIAGNOSE statement 90  
   FEATURES statement 140  
   JOURNALING statement 162  
 DISABLE SNA command 332  
 disabled wait state if parm disk is not  
   CMS-formatted 51  
 DISCARD command 747  
 DISCONNECT operand  
   SYSTEM\_USERIDS statement 233  
 disconnect status of system operator, specifying 232  
 disconnecting  
   EREP virtual machine 317  
   primary system operator  
   for EREP 316  
 discontinuous  
   expanded storage 593  
   saved segment  
   modifying read-only 390  
 disk  
   clearing temporary 384  
   disk (*continued*)  
     CP-owned  
       defining 73  
       preventing new operations 118  
       stopping new operations 118  
   DISK LOAD command 396  
   display  
     devices  
       defining 197  
       unsupported, defining 209  
   DISPLAY HOST command 393  
   display printers 713, 769  
   display terminal  
     coding an RDEVICE macro 733, 754  
       3101 734  
       3178 734  
       3179 734  
       3180 734  
       3191 734  
       3192 734  
       3193 734  
       3194 734  
       3250 734  
       3270-PC 734  
       3277 734  
       3278 734  
       3279 734  
       3290 734  
       5080 740  
       5550 734  
       6090 740  
       7171 Device Attachment Control Unit 734  
     coding an RDEVICE macro for 734  
     configuration guide 49  
     defining in system configuration file 49  
     establishing the SNA environment 330  
     graphic  
       coding an RDEVICE macro for 5080 740  
       coding an RDEVICE macro for 6090 740  
     operator identification card 49  
     planning for 10  
   DISPLAY\_STORAGE operand  
     TRANSLATE\_TABLE statement 243  
   displaying  
     commands available to a user 446  
   distributed IUCV 116  
   document processor  
     coding an RDEVICE macro 731  
   documentation  
     considerations when redefining classes 438  
   DPA (Dynamic Paging Area)  
     size considerations 590  
   DPMSIZE operand  
     delayed purge queue 213  
     RDEVICE statement 213  
   DPS operand  
     RDEVICE statement 210  
   DRAIN statement 52, 118  
   DRAWLOGO sample utility program  
     description 642  
     usage 282

DROPBUF command 399  
 DTIGEN macro 330  
 dual registration 381  
 dump  
   space  
     allocating DASD space for 614  
     stand-alone, utility 403  
     symptom records  
       specifying user ID to accumulate 232  
 DUMP HOST command 393  
 DUMP operand  
   CP\_OWNED statement 74  
   SYSTEM\_USERIDS statement 233  
 dumps 396  
 dynamic  
   I/O reconfiguration 12  
   switching device  
     coding an RDEVICE macro 762  
 dynamic I/O  
   disabling  
     for specific devices 105  
     on your processor 136  
   enabling  
     for specific devices 105  
     on your processor 136  
   on your processor 136  
 dynamic I/O features  
   on your processor 136  
 dynamic switching device  
   defining 209

**E**

EAST operand  
   TIMEZONE\_DEFINITION statement 239  
 ECKD (Extended Count Key Data)  
   devices  
     defining length of map records 269  
     finding CSE tracks 269  
     using for cross system link 650  
 EDEVICE statement 52, 121  
 ELAN 738  
 emergency message  
   defining  
     consoles to receive 124  
 EMERGENCY\_MESSAGE\_CONSOLES  
   statement 52, 124  
 emulated device, defining 121  
 EMULATED\_3270 operand  
   RDEVICE statement 198  
 enable  
   cross system spool 353  
   dynamic I/O changes  
     for specific devices 105  
     on your processor 136  
   I/O throttling 105  
   log on password suppression 136  
   SNA communication 331  
   steps for CSE 345  
   system to set and query journaling facility 160  
 ENABLE CMD statement 52, 126

ENABLE command 331  
 ENABLE COMMAND statement 52, 126  
 ENABLE DIAGNOSE statement 52, 128  
 ENABLE EXITS statement 52, 130  
 ENABLE operand  
   ASSOCIATE EXIT statement 61  
   DEFINE ALIAS statement 82  
   DEFINE CMD statement 85  
   DEFINE COMMAND statement 85  
   DEFINE DIAGNOSE statement 91  
   FEATURES statement 142  
 enabling, CP exit 43  
 end  
   new operations on real DASD 118  
   use of X'15' 483  
   users issuing multiple CP commands 483  
 end HCPSYS assembly 668  
 ENFORCE\_BY\_VOLID statement 52, 132  
 enforcing  
   attachment of DASD devices by their VOLIDs 132  
   link protocols 358  
 entry block, command table  
   defining new CP 84  
 entry point  
   assigning to exit point 60, 64  
   specifying for new CP command 84  
 EP (Emulation Program)  
   setting up a virtual machine for 321  
   used with 3720 49  
 EPNAME operand  
   ASSOCIATE EXIT statement 61  
   ASSOCIATE MESSAGES statement 65  
   ASSOCIATE MSGS statement 65  
   DEFINE CMD statement 85  
   DEFINE COMMAND statement 85  
   DEFINE DIAGNOSE statement 91  
   EXTERNAL\_SYNTAX statement 135  
   MODIFY CMD statement 166  
   MODIFY COMMAND statement 166  
   MODIFY DIAGNOSE statement 170  
 EQUATE statement 52, 133  
 EREP (Environmental Record Editing and Printing)  
   specifying user ID to accumulate records 232  
 EREP operand  
   of RECORDING command 316  
 EREP records  
   retrieval of  
     starting manually 316  
 EREP virtual machine  
   specifying 316  
   starting record retrieval manually 316  
 EREP1 operand  
   SYSTEM\_USERIDS statement 233  
 EREP2 operand  
   SYSTEM\_USERIDS statement 233  
 error  
   ignoring system configuration file 241  
   recording  
     setting up virtual machine(s) for 314  
     virtual machine, defining a user ID for 669

- escape character
  - default 67
  - setting 67, 572
- ESCAPE operand
  - CHARACTER\_DEFAULTS statement 67
- ESCM (Enterprise Systems Communication Manager)
  - \*CONFIG system service 494, 497
- ESCON devices
  - configuration guide 49
  - defining in system configuration file 49
- ESM (External Security Manager)
  - using to audit security 384
  - using to protect security 384
- establish
  - default for
    - logical character-delete symbol 67
    - logical escape symbol 67
    - logical line-delete symbol 67
    - logical line-end symbol 67
    - logical tab symbol 67
    - number of RETRIEVE buffers 136
    - system identifier 228
    - text of status fields 293
  - file mode 70
  - maximum number of users 136
  - system attributes 136
  - system date format 225
  - system identifier 226
- establish number of pages for CP storage for CSE trace tables 658
- Ethernet communication adapter
  - defining in system configuration file 49
- example
  - 3203 UCS buffer image 416
  - 3211 UCS buffer image 417
  - 3262 UCS buffer image 417
  - alias, creating for CP command 83
  - assigning commands to kinds of users 435
  - assigning entry points to exit points 63
  - associating privilege classes with commands and users 436
  - changing privilege classes 168, 171
  - CHOOSE\_LOGO statements 289
  - CKD-formatted CSE area map 361, 362
  - class override file 439
  - coding FCB macroinstruction for 4248 printer 421
  - commands, disabling CP 111
  - commands, enabling CP 127
  - communications controller functions 322
  - configuration file statement
    - ALTERNATE\_OPERATORS 59
    - ASSOCIATE EXITS 63
    - ASSOCIATE MESSAGES 66
    - ASSOCIATE MSGS 66
    - CHARACTER\_DEFAULTS 69
    - CP\_ACCESS 71
    - CP\_ADDON\_INITIALIZE\_ROUTINES 72
    - CP\_OWNED 75, 159
    - CPXLOAD 80
    - DEFINE COMMAND 83, 88
    - DEFINE DIAGNOSE 93
- example (continued)
  - configuration file statement (continued)
    - DEFINE EXIT 96
    - DEVICES 108
    - DISABLE CMD 111
    - DISABLE COMMAND 111
    - DISABLE DIAGNOSE 113
    - DISABLE EXITS 115
    - DRAIN 119
    - EMERGENCY\_MESSAGE\_CONSOLES 125
    - ENABLE CMD 127
    - ENABLE COMMAND 127
    - ENABLE DIAGNOSE 129
    - ENABLE EXITS 131
    - EQUATE 134
    - FEATURES 134, 147, 156
    - FORM\_DEFAULT 150
    - HOT\_IO\_RATE 154
    - IMBED 155
    - INIT\_MITIME 157
    - JOURNALING 163
    - LOGO\_CONFIG 164
    - MODIFY COMMAND 168
    - MODIFY DIAGNOSE 171
    - MODIFY EXIT 174
    - OPERATOR\_CONSOLES 181
    - PRINTER\_TITLE 183
    - PRIV\_CLASSES 185
    - RDEVICE (3800 printers) 215
    - RDEVICE (AFP) 189
    - RDEVICE (card punches) 191
    - RDEVICE (card readers) 192
    - RDEVICE (DASD) 196
    - RDEVICE (graphic display devices) 199
    - RDEVICE (impact printers) 204
    - RDEVICE (line adapters) 194
    - RDEVICE (special devices) 205
    - RDEVICE (tape units) 207
    - RDEVICE (unsupported devices) 211
    - START 119, 221
    - STORAGE 224
    - SYSTEM\_DATEFORMAT 225
    - SYSTEM\_IDENTIFIER 227, 228
    - SYSTEM\_IDENTIFIER\_DEFAULT 227, 228
    - SYSTEM\_RESIDENCE 231
    - SYSTEM\_USERIDS 234
    - THROTTLE 236
    - TIMEZONE\_BOUNDARY 237, 238
    - TIMEZONE\_DEFINITION 239
    - TOLERATE\_CONFIG\_ERRORS 242
    - TRANSLATE\_TABLE 245
    - USER\_DEFAULTS 250
    - USER\_VOLUME\_EXCLUDE 252
    - USER\_VOLUME\_INCLUDE 254
    - USER\_VOLUME\_LIST 256
    - USERFORM 247
    - XLINK\_SYSTEM\_EXCLUDE 263
    - XLINK\_SYSTEM\_INCLUDE 264
    - XLINK\_VOLUME\_EXCLUDE 266
    - XLINK\_VOLUME\_INCLUDE 270
    - XSPPOOL\_SYSTEM 272

example (continued)

- configuration file statement (continued)
  - XSPPOOL\_TRACE 273
  - XSPPOOL\_XLIST\_INPUT 275
  - XSPPOOL\_XLIST\_OUTPUT 277
- CP command, creating alias for 83
- CP system execution space, loading CP routines into 80
- CPACCESS command 32, 36
- CPCACHE command 33
- CPCACHE FILES file 33
- CPLISTFILE command 33, 34
- CPRELEASE command 32, 35
- CPTYPE command 34
- creating alias for CP command 83
- CSE spooling commands 369
- dedicated DASD 622
- dedicating a device 622
- defining minidisk on a cached DASD 630
- DIAGNOSE codes, disabling 113
- DIAGNOSE codes, enabling 129
- directory calculation 609
- directory entries for CSE 342
- directory entries for user class restructure 445
- disabling CP commands 111
- disabling DIAGNOSE codes 113
- displaying
  - DIAGNOSE codes 93
- ECKD-formatted CSE area map 364
- enabling CP commands 127
- enabling DIAGNOSE codes 129
- entry points, assigning to exit points 63
- exit point
  - assigning entry points to 63
  - defining 96
  - modifying 174
  - start having CP use 131
  - stopping CP from using 115
- expanded storage fragmentation 594
- generating the stand-alone dump utility 406
- GENIMAGE utility 411
- INPUT\_AREA statement 291
- IPL command 39
- LINK command 35, 620
- loading CP routines into CP's system execution space 80
- macro
  - CSESYS 657
  - SYSACNT 661
  - SYSADDIN 663
  - SYSMPVOL 666
  - SYSDUMP 667
  - SYSEND 668
  - YSEREP 669
  - SYSEXCL 671
  - SYSFCN 674
  - SYSFORM 676
  - SYSID 679
  - SYSINCL 680
  - SYSJRL 684
  - SYSMAXU 686

example (continued)

- macro (continued)
  - SYSOPR 688
  - SYSPLAS 692
  - SYSRES 695
  - SYSSYMP 699
  - SYSTIME 702
  - SYSUVOL 703
- message repositories, using local 66
- named saved system calculation 615
- ONLINE\_MESSAGE statement 292
- operator restrictions to increase system security 382
- OVERRIDE control statement 439
- overriding commands 168
- overriding DIAGNOSE codes 171
- paging space calculation 611
- pre-profile exec 342
- PVM 323
- QUERY CPDISKS command 31, 35
- QUERY LOGMSG command 31, 35
- RDEVICE macro
  - 2741 755
  - 3151 755
  - 3161 755
  - 3162 755
  - 3163 755
  - 3167 755
  - 3800 printer 725
  - advanced function printers 709
  - card punch 730
  - card readers 728
  - communication controllers 755, 758
  - CTCA 760
  - DASD 748
  - defining system console 767
  - display printers 714
  - display terminals 733, 736, 740
  - document processor 731
  - FBA DASD 750
  - line adapters 752
  - MCCU 760
  - printers 710, 719
  - tape drives 743
  - telecommunication integrated adapters 739
  - unsupported DASD 765
- redefining commands 168
- redefining DIAGNOSE codes 171
- RIOGEN macro 767
- SAPL screen 40
- secondary pre-PROFILE EXEC 344
- SET SHARED command 620
- SET SHARED ON command 626
- share minidisk among virtual machines 619
- spooling space calculation 615
- start having CP use exit point 131
- STATUS statement 294
- stopping CP from using exit point 115
- SYSEXCL macro 671
- SYSINCL macro 680
- SYSOPTS macro 690

- example (*continued*)
  - SYSSTORE macro 698
  - system execution space, loading CP routines into 80
  - turning on or off
    - DIAGNOSE codes 113, 129
  - user directory control statement
    - ACCOUNT 457
    - ACIGROUP 458
    - APPCPASS 460
    - CLASS 462
    - CONSOLE 464
    - CPU 466
    - CRYPTO 470
    - D8ONECMD 484
    - DATEFORMAT 476
    - DEDICATE 479
    - DIRECTORY 481
    - INCLUDE 487
    - IPL 492
    - IUCV 498
    - LINK 503
    - LOAD 506
    - LOGONBY 509
    - MACHINE 511
    - MDISK 519
    - MINIOPT 526
    - NAMESAVE 528
    - NICDEF 531
    - NOPDATA 532
    - OPTION 538
    - POOL 540
    - POSIXGLIST 542
    - POSIXGROUP 544
    - POSIXINFO 547
    - POSIXOPT 549
    - PROFILE 552
    - SCREEN 554
    - SHARE 556
    - SPECIAL 560
    - SPOOL 564
    - SPOOLFILE 565
    - SYSAFFIN 570
    - USER 576
    - XAUTOLOG 578
    - XCONFIG 584
    - XSTORE 586
  - using local message repositories 66
  - utility programs 641
  - VMLINK CMS command 620
  - warm start area calculation 607
- exclude
  - DASD volumes from cross system link 266, 649
  - input spool files from cross system spooling and commands 274, 276
  - systems from cross system link 263
  - volumes from user volume list 251, 671
- exclusion list
  - including user ID on 371
- exclusive
  - access mode 327, 387
- exclusive (*continued*)
  - access mode, authorized to use 535
- EXEC
  - HCPADMP 409
- executable code
  - loading into CP's system execution space 76
- exit point
  - assigning entry points and external symbols to 60
  - assigning entry points to 64
  - changing 172
  - creating 94
  - permitting CP to call entry points and external symbols associated with 130
  - preventing CP from calling entry points and external symbols associated with 114
  - removing 172
- exits
  - accounting
    - CP 310
  - CP
    - accounting records 310
    - HCPACU module 310
    - HCPMSU module 373
    - message function 373
  - HCPACU module
    - contents 311
    - entry point HCPACUOF 313
    - entry point HCPACUON 312
    - function 310
    - usage notes 311
    - when called by CP 310
  - HCPMSU module
    - contents 373
    - entry point HCPMSUEX 374
    - function 373
    - process flow 377
    - usage notes 373
    - when called by CP 373
  - installation-wide
    - accounting, CP 310
    - CP accounting records 310
    - CP message function 373
    - HCPACU module 310
    - HCPMSU module 373
- expanded storage facility
  - administration overview 7
  - attaching to a virtual machine 592
  - attaching to CP 596
  - considerations with shared CMS minidisks 592
  - dedicating to a virtual machine 585
  - discontinuous 593
  - fragmentation 593
  - mapping 593
  - overview 591
  - planning 9
  - planning overview 7
  - retained 596, 597
  - use as a minidisk cache 597
- extended
  - color option
    - defining 553



- extended (*continued*)
  - count-key-data (CKD)
    - DASD 335
  - highlighting option
    - defining 553
- extended-configuration facility control parameters 580
- extensions, initialization time starting CP 663
- extent field on SAPL screen 41
- external symbol
  - assigning to exit point 60, 64
- EXTERNAL\_SYNTAX statement 53, 135

## F

- FACILITIES operand
  - JOURNALING statement 161
- FAIL operand
  - D8ONECMD statement 483
- FBA (Fixed-Block Architecture)
  - DASD 603
    - coding an RDEVICE macro 749
  - device geometry 604
  - fundamentals 603
- FCB (Forms Control Buffer)
  - 3203 printer 413
  - 3211 printer 413
  - 3262 printer 413
  - 4245 printer 413
  - 4248 printer 413
  - adding a new image
    - 3203 printer 418
    - 3211 printer 418
    - 3262 printer 418
    - 4245 printer 418
    - 4248 printer 418, 420
  - example of new image 419, 421
  - IBM provided 418
  - macroinstruction 418, 420
  - naming convention 416
- FCB operand
  - RDEVICE statement 202, 214
- FCP operand
  - RDEVICE statement 205
- FEATURES statement 53, 136
- file
  - adding to an image library 424
  - CPCACHE FILES 33
  - defining for input area 291
  - list used by SAPL 42
  - logo configuration
    - description 279
    - specifying file name and file type 164
    - statements 279
    - summary of statements 279
  - logo picture
    - choosing 285
    - selection 288
    - separator pages 289, 290
    - width 288
  - major VM/ESA to know 3

- file (*continued*)
  - pool
    - defined 622
    - server machine 622
  - spool
    - class, specifying for impact printers 202
  - system configuration
    - ignoring errors in 241
    - imbedding files into 155
    - summary of statements 51
    - USER\_DEFAULTS 248
  - user class restructure HELP 437
  - user defaults
    - specifying default attributes 248
- file mode
  - establishing 70
- file name
  - of logo configuration file, specifying 164
  - of system configuration file, specifying 42
- FILE operand
  - CHOOSE\_LOGO statement 287
- file type
  - of logo configuration file, specifying 164
  - of system configuration file, specifying 43
- FILTER operand
  - TRANSLATE\_TABLE statement 243
- FLASH operand
  - RDEVICE statement 214
- FOLDUP operand
  - RDEVICE statement 203
- FOLLOWING operand
  - ASSOCIATE EXIT statement 60
  - ASSOCIATE MESSAGES statement 64
  - ASSOCIATE MSGS statement 64
- FORCE command 299, 332
- form
  - names, creating list of user 247
  - numbers, creating list of operator 247
  - specifying default values 675
- FORM operand
  - RDEVICE statement for 3800 printers 214
  - RDEVICE statement for card punches 190
  - RDEVICE statement for impact printers 203
- FORM\_DEFAULT statement 53, 150
- format
  - CKD-formatted CSE area 361
  - CSE area 350
  - ECKD-formatted CSE area 363
  - system configuration file 55
- fragmentation
  - expanded storage 593
- free storage used by CP for a user ID 538
- full-pack minidisk 625, 630
  - restricting and auditing 382
- FULLPROMPT 396

## G

- gateway
  - system
    - identifying 226, 228

GATEWAY operand  
 SYSTEM\_IDENTIFIER statement 226  
 SYSTEM\_IDENTIFIER\_DEFAULT statement 228

generate  
 default system identifier 228  
 default user form names 150  
 list of cp-owned DASD volumes 664  
 list of CP-owned volumes 73  
 list of operator form numbers 247  
 list of user DASD volumes 255  
 list of user form names 247  
 system date format 225  
 system identifier 226, 228  
 XLINK utility program 349

GENIMAGE utility 411

global attributes 257

GLOBALDEFS user directory control statement 485

GLOBALOPTS user directory control statement 486

GLOBALV variables 399

glossary information 785

GRAF operand  
 HOT\_IO\_RATE statement 152

graphic display devices  
 defining 197

GRAPHICS\_APL operand  
 TRANSLATE\_TABLE statement 243

GRAPHICS\_NONAPL operand  
 TRANSLATE\_TABLE statement 243

GRAPHICS\_TEXT operand  
 TRANSLATE\_TABLE statement 243

group name, defining for virtual machine 458

group of systems, naming 133

guest devices, attaching 478

## H

halt  
 new operations on real DASD 118  
 use of X'15' 483  
 users issuing multiple CP commands 483

hardware, cabling 341

HCD  
 CP-owned DASDs 7  
 DASDs used for minidisks 8  
 dedicated DASDs 8  
 dedicated unit record devices 11  
 I/O reconfiguration 12  
 IODF statement 53, 158  
 NOHCD IPL option 43  
 spooled unit record devices 11  
 terminals 10

HCM  
 CP-owned DASDs 7  
 DASDs used for minidisks 8  
 dedicated DASDs 8  
 dedicated unit record devices 11  
 I/O reconfiguration 12  
 IODF statement 53, 158  
 NOHCD IPL option 43  
 spooled unit record devices 11  
 terminals 10

HCPACU exit module  
 contents 311  
 function 310  
 HCPACUOF entry point  
 AOFARM parameter list 314  
 function 313  
 input registers 312, 314  
 return code 314  
 when called by CP 313

HCPACUON entry point  
 AONPARM parameter list 312  
 function 312  
 input registers 312  
 return codes 313  
 when called by CP 312

usage notes 311  
 when called by CP 310

HCPBOX ASSEMBLE file  
 overriding 282

HCPDROP macro  
 HCPACU module 311  
 HCPMSU module 374

HCPENTER macro  
 HCPACU module 311  
 HCPMSU module 374

HCPEPILG macro  
 HCPACU module 311  
 HCPMSU module 374

HCPEXIT macro  
 HCPACU module 311  
 HCPMSU module 374

HCPMSU exit module  
 contents 373  
 function 373  
 HCPMSUOX entry point  
 function 374  
 input registers 374  
 parameter list 374  
 return codes 374  
 when called by CP 374

process flow 377  
 usage notes 373  
 when called by CP 373

HCPPROLG macro  
 HCPACU module 311  
 HCPMSU module 374

HCPRIO ASSEMBLE file  
 coding 705  
 device support 45  
 macroinstructions 45, 705

HCPSADMP example 407

HCPSADMP EXEC 403, 409

HCPSAVBK save area  
 contents 312  
 used by CP accounting exit 312  
 used by CP message function exit 374

HCPSDC ASSEMBLE file 404

HCPSDC TEXT file 404

HCPSYS ASSEMBLE file  
 coding 644  
 macroinstruction 643

HCPYSYS ASSEMBLE file (*continued*)

- CSELDEV 646
- CSELVOL EXCLUDE 649
- CSELVOL INCLUDE 650
- CSESYS 653
- CSETRACE 658
- CSEUSER 659
- SYSACNT 661
- SYSADDIN 663
- SYSCPVOL 447, 664
- SYSDUMP 667
- SYSEND 668
- SYSEREP 669
- SYSEXCL 671
- SYSFCN 575, 673
- SYSFORM 675
- SYSID 677
- SYSINCL 680
- SYSJRL 682
- SYSMAXU 686
- SYSOPR 687
- SYSPLAS 691
- SYSRES 693
- SYSSTORE 697
- SYSSYMP 699
- SYSTIME 701
- SYSUVOL 703

HCPUSING macro, HCPACU module 311

header for global definition section 485

HELP files 437

high-level data link control (X.25 network) 738

HIPERSockets operand

- RDEVICE statement 205

HOLD operand

- RDEVICE statement 198

HOLDING operand

- STATUS statement 293

host access list size, defining 581

hot I/O rate

- changing 152
- setting 152

HOT\_IO\_RATE statement 53, 152

HW\_SERVICE operand

- PRIV\_CLASSES statement 184

**I**

I/O (input/output)

- avoiding DASD bottlenecks 337
- configuration
  - IOCP 184, 673, 705
  - task overview 12
- device
  - definitions 45
  - distributing load among them 336
  - sensing 45
- devices
  - setting hot I/O rate for 152
- dynamic
  - disabling for specific devices 105
  - disabling on your processor 136

I/O (input/output) (*continued*)

- dynamic (*continued*)
  - enabling for specific devices 105
  - enabling on your processor 136
- machine check
  - initializing devices after 136
- performance measurement
  - for shared DASD 365
- reconfiguration 12
- reconfiguration by z/VM 12

I/O rate, hot

- changing 152
- setting 152

IBM-defined user class

- changing back to 442

IBM1\_ADAPTER operand

- RDEVICE statement 193

IBMCLASS operand

- DEFINE ALIAS statement 82
- DEFINE CMD statement 86
- DEFINE COMMAND statement 86
- DISABLE CMD statement 110
- DISABLE COMMAND statement 110
- ENABLE CMD statement 126
- ENABLE COMMAND statement 126
- MODIFY CMD statement 166
- MODIFY COMMAND statement 166

identification of users, requirements for 391

identify

- 4-digit years 225
- minidisk to CP 70
- processor running z/VM 226, 228
- system gateway 226, 228
- volumes for cross system spooling 73

ignore

- system configuration file errors 241

ILAN 738

image library

- adding files to 424
- deleting members from 424
- description of 411
- displaying information about 424
- installing 423
- keeping backup copies of 425
- map creating 424
- modifying 424
- printer
  - creating 411
  - modifying 411
- purging 425
- replacing members 424
- replacing modules 424

IMAGE\_LIBRARY operand

- RDEVICE statement 203, 214

IMAGELIB utility 423

IMAGEMOD utility 424

imbed files into system configuration file 155

IMBED statement 53, 155

impact printer 716

- creating image library for 424
- creating text deck for 412

- impact printer (*continued*)
  - defining 201
  - image library, description 411
- IMPACT\_PRINTER operand
  - RDEVICE statement 202
- include
  - DASD in cross system link 268
  - DASD volumes in cross system link 650
  - journaling facility in system 160
  - password suppression facility in system 136
  - systems in cross system link 264
  - volumes in user volume list 253, 680
- INCLUDE user directory control statement 487
- INDEX operand
  - RDEVICE statement 203
- INDICATE LOAD command 601
- INIT\_MITIME statement 53, 157
- initial state of CP file system 31
- initialize
  - devices
    - after IOMCK 136
    - when added to the system 136
  - minidisk volumes for cross system link 349
  - options for CP, specifying 689
  - preventing devices at IPL 105
  - specific devices at IPL 105
- input
  - area, defining file for 291
- INPUT operand
  - TRANSLATE\_TABLE statement 244
- INPUT\_AREA statement 279, 291
- input/output configuration (IOCP) utility 176, 184, 673, 705
- install
  - cross system extension 345
  - CSE, planning 334
  - image library 423
- integrated communication adapters
  - defining 738
  - defining in system configuration file 49
- Integrated Cryptographic Facility 383
  - defining for virtual machines 388
  - maximum number of users 388
  - use with VM/ESA 387
- integrity problems, reporting 400
- internal trace table
  - allocating space for 222, 697
- interpretive-execution mode 390
- interrupt, unsolicited
  - changing 152
  - setting 152
- INVAR operand
  - DEFINE DIAGNOSE statement 91
- INVXC operand
  - DEFINE DIAGNOSE statement 91
- IOCP (input/output configuration) 184, 673, 705
- IOCP\_READ operand
  - PRIV\_CLASSES statement 184
- IOCP\_WRITE operand
  - PRIV\_CLASSES statement 184
- IOPRIORITY user directory control statement 488

- IPL (Initial Program Load)
  - imbedding files into system configuration file 155
  - parameters on SAPL screen 41
  - parameters passed via SAPL 42
  - process 31
- IPL at log on 490
- IPL command 491
- IPL user directory control statement 490
- I POLL function, with GCS 332
- ISFC (Inter-System Facility for Communications)
  - accounting record 304
- isolation 337
- IUCV (Inter-User Communication Vehicle)
  - authorizing virtual machines that use 493
  - IUCV user directory control statement 493

## J

- joining systems 353
- journal
  - accounting record 301
  - AUTOLOG command 385
  - including 682
  - LINK command 385
  - LOGON command 385
  - XAUTOLOG command 385
- journaling
  - security
    - including facility in system 160
- JOURNALING statement 53, 160

## L

- LAN (Local Area Network)
  - 802.3 738
  - adapters
    - defining 205
    - IBM Token Ring 738
  - LAN\_ADAPTER operand
    - RDEVICE statement 205
  - LDEV operand
    - CHOOSE\_LOGO statement 286
  - LET operand
    - CPXLOAD statement 78
  - limit
    - I/O rate for specific devices 105
    - number of retrieve buffers 136
    - number of users 136
  - LIMIT operand
    - RDEVICE statement 203, 214
  - line adapters
    - coding an RDEVICE macro 754
    - defining 193
  - line delete character
    - defining for virtual machine 572
    - definition 67
  - line end character
    - defining for virtual machine 572
    - definition 67
  - LINE\_DELETE operand
    - CHARACTER\_DEFAULTS statement 68

LINE\_END operand  
   CHARACTER\_DEFAULTS statement 68  
 link  
   access modes 387  
   cross system  
     excluding systems 263  
     excluding volumes 266, 649  
     including systems 264  
     including volumes 264, 268, 650  
   defining 359  
   defining minidisk 358  
   protocols 358  
   verifying 353  
 LINK command 299, 501, 515, 535, 620  
   prompting for password on 144  
 LINK operand  
   JOURNALING statement 161  
 LINK statements 395  
 LINK user directory control statement 9, 500, 620  
 LINKS option  
   DRAIN statement 119  
   START statement 220  
 list  
   defining size of host access 581  
   of consoles to receive emergency messages 124  
   of CP-owned volumes, defining 73  
   of possible system consoles, defining 180  
   of user DASD volumes, generating 255  
   user form names, creating 247  
 load  
   CP routines into CP's system execution space 76  
 load origin field on SAPL screen 41  
 LOAD user directory control statement 504  
 LOADDEV user directory control statement 507  
 LOCAL operand  
   CHOOSE\_LOGO statement 286  
 local time zone  
   choosing at IPL 237  
   defining 239  
   determining 237  
 LOCATE command 393  
 location, CSE area 361  
 LOCK operand  
   CPXLOAD statement 77  
   D8ONECMD statement 483  
 lock pages 394, 397  
 LOCKOUT operand  
   JOURNALING statement 162  
 log  
   messages  
     QUERY LOGMSG command 31  
     specifying whether CP should display 136  
 log on  
   automatically  
     recording invalid attempts 160  
   defining new CP command to issue before 84  
   identification  
     accounting virtual machine 661  
     error recording virtual machine 669  
     primary system operator 687  
     system dump virtual machine 667  
   log on (*continued*)  
     identification, defining virtual machine user 572  
     limit  
       overriding 535  
       specifying 686  
     maximum number of users  
       specifying 136  
     password defining for virtual machine user 572  
     password suppression 387  
       including facility in system 136  
 LOG operand  
   D8ONECMD statement 483  
 logical character delete symbol  
   establishing default 67  
 logical device  
   choosing logo 285  
 logical escape character  
   establishing default 67  
 logical line delete character  
   establishing default 67  
 logical line end character  
   establishing default 67  
 logical symbol, establishing default  
   for character delete 67  
   for escape 67  
   for line delete 67  
   for line end 67  
   for tab 67  
 logical tab character  
   establishing default 67  
 LOGMSG\_FROM\_FILE operand  
   disabling on FEATURES statement 141  
   enabling on FEATURES statement 142  
 logo configuration file  
   contents 28  
   description 279  
   limit and audit access 380  
   overview 28  
   specifying file name and file type 164  
   statements 279  
     CHOOSE\_LOGO 285  
     INPUT\_AREA 291  
     ONLINE\_MESSAGE 292  
     STATUS 293  
     summary of statements 279  
 logo picture file  
   choosing 285  
   selection 288  
   separator pages 289, 290  
   width 288  
 logo screens, creating 282  
 LOGO\_CONFIG statement 53, 164  
 LOGON command 299, 326  
   journaling 385  
   prompting for password on 144  
 LOGON operand  
   JOURNALING statement 162  
 LOGONBY user directory control statement 509

## M

- MAC operand
  - DEFINE CMD statement 86
  - DEFINE COMMAND statement 86
  - DEFINE DIAGNOSE statement 91
- machine check
  - I/O
    - initializing devices after 136
- MACHINE user directory control statement 510
- macroinstruction
  - CP
    - HCPDROP 311, 374
    - HCPENTER 311, 374
    - HCPEPILG 311, 374
    - HCPEXIT 311, 374
    - HCPPROLG 311, 374
    - HCPUSING 311
  - CSELDEV 646
  - CSELVOL EXCLUDE 338, 359, 649
  - CSELVOL INCLUDE 338, 359, 650
  - CSESYS 653
  - CSETRACE 658
  - CSEUSER 371, 659
  - DTIGEN 330
  - FCB 418, 420
  - HCPSYS 643
  - RDEVICE 706
  - RIOGEN 766
  - SYSACNT 6, 661
  - SYSADDIN 663
  - SYSCPVOL 337, 447, 620, 664
  - SYSDUMP 6, 667
  - SYSEND 668
  - YSEREP 6, 669
  - SYSEXCL 671
  - SYSFCN 575, 673
  - SYSFORM 675
  - SYSID 677
  - SYSINCL 680
  - SYSJRL 682
  - SYSMAXU 686
  - SYSOPR 6, 687
  - SYSOPTS 689
  - SYSPCLAS 691
  - SYSRES 391, 605
  - SYSSTORE 697
  - SYSSYMP 699
  - SYSTIME 701
  - SYSUVOL 8, 620, 703
  - UCSI 415
  - UCSICW 415
- main storage, allocation of 397
- MAINT virtual machine 32
- MAKEBUF command 399
- map record
  - and number of sharing VM systems 359
  - defaults 652
  - displaying indicators 363
  - length 652
  - number 652
  - on CSE area 363
- MAP\_RECORD\_LENGTH operand
  - XLINK\_VOLUME\_INCLUDE statement 268
- MAP\_RECORDS operand
  - XLINK\_VOLUME\_INCLUDE statement 268
- mapping
  - expanded storage 593
- maximum
  - number of users, specifying 136
- maximum systems supported
  - for cross system link 334, 338, 656
- maximum user limit
  - overriding 535
- MAXSTORAGE user directory control statement 512
- MAXUSERS operand
  - FEATURES statement 144
- MCCU (Multisystem Channel Communications Unit)
  - requirement for CSE 335
- MDC (minidisk caching feature)
  - in CSE 339
- MDISK user directory control statement 9, 339, 513, 625
- member
  - deleting from an image library 424
  - replacing in an image library 424
- MEMBER operand
  - CPXLOAD statement 77
- message
  - commands, CP, altering output 373
  - log on
    - specifying whether CP should display 136
    - online, changing 292
- MESSAGE and QUERY commands
  - in CSE 334
- MESSAGE command, altering output 373
- MESSAGE operand
  - JOURNALING statement 162
- migration
  - incompatibility note 656, 678
  - planning 3
- minidisk
  - access modes 515
  - accessing
    - by CP 70
  - as units of data control 358
  - available for caching 600
  - cache performance considerations 602
  - cache using expanded storage as a 597
  - cache, special considerations 601
  - caching 600
  - clearing 398
  - CP-accessed 33, 34
  - DASDs used for 8
  - defining for a virtual machine 513
  - defining full-pack minidisk 630
  - defining links to 500
  - full-pack 630
  - full-pack, defining 625
  - full-pack, restricting and auditing 382
  - full-pack, using to share DASD 625
  - initializing CSE area on 349
  - non-full-pack 630

minidisk (*continued*)  
   offset field on SAPL screen 41  
   parm disk 51  
   restrictions 521  
   shared CMS expanded storage considerations 592  
   used to share data 619  
   using to share DASD between virtual machines 619  
   using with cached DASD 629  
   write-through 597  
 minidisk cache enhancements 598  
 minimum content of a system configuration file 25  
 MINIMUM operand  
   CHOOSE\_LOGO statement 287  
 MINIOPT user directory control statement 524  
 MINIPROMPT 396  
 mode switch procedure for 3390 devices 616  
 MODEL operand  
   CHOOSE\_LOGO statement 286  
   RDEVICE statement 198, 213  
 modify  
   image library 411  
   read-only saved segments 390  
 MODIFY DIAGNOSE statement 53, 169  
 MODIFY EXIT statement 53, 172  
 MODIFY LAN statement 53, 175  
 MODIFY PRIV\_CLASSES Statement 53, 176  
 MODIFY VSWITCH Statement 53, 178  
 modifying virtual LAN segment 175  
 modifying virtual switch segment 178  
 module  
   name field on SAPL screen 41  
   replacing in an image library 424  
   XLINK 349  
 MORE operand  
   STATUS statement 293  
 MP operand  
   CPXLOAD statement 77  
 MPLF (Multi-Path Lock Facility)  
   real  
     OPTION user directory control statement 535,  
     627  
     sharing DASD 627  
   RPQ  
     OPTION user directory control statement 535,  
     627  
     sharing DASD 627  
 MSG command 276  
 MSGNOH command 276  
   altering output 373  
 multisystem complex, defining statements that  
   apply 569

## N

name groups of systems 133  
 NAMESAVE user directory control statement 527  
 naming convention  
   FCB 416  
   UCS buffer image 416  
 NCP (Network Control Program)  
   used with 3720 49

NCP (Network Control Program) (*continued*)  
   used with 3745 49  
 networking implications of CSE 357  
 new  
   alias  
     creating for existing CP command 81  
   commands  
     creating CP 84  
   DIAGNOSE codes  
     creating 90  
   exit  
     creating 94  
   version  
     creating for existing CP command 84  
 NICDEF user directory control statement 529  
 NO\_SPOOLING operand  
   RDEVICE statement 190, 192, 202, 213  
 NOAUTOLOG operand  
   SYSTEM\_USERIDS statement 233  
 NOCONTROL operand  
   CPXLOAD statement 78  
 NODELAY operand  
   CPXLOAD statement 78  
 NODISCONNECT operand  
   SYSTEM\_USERIDS statement 233  
 NOLET operand  
   CPXLOAD statement 78  
 NOLIMIT operand  
   FEATURES statement 144  
 NOLOCK operand  
   CPXLOAD statement 77  
 NOMP operand  
   CPXLOAD statement 77  
 non-disruptive transition support 392  
 non-full-pack minidisk 630  
 NONMP operand  
   CPXLOAD statement 77  
 NOPDATA user directory control statement 532  
 NOPROMPT 397  
 NOREPLACE 397  
 NOTACCEPTED operand  
   DEVICES statement 106  
   STATUS statement 293  
 notational convention  
   for commands xvi  
 notices 781  
 NOTSILENT operand  
   MODIFY CMD statement 167  
   MODIFY COMMAND statement 167  
 NSS (named saved system)  
   allocating DASD space for 615  
   restricted, authorizing virtual machines to use 527  
 number of  
   retrieve buffers, specifying maximum 136  
   users, specifying maximum 136

## O

OFF operand  
   D8ONECMD statement 483  
   HOT\_IO\_RATE statement 153

- online
  - bringing specified devices 105
  - messages 279
    - changing 292
- ONLINE\_MESSAGE statement 292
- OpenExtensions planning 13
- OPER\_IDENT\_READER
  - RDEVICE statement 198
- operating modes for 3390 devices, switching 615
- operator
  - console
    - defining 766
    - defining list of possible 180
  - defining directory entry 342
  - form numbers, creating list of 247
  - identification card for displays 49
  - primary system, specifying a user ID for 687
  - system
    - specifying disconnect status 232
    - specifying user ID 232
- OPERATOR operand
  - PRIV\_CLASSES statement 185
  - SYSTEM\_USERIDS statement 233
- OPERATOR\_CONSOLES statement 53, 180
- OPERSYMP virtual machine
  - sample directory entry 318
  - SYSSYMP macroinstruction 699
  - system configuration file
    - SYSTEM\_USERIDS statement 234
- option
  - for CP, specifying initialization 689
  - specifying IPL parameters 42
- OPTION user directory control statement 533
- ORDER command 368
- order of statements in a system configuration file 57
- organization review of CP-owned DASD 337
- original spool descriptor 368
- OSA operand
  - RDEVICE statement 205
- outages 372
- output
  - choosing logo for separator pages 285
  - devices for stand-alone dump utility 403, 406
- OUTPUT operand
  - TRANSLATE\_TABLE statement 244
- override
  - CP command 165
- OVERRIDE control statement 438
- override file 433, 438
- OVERRIDE utility 382, 441
- overriding console that CP will use 39
- overriding SAPL defaults 39
- overview
  - planning and administration tasks 3
- OWN operand
  - CP\_OWNED statement 73

**P**

- page allocation
  - for XSPPOOL trace tables 273
- PAGE operand
  - DRAIN statement 119
  - START statement 220
- page protection 390
- PAGES operand
  - XSPPOOL\_TRACE statement 273
- pages, lock and unlock 394, 397
- paging 398
  - space 338, 610
- Parallel Access Volumes 631
- parameter list
  - AOFPARAM 314
  - AONPARAM 312
  - HCPACUOF entry point 314
  - HCPACUON entry point 312
  - HCPMSUEX entry point 374
- parm disk 51
  - limit and audit access 380
  - number 43
  - offset 43
  - placing information on 23
  - purpose 23
  - real address 43
- password
  - automatic deactivation of restricted 386
  - defining for virtual machine user 572
  - directory
    - authorization bypass 385
    - suppression facility, including 682
    - suppression facility, including in system 136
- PASSWORDS\_ON\_CMDS operand
  - FEATURES statement 144
- pending retained Expanded Storage 597
- PEP (partitioned emulation program)
  - used with 3720 49
- performance
  - administration tasks 12
  - allocating directory space 608
  - CP file system 33
  - DASD considerations for sharing spool files 337
  - DASD for dynamic paging area 590
  - GCS 332
  - planning 12
  - SET D8ONECMD command 381
  - SNA/CCS 332
- PERM space
  - for CSE area 340
- PERMANENT operand
  - CPXLOAD statement 78
- picture file, logo
  - choosing 285
  - selection 288
  - separator pages 289, 290
  - width 288
- planning
  - considerations 3
  - CSE installation 334
  - migration from a previous VM release 3
  - overview 5
  - performance tasks 12
  - real and virtual storage 7



planning (*continued*)  
   SNA/CCS 329  
   system 4  
   tasks 4  
   user class structure 433  
   user tasks 5  
 POOL user directory control statement 540  
 POSIX  
   directory definitions 448  
   POSIXGLIST directory control statement 541  
   POSIXGROUP directory control statement 543  
   POSIXINFO directory control statement 545  
   POSIXOPT directory control statement 548  
   user planning and administration 6  
 POSIX groups  
   continuation comma 452  
   continued directory control statements 452  
   directory definition order 448  
   Global definitions 448  
   LOAD user directory control statement 504  
   POSIXGLIST 541  
   POSIXGROUP 543  
   POSIXINFO 545  
   POSIXOPT 548  
   quoted string operands 452  
 POSIXGLIST user directory control statement 541  
 POSIXGROUP user directory control statement 543  
 POSIXINFO user directory control statement 545  
 POSIXOPT user directory control statement 548  
 PRECEDING operand  
   ASSOCIATE EXIT statement 61  
   ASSOCIATE MESSAGES statement 65  
   ASSOCIATE MSGS statement 65  
 preparation package 335  
 prepare  
   an override file 438  
   DASD volumes for CSE 361  
   for CSE 341  
 prevent  
   new operations on real DASD 118  
   systems from accessing other system's  
   minidisks 263  
   use of X'15' 483  
   users issuing multiple CP commands 483  
 primary system  
   console  
     defining on OPERATOR\_CONSOLES  
     statement 180  
     defining on RIOGEN macro 766  
   operator  
     specifying a user ID for 687  
 print buffer image  
   adding 414  
   IBM provided 412  
 printer  
   3800  
     defining 189, 212  
   3820  
     defining 189  
   3825  
     defining 189  
   printer (*continued*)  
     3827  
       defining 189  
   3828  
     defining 189  
   3835  
     defining 189  
   3900  
     defining 189, 212  
   advanced function  
     defining 189  
   coding an RDEVICE macro  
     3203 716  
     3211 716  
     3262 716  
     3800 721  
     3820 710  
     4245 716  
     4248 716  
     6262 716  
     advanced function 708  
     common control unit 708  
   configuration guide 47, 769  
   dedicating to virtual machine 477  
   defining in system configuration file 47  
   file  
     not accessible to other systems 659  
   form printer default 675  
   format, stand-alone dump 780  
   image library 411  
   impact  
     defining 201  
     output classification titles 691  
     output separator pages 289, 290  
     sharing with other users 477  
     specifying default form values 675  
     spooled, defining for virtual machine 562  
   unsupported  
     defining 209  
   PRINTER operand  
     FORM\_DEFAULT statement 150  
     RDEVICE statement 210  
   PRINTER\_TITLE statement 53, 182  
   PRIV\_CLASSES statement 53, 184  
   PRIVCLASSANY operand  
     DEFINE CMD statement 86  
     DEFINE COMMAND statement 86  
     DEFINE DIAGNOSE statement 91  
     MODIFY CMD statement 166  
     MODIFY COMMAND statement 166  
     MODIFY DIAGNOSE statement 170  
   PRIVCLASSES operand  
     DEFINE CMD statement 86  
     DEFINE COMMAND statement 86  
     DEFINE DIAGNOSE statement 91  
     MODIFY CMD statement 166  
     MODIFY COMMAND statement 166  
     MODIFY DIAGNOSE statement 170  
   privilege classes  
     accounting record 306, 307, 309  
     assigning to commands and DIAGNOSE codes 438

- privilege classes (*continued*)
  - authorizing CP functions, changing 184
  - changing 433
  - changing CP function of 673
  - changing individual operands of QUERY and SET commands 440
  - controlling which users can change 136
  - defining for a virtual machine 444
  - definition of 392
  - IBM-defined 433
  - overriding 392, 396
  - tailoring 392, 396
- problems, reporting 400
- PROC operand
  - DEFINE CMD statement 86
  - DEFINE COMMAND statement 86
  - DEFINE DIAGNOSE statement 91
- processor
  - defining a system identifier for 677
  - defining virtual 465
- production environment
  - phasing CSE into 356
- profile
  - directory 455
  - entry, defining start of in source directory 551
  - invoked as part of USER statement 487
  - PVM modifications for CSE 350
- PROFILE user directory control statement 551
- Programmable Operator Facility
  - security 382
- programming interface information 782
- prompt at start-up, requesting 43
- proprietary data protection 391
- PROT operand
  - DEFINE CMD statement 86
  - DEFINE COMMAND statement 86
  - DEFINE DIAGNOSE statement 91
- protection keys 397
- protocols, link 358
- PSF/VM (Print Services Facility/VM)
  - setting up virtual machines 326
- PTF (Program Temporary Fix) 401
- PUN operand
  - RDEVICE statement 190
- punch
  - unsupported
    - defining 209
- punch file
  - not accessible to other systems 659
- PUNCH operand
  - FORM\_DEFAULT statement 150
  - RDEVICE statement 210
- PURGE command 369
- PURGE IMG command 425
- PURGE UCR command 442
- PVM (VM/Pass-Through Facility)
  - directories 395
  - functions for cross system spool 371
  - need for CSE 335
  - networks in CSE 357

- PVM (VM/Pass-Through Facility) (*continued*)
  - PVM CONFIG
    - modifications for CSE 350
    - run by the CVM 345

## Q

- qualifiers in system configuration files 56
- query
  - journaling facility, enabling system to 160
- QUERY (User ID) command 277
- QUERY and MESSAGE command
  - in CSE 334
- QUERY CACHE command 747
- QUERY COMMAND command 446
- QUERY command operand
  - changing privilege classes of 440
- QUERY DASDFW command 748
- QUERY FILES command 369
- QUERY FRAMES command 589
- QUERY IMG command 424
- QUERY LOGMSG command 143
- QUERY NAMES command 277
- QUERY operand
  - DEFINE ALIAS statement 81
  - DEFINE CMD statement 84
  - DEFINE COMMAND statement 84
  - DISABLE CMD statement 110
  - DISABLE COMMAND statement 110
  - ENABLE CMD statement 126
  - ENABLE COMMAND statement 126
  - MODIFY CMD statement 165
  - MODIFY COMMAND statement 165
- QUERY PINNED command 747
- QUERY READER/PRINTER/PUNCH 369
- QUERY RETRIEVE command 146
- QUERY RSAW command 747
- QUERY UCR command 442
- QUERY UCR COUNT command 442
- QUERY XSTORE command 594
- QUERY XSTORE MAP command 592, 593
- Queued-I/O assist, control by DEDICATE statement 478
- quoted string operands
  - quoted string 452
  - rules 452
  - with control statements 452

## R

- RACF/VM (Resource Access Control Facility/VM)
  - dual registration 381
  - general description 379
- rate, hot I/O
  - changing 152
  - setting 152
- RDEVICE macro
  - general format 706
  - how to code
    - 2701 data adapter 752
    - 2741 terminal 754

RDEVICE macro *(continued)*  
 how to code *(continued)*  
 3088 multisystem channel communication unit 760  
 3101 display 734  
 3151 display terminal 754  
 3161 display terminal 754  
 3162 display terminal 754  
 3163 display terminal 754  
 3167 display terminal 754  
 3174 controller 757  
 3178 display 734  
 3179 display 734  
 3180 display 734  
 3191 display 734  
 3192 display 734  
 3193 display 734  
 3194 display 734  
 3203 printer 716  
 3211 printer 716  
 3250 display 733, 734  
 3262 display printer 713  
 3262 printer 716  
 3268 display printer 713  
 3270-PC 734  
 3274 controller 757  
 3277 display 734  
 3278 display 734  
 3279 display 734  
 3286 display printer 713  
 3287 display printer 713  
 3288 display printer 713  
 3289 display printer 713  
 3290 display 734  
 3350 DASD 746  
 3350 DASD (in 3330 compatibility mode) 746  
 3370 749  
 3375 DASD 746  
 3380 DASD 746  
 3390 DASD 746  
 3422 tape drive 743  
 3430 tape drive 743  
 3480 tape drive 743  
 3490 tape drive 743  
 3505 card reader 727  
 3525 card punch 729  
 3590 tape drive 743  
 3705 communication controller 757  
 3705 line adapter 754  
 3720 communication controller 757  
 3725 communication controller 757  
 3725 line adapter 754  
 3737 Remote Channel-to-Channel Unit 760  
 3745 communication controller 757  
 3745 line adapter 754  
 3800 printer 721  
 3812 printer 713  
 3816 printer 713  
 3820 710  
 3850 DASD 746  
 3890 document processor 731

RDEVICE macro *(continued)*  
 how to code *(continued)*  
 4214 display printer 713  
 4224 display printer 713  
 4234 display printer 713  
 4245 display printer 713  
 4245 printer 716  
 4248 printer 716  
 5080 graphics display 740  
 5210 display printer 713  
 5550 display 734  
 6090 graphics display 740  
 6262 display printer 713  
 6262 printer 716  
 7171 Device Attachment Control Unit 734  
 9332 749  
 9335 749  
 9336 749  
 9345 DASD 746  
 advanced function printer 708  
 alternate system console 766  
 common control unit printer 708  
 CTCA 760  
 dynamic switching devices 762  
 FB-512 749  
 FBA DASD 749  
 primary system console 766  
 teleprocessing integrated adapter 738  
 unsupported devices 763  
 RDEVICE statement 53, 188  
 3800 printers 212  
 3900 printers 212  
 advanced function printers 189  
 card punch 190  
 card readers 192  
 channel-to-channel adapters 205  
 communication controllers 193  
 DASD 195  
 graphic display devices 197  
 impact printers 201  
 LAN adapters 205  
 line adapters 193  
 special devices 205  
 tape units 207  
 unsupported devices 209  
 READCARD command 396  
 reader  
 defining 192  
 unsupported  
 defining 209  
 reader file  
 composite 396  
 not accessible to other systems 659  
 READER operand  
 FORM\_DEFAULT statement 150  
 RDEVICE statement 192, 210  
 real  
 channel program support 397  
 device set, adding to 188  
 reserve/release  
 when to use 624

- real storage
  - administration 7, 589
  - configuration, defining 697
  - configuring 222, 697
  - planning 7, 589
  - requirements, CP module 589
- RECEIVE command 396
- RECL operand
  - CSELVOL INCLUDE operand 651
- record qualifiers in system configuration files 56
- RECORDING command
  - EREP operand of 316
- RECS operand
  - CSELVOL INCLUDE operand 651
- redefine
  - command privilege classes 433
  - CP command 165
  - text of status fields 293
- Remote Spooling Communications Subsystem Networking (RSCS)
  - directory options 395
  - in CSE 357
- replace
  - members in an image library 424
  - modules in an image library 424
  - translate tables 243
- REPLACE operand
  - ASSOCIATE EXIT statement 60
  - ASSOCIATE MESSAGES statement 64
  - ASSOCIATE MSGS statement 64
- reporting z/VM integrity problems 400
- request
  - sense ID
    - using information from 105
- requirement
  - system
    - for CSE 334
- reserve release 210
- RESERVE\_RELEASE operand
  - RDEVICE statement 210
- reserve/release
  - concurrent virtual and real, when to use 626
  - description 619
  - real, when to use 624
  - restrictions 626
  - type
    - concurrent virtual and real 625
    - real 622
    - virtual 619
  - virtual
    - DASD sharing without using 622
    - when to use 621
- RESERVED operand
  - CP\_OWNED statement 73
  - XSPOOL\_SYSTEM statement 271
- RESET operand
  - MODIFY CMD statement 166
  - MODIFY COMMAND statement 166
  - MODIFY DIAGNOSE statement 169
- residence disk, CP 693
- resident pages dumped with stand-alone dump utility 403
- resource
  - managing 495
- response suppression, command 86, 166
- restart
  - real DASD 220
- restrict
  - users issuing multiple CP commands 483
- restricted password
  - automatic deactivation of 386
- restriction
  - 3525 card punch 190
  - 3705, 3720, 3725, and 3745 BSC line adapters 193
  - 3850 Mass Storage System 195
  - about migration incompatibilities 656, 678
  - against using 2250 display 124
  - against using 3250 display 124
  - defining advanced function printers 189
  - DIAGNOSE codes 394
  - minidisk 521
  - removed
    - issuing multiple CP commands 483
    - specifying real storage available for CP 697
  - reserve/release 626
  - turning off hot I/O rate 153
  - users migrating from VM/XA SP 451, 454
  - VM/ESA commands 393
- RETAIN XSTORE command 591, 593, 596
- retained Expanded Storage 596, 597
- retrieve buffer
  - specifying
    - default number of 136
    - maximum number of 136
- RETRIEVE command (CP)
  - retrieving EREP records 316
- RETRIEVE operand
  - FEATURES statement 145
- RIOGEN macro 766
- RMSIZE operand
  - SYSSTORE macro 697
- RUNNING operand
  - STATUS statement 294
- running programs
  - utilities 37

## S

- SAMPEXEC files 641
- sample utility program
  - DRAWLOGO 282, 642
- SAMPXEDI files 641
- save area, HCPSAVBK
  - contents 312
  - used by CP accounting exit 312
  - used by CP message function exit 374
- saved segment
  - modifying read-only 390
  - nonrestricted, authorizing virtual machines to use 527
  - restricted, authorizing virtual machines to use 527

SAY statement 53, 217, 219

SCIF (Single Console Image Facility)  
 restricting use of 381

screen  
 logo  
 choosing picture file 285

SCREEN user directory control statement 553

SCSI disks 635

security  
 considerations 379, 380  
 external security manager 383  
 facilities 379, 383  
 guidelines 380  
 Integrated Cryptographic Facility 383  
 maintaining system integrity 389, 401  
 of z/VM installation 142  
 products that enhance 379  
 security journaling 160  
 system 234  
 use badge reader to log on 198  
 using an Integrated Cryptographic Facility 387

semantic statements in system configuration file 27

sense ID request  
 specifying whether CP should issue 689  
 using information from 105

SENSE operand  
 SYSOPTS macro 689

sensing devices at IPL time 46

SEPARATOR operand  
 RDEVICE statement 203, 215

separator page  
 logo  
 choosing 285

serialization 392

service  
 pool  
 definition 325  
 manager 326  
 virtual machines, setting up 325  
 virtual machine  
 defining directory entry 342  
 excluding from spooling 659  
 setting up 295

service, system  
 \*ACCOUNT 233  
 \*BLOCKIO 494  
 \*CONFIG 494  
 \*CRM 494  
 \*IDENT 496  
 \*LOGREC 314, 494, 669  
 \*MONITOR 494  
 \*MSG 494  
 \*MSGALL 494  
 \*RPI 494  
 \*SIGNAL 494  
 \*SYMPTOM 318, 494  
 \*SYMPTOM, specifying virtual machines for 699  
 accounting 295, 661  
 collecting accounting records 297  
 collecting EREP records 315  
 collecting symptom records 319

service, system (*continued*)  
 communication requirements 497  
 CP recording 295  
 symptom record recording 295

session  
 concurrent copy 566  
 multiple concurrent 371

set  
 hot I/O rate 152  
 I/O throttling rate 105  
 journaling facility, enabling system to 160  
 time of (TOD) clock 701

SET CACHE command 747

SET command operand  
 changing privilege classes of 440

SET CONCEAL OFF command 534

SET D8ONECMD command 381, 393

SET DASDFW command 747

SET DUMP command 614

SET LOGMSG command 141

SET MAXUSERS command 535

SET MDCACHE SYSTEM 393

SET OBSERVER command 393

SET operand  
 DEFINE ALIAS statement 82  
 DEFINE CMD statement 84  
 DEFINE COMMAND statement 84  
 DISABLE CMD statement 110  
 DISABLE COMMAND statement 110  
 ENABLE CMD statement 126  
 ENABLE COMMAND statement 126  
 MODIFY CMD statement 166  
 MODIFY COMMAND statement 166

SET PFnn RETRIEVE command 145

SET PRIVCLASS command 393  
 controlling which users can use 136

SET RDEVICE command 393

SET RETRIEVE command 146

SET SECUSER command 393

SET SHARED command 601, 620

SET SHARED OFF command 620

SET SHARED ON command 626

SET SRM command 332, 537

SET statement 53

SET SYSOPER command 393

SET TIMEZONE command 394

set up  
 a CSE complex 334  
 data storage management virtual machines 327  
 Print Services Facility/VM virtual machines 326  
 service pool virtual machine 325

SET VDISK SYSLIM command 394

SET\_ADDRESS operand  
 RDEVICE statement 193

SET\_AND\_QUERY operand  
 JOURNALING statement 161

SET\_PRIVCLASS operand  
 disabling on FEATURES statement 141  
 enabling on FEATURES statement 144

SHARE user directory control statement 555

SHARE\_SPOOL operand  
   XSPPOOL\_SYSTEM statement 271  
 shared  
   access to minidisks, DASD for 338  
   CMS minidisks expanded storage considerations 592  
   DASD 365  
     defining 195  
     omitting devices 105  
     specifying devices 105  
   data  
     reserve/release 210  
     file pools, DASD space used for 9  
     segment 391, 398  
     spool files, DASD for 337  
 Shared File System (SFS)  
   sharing DASD 622  
 SHARED operand  
   CP\_OWNED statement 74  
   RDEVICE statement 195  
 sharing  
   CPU power 555  
   DASD 619, 622  
     using Parallel Access Volumes 632  
   volumes 336  
 SHOW\_ACCOUNT operand  
   FEATURES statement 143  
 SHUTDOWN command 299, 394  
 SILENT operand  
   DEFINE CMD statement 86  
   DEFINE COMMAND statement 86  
   MODIFY CMD statement 166  
   MODIFY COMMAND statement 166  
 single source directory 341  
 SLOT operand  
   XLINK\_SYSTEM\_INCLUDE 264  
   XSPPOOL\_SYSTEM 271  
 SMSG command 276  
   altering output 373  
   to start CSECOM task 353  
 SNA (Systems Network Architecture)  
   enabling communication 331  
   environment 329  
   networks in CSE 358  
 SNA/CCS (System Network Architecture/Console Communication Services)  
   accounting record 304  
   defining to VSCS 331  
   defining VCNA to 330  
   functions 329  
   improving performance of 332  
   planning 329  
   terminal environment, establishing 330  
   terminals  
     choosing logo picture file for 285  
 SNAPDUMP command 394  
 source code, adding your own 310  
 space  
   allocating  
     for internal trace tables 222, 697  
     for XSPPOOL trace tables 273  
   space (*continued*)  
     clearing TDISK 136  
     storage  
       configuring 222, 697  
       used for shared file pools 9  
   special devices  
     defining 205  
   SPECIAL operand  
     HOT\_IO\_RATE statement 152  
   SPECIAL user directory control statement 557  
   specify  
     access list size, host 581  
     CP's displaying of log messages 136  
     DASD excluded from cross system link 266, 649  
     DASD to be included in cross system link 268  
     DASD volumes included in cross system link 650  
     default number of RETRIEVE buffers 136  
     file for input area 291  
     file name and type of logo configuration file 164  
     forms as NARROW 247  
     host access list size 581  
     inclusion of journaling facility 160  
     inclusion of password suppression facility 136  
     initialization options for CP 689  
     maximum number of nonprimary address spaces 583  
     maximum number of RETRIEVE buffers 136  
     maximum number of users 136  
     replacements for standard translate tables 243  
     size of host access list 581  
     spool file class for impact printers 202  
     systems excluded from cross system link 263  
     systems in cross system spooling 271  
     systems participating in cross system commands 271  
     systems to be included in cross system link 264  
     total size of nonprimary address spaces 583  
     user IDs to perform special functions 232  
     volumes excluded from user volume list 251, 671  
     volumes included in user volume list 680  
     volumes to be included in user volume list 253  
     whether CP builds device blocks 105  
     whether CP uses use sense ID information 105  
   spool  
     activate CVM 355  
     cross system 366  
     descriptor 368  
     enabling 353  
     file  
       access to 370  
       characteristics 565  
       maximum number allowed 565  
       transfer data to 532  
     files, DASD for 337  
     synchronization request tracing 371  
     verifying 354  
   SPOOL command 665  
   spool file  
     classes  
       containing classification titles 182  
       specifying for impact printers 202

- spool file *(continued)*
  - input
    - copying (reader) 369
    - excluding from cross system spooling 274
  - numbering
    - numbering 370
  - output
    - classes containing classification titles 182
    - copying (print and punch) 370
    - excluding from cross system spooling 276
- SPOOL operand
  - START statement 220
- SPOOL user directory control statement 562
- spooled unit record devices 11
- SPOOLFILE user directory control statement 565
  - in CSE 341
- spooling space
  - example 615
- spooling, cross system
  - excluding virtual machines' input spool files from 274
  - excluding virtual machines' output spool files from 276
  - identifying volumes for 73
  - specifying systems participating in 271
  - trace tables
    - allocating space for 273
- SPXTAPE command 665
- stable access mode 327, 387
- stable access mode, authorized to use 536
- StackBufferCreate 399
- StackBufferDelete 399
- stacks, program 398
- stand-alone dump utility
  - creating 403
  - description of 403
  - devices to IPL 405
  - devices to send dump output 406
  - example of configuring 406
  - HCPSADMP EXEC 409
  - output devices 406
  - overview 403
  - processing dump data on tape 409
  - taking a dump 408
- Stand-Alone Loader Creation Utility (SALIPL)
  - using 37
- Stand-Alone Program Loader (SAPL)
  - creating 37
  - file list 42
  - overriding console that CP will use 39
  - overriding defaults 39
  - overview 37
  - passing IPL parameters 42
- start
  - prompt 31, 43
  - real DASD 220
- START CSECOM command 353
- START statement 53, 220
- starting CP extensions at initialization time 663
- starting VSM 332

- statement
  - ALTERNATE\_OPERATORS 58
  - ASSOCIATE
    - EXIT 60
    - MESSAGES 64
    - MSGS 64
  - ASSOCIATE EXIT 51
  - ASSOCIATE MESSAGES/MSGS 51
  - CHARACTER\_DEFAULTS 51, 67
  - CHARACTER\_OPERATORS 51
  - CHOOSE\_LOGO 279, 285
- configuration
  - logo 279
  - system 51
- CP\_ACCESS 52, 70
- CP\_ADDON\_INITIALIZE\_ROUTINES 52
- CP\_ADDON\_INITIALIZE\_ROUTINES 72
- CP\_OWNED 52, 73
- CPXLOAD 52, 76
- DEFINE
  - CMD 84
  - COMMAND 84
  - DEFINE ALIAS 81
  - DEFINE EXIT 94
  - DIAGNOSE 90
- DEFINE ALIAS 52
- DEFINE COMMAND/CMD 52
- DEFINE DIAGNOSE 52
- DEFINE EXIT 52
- DEFINE LAN 52
- DEFINE VSWITCH 52
- DEVICES 52, 105
- DISABLE
  - CMD 110
  - COMMAND 110
  - DIAGNOSE 112
  - EXITS 114
- DISABLE COMMAND/CMD 52
- DISABLE DIAGNOSE 52
- DISABLE EXITS 52
- DISTRIBUTE 52, 116
- DRAIN 52, 118
- EDEVICE 52, 121
- EMERGENCY\_MESSAGE\_CONSOLES 52, 124
- ENABLE
  - CMD 126
  - COMMAND 126
  - DIAGNOSE 128
  - EXITS 130
- ENABLE COMMAND/CMD 52
- ENABLE DIAGNOSE 52
- ENABLE EXITS 52
- ENFORCE\_BY\_VOLIDS 52
- EQUATE 52, 133
- EXTERNAL\_SYNTAX 53, 135
- FEATURES 53, 136
- FORM\_DEFAULT 53, 150
- HOT\_IO\_RATE 53, 152
- IMBED 53, 155
- INIT\_MITIME 53, 157
- INPUT\_AREA 279, 291

statement (*continued*)

- IODF 53, 158
- JOURNALING 53, 160
- LOGO\_CONFIG 53, 164
- MODIFY
  - DIAGNOSE 169
  - MODIFY EXIT 172
- MODIFY CMD 165
- MODIFY COMMAND 165
- MODIFY COMMAND/CMD 53
- MODIFY DIAGNOSE 53
- MODIFY EXIT 53
- MODIFY LAN 53
- MODIFY PRIV\_CLASSES 53, 176
- MODIFY VSWITCH 53
- ONLINE\_MESSAGE 279, 292
- OPERATOR\_CONSOLES 53, 180
- PRINTER\_TITLE 53, 182
- PRIV\_CLASSES 53, 184
- PRODUCT 53
- RDEVICE 53, 188
  - 3800 printers 212
  - 3900 printers 212
  - advanced function printers 189
  - card punch 190
  - card readers 192
  - channel-to-channel adapters 205
  - communication controllers 193
  - DASD 195
  - graphic display devices 197
  - impact printers 201
  - LAN adapters 205
  - line adapters 193
  - special devices 205
  - tape units 207
  - unsupported devices 209
- SAY 53, 217
- SET 53, 219
- START 53, 220
- STATUS 279, 293
- STORAGE 53, 222
- summary 279
- summary of system configuration file 51
- SYSTEM\_DATEFORMAT 53, 225
- SYSTEM\_IDENTIFIER 54, 226
- SYSTEM\_IDENTIFIER\_DEFAULT 228
- SYSTEM\_IDENTIFIER\_DEFAULT 54
- SYSTEM\_RESIDENCE 54, 230
- SYSTEM\_USERIDS 54, 232
- THROTTLE 54, 236
- TIMEZONE\_BOUNDARY 54, 237
- TIMEZONE\_DEFINITION 54, 239
- TOLERATE\_CONFIG\_ERRORS 241
- TOLERATE\_CONFIG\_ERRORS 54
- TRANSLATE\_TABLE 54, 243
- USER\_DEFAULTS 54, 248
- USER\_VOLUME\_EXCLUDE 54, 251
- USER\_VOLUME\_INCLUDE 54, 253
- USER\_VOLUME\_LIST 54, 255
- USERFORM 54, 247
- VMLAN 54

statement (*continued*)

- XLINK\_DEVICE\_DEFAULTS 54, 259
- XLINK\_SYSTEM\_EXCLUDE 54, 263
- XLINK\_SYSTEM\_INCLUDE 54, 264
- XLINK\_VOLUME\_EXCLUDE 54, 266
- XLINK\_VOLUME\_INCLUDE 54, 268
- XSPPOOL\_SYSTEM 54, 271
- XSPPOOL\_TRACE 54, 273
- XSPPOOL\_XLIST\_INPUT 54, 274
- XSPPOOL\_XLIST\_OUTPUT 55, 276
- status
  - fields on screen, changing text 293
- STATUS statement 279, 293
- STDEVOPT user directory control statement 566
- STIDP instruction 465
- stop
  - new operations on real DASD 118
  - use of X'15' 483
  - users issuing multiple CP commands 483
- storage
  - access verification 384
  - administration 7
  - allocation 397
    - for XSPPOOL trace tables 273
  - calculations 604
  - capacities, DASD 604
  - configuration, defining 697
  - DASD 603
  - dynamic paging area 590
  - expanded 591
  - expanded, planning 9
  - free, used by CP 538
  - organization
    - configuring 222, 697
  - planning 7
  - protection
    - main storage 397
    - paging 398
    - shared segments 398
  - specifying amount of real storage used by CP 44
  - task overview 7
  - using expanded storage as a minidisk cache 597
  - virtual defining for virtual machine user 572
  - virtual machine considerations 590
  - virtual storage, planning 9
- STORAGE statement 53, 222
- STORAGE user directory control statement 568
- STORE HOST command 394
- SUBCMD operand
  - DEFINE ALIAS statement 81
  - DEFINE CMD statement 84
  - DEFINE COMMAND statement 84
  - DISABLE CMD statement 110
  - DISABLE COMMAND statement 110
  - ENABLE CMD statement 126
  - ENABLE COMMAND statement 126
  - MODIFY CMD statement 165
  - MODIFY COMMAND statement 165
- summary of
  - logo configuration file statements 279
  - system configuration file statements 51



- supported devices 45
- suppress
  - password on command line 136, 387
- SWITCH operand
  - HOT\_IO\_RATE statement 153
  - RDEVICE statement 210
- switching devices, dynamic
  - coding an RDEVICE macro 762
  - defining 209
- switching operating modes for 3390 devices 615
- symbol
  - character delete
    - establishing default 67
  - escape
    - establishing default 67
  - line delete
    - establishing default 67
  - line end
    - establishing default 67
  - tab
    - establishing default 67
- symbol, external
  - assigning to exit point 60, 64
- symbols, defining for virtual machine 572
- symptom record
  - specifying user ID to accumulate 232
- symptom record recording
  - disassociating a user ID from retrieval of 320
  - OPERSYMP symptom recording virtual machine 318
  - setting up virtual machines for 318
- SYMPTOM1 operand
  - SYSTEM\_USERIDS statement 233
- SYMPTOM2 operand
  - SYSTEM\_USERIDS statement 233
- syntactic statements in system configuration file 27
- syntax diagram
  - examples
    - default xvii
    - fragment xvii
    - return arrow xvi
    - symbols xvi
    - variable xvi
  - table xvi
- syntax, checking statements in system configuration file 57
- syntax, explanation of, notational xvi
- SYSACNT macro 6, 661
- SYSADDIN macro 663
- SYSAFFIN user directory control statement 569
- SYSCLEAR 398
- SYSCLR operand of SYSRES macro 391
- SYSCPVOL macro 620, 664
  - list 447
  - search order 447
- SYSDUMP macro 6, 667
- SYSEND macro 668
- YSEREP macro 6, 669
- SYSEXCL macro 671
- SYSFCN macro 575, 673
- SYSFORM macro 675
- SYSID macro 677
- SYSINCL macro 680
- SYSJRL macro 682
- SYSMAXU macro 686
- SYSOPR macro 6, 687
- SYSOPTS macro 689
- SYSPCLAS macro 691
- SYSRES macro 391, 605, 693
- SYSSTORE macro 697
- SYSSYMP macro 699
- system
  - administration tasks 4
  - attributes, establishing 136
  - bring-up virtual machine 6
  - control files 3
  - definition
    - of set of real devices 188
  - dump virtual machine, defining a user ID for 667
  - function
    - assigning new classes 438, 443
  - identifier, processor 677
  - integrity 383
    - general definition 389
    - guidelines 390
    - maintaining 389
  - joining 353
  - maximum number supported for cross system link 334, 338
  - modification requirements 390
  - naming groups 133
  - not active after start CSECOM 355
  - operator, automatic exclusion of 659
  - operator, primary user ID 687
  - outages, hardware 372
  - participating in cross system commands and spooling, specifying 271
  - planning tasks 4
  - processor identifier 677
  - residence disk 693
  - residence volume 338
  - time zone, defining 239
- system command, cross
  - excluding virtual machines' input spool files from 274
  - excluding virtual machines' output spool files from 276
  - specifying systems participating in 271
- system configuration file 51
  - advantages 23
  - case 56
  - checking statement syntax 57
  - comments 55
  - content overview 25
  - continuations 55
  - converting to xx
  - CP file system, overview 27
  - format 55
  - general rules 55
  - ignoring errors in 241
  - imbedding files into 155
  - limit and audit access 380

system configuration file (*continued*)

- minimum content 25
- order of statements 57
- overview 23
- real devices, overview 25
- record qualifiers 56
- semantic statements 27
- specifying file name via SAPL 42
- specifying file type via SAPL 43
- spool file processing, overview 27
- statements
  - ALTERNATE\_OPERATORS 58
  - ASSOCIATE EXIT 60
  - ASSOCIATE MESSAGES 64
  - ASSOCIATE MSGS 64
  - CHARACTER\_DEFAULTS 67
  - CP\_ACCESS 70
  - CP\_ADDON\_INITIALIZE\_ROUTINES 72
  - CP\_OWNED 73
  - CPXLOAD 76
  - DEFINE ALIAS 81
  - DEFINE CMD 84
  - DEFINE COMMAND 84
  - DEFINE DIAGNOSE 90
  - DEFINE EXIT 94
  - DEVICES 105
  - DISABLE CMD 110
  - DISABLE COMMAND 110
  - DISABLE DIAGNOSE 112
  - DISABLE EXITS 114
  - DISTRIBUTE 116
  - DRAIN 118
  - EMERGENCY\_MESSAGE\_CONSOLES 124
  - ENABLE CMD 126
  - ENABLE COMMAND 126
  - ENABLE DIAGNOSE 128
  - ENABLE EXITS 130
  - ENFORCE\_BY\_VOLID 132
  - EQUATE 133
  - EXTERNAL\_SYNTAX 135
  - FEATURES 136
  - FORM\_DEFAULT 150
  - HOT\_IO\_RATE 152
  - IMBED 155
  - INIT\_MITIME 157
  - IODF 158
  - JOURNALING 160
  - LOGO\_CONFIG 164
  - MODIFY CMD 165
  - MODIFY COMMAND 165
  - MODIFY DIAGNOSE 169
  - MODIFY EXIT 172
  - MODIFY PRIV\_CLASSES 176
  - OPERATOR\_CONSOLES 180
  - PRINTER\_TITLE 182
  - PRIV\_CLASSES 184
  - RDEVICE 188
  - SAY 217
  - SET 219
  - START 220
  - STORAGE 222

system configuration file (*continued*)

- statements (*continued*)
  - summary 51
  - SYSTEM\_DATEFORMAT 225
  - SYSTEM\_IDENTIFIER 226
  - SYSTEM\_IDENTIFIER\_DEFAULT 228
  - SYSTEM\_RESIDENCE 230
  - SYSTEM\_USERIDS 232
  - THROTTLE 236
  - TIMEZONE\_BOUNDARY 237
  - TIMEZONE\_DEFINITION 239
  - TOLERATE\_CONFIG\_ERRORS 241
  - TRANSLATE\_TABLE 243
  - USER\_DEFAULTS 248
  - USER\_VOLUME\_EXCLUDE 251
  - USER\_VOLUME\_INCLUDE 253
  - USER\_VOLUME\_LIST 255
  - USERFORM 247
  - XLINK\_DEVICE\_DEFAULTS 259
  - XLINK\_SYSTEM\_EXCLUDE 263
  - XLINK\_SYSTEM\_INCLUDE 264
  - XLINK\_VOLUME\_EXCLUDE 266
  - XLINK\_VOLUME\_INCLUDE 268
  - XSPPOOL\_SYSTEM 271
  - XSPPOOL\_TRACE 273
  - XSPPOOL\_XLIST\_INPUT 274
  - XSPPOOL\_XLIST\_OUTPUT 276
- statements, summary 25
- syntactic statements 27
- updating 34
- USER\_DEFAULTS 248
- using 23
  - XLINK\_VOLUME\_EXCLUDE statement 338
  - XLINK\_VOLUME\_INCLUDE statement 338
- system console
  - alternate
    - defining on RIOGEN macro 766
  - defining
    - alternate 766
    - list of possible 180
    - primary 766
  - primary
    - defining on RIOGEN macro 766
- system data file
  - user class restructure 441
- system directory
  - DIAGNOSE code X'3C' 394
  - DIAGNOSE code X'84' 394, 396
  - overview 395
- system dump
  - specifying user ID to accumulate records 232
- system execution space, CP
  - loading CP routines into 76
- system generation
  - CSELVOL EXCLUDE generation macro 649
  - CSELVOL INCLUDE generation macro 650
  - RIOGEN generation macro 766
  - screen logo file
    - choosing 285
  - SYSEXCL generation macro 671
  - SYSINCL generation macro 680

- system generation (*continued*)
  - SYSOPTS generation macro 689
  - SYSSTORE generation macro 697
- system identifier
  - creating 226, 228
- system link, cross
  - excluding DASD 266
  - excluding DASD volumes 649
  - excluding systems 263
  - including DASD 268
  - including DASD volumes 650
  - including systems 264
- system log message
  - specifying whether CP should display 136
- system operator
  - specifying disconnect status 232
  - specifying user ID 232
- system programming
  - journaling
    - including facility in system 160
- system residence volume
  - defining 73
  - describing layout 230
- system service
  - \*ACCOUNT 233
  - \*BLOCKIO 494
  - \*CONFIG 494
  - \*CRM 494
  - \*IDENT 496
  - \*LOGREC 314, 494, 669
  - \*MONITOR 494
  - \*MSG 494
  - \*MSGALL 494
  - \*RPI 494
  - \*SIGNAL 494
  - \*SYMPTOM 318, 494
  - \*SYMPTOM, specifying virtual machines for 699
  - accounting 295, 661
  - collecting accounting records 297
  - collecting EREP records 315
  - collecting symptom records 319
  - communication requirements 497
  - CP recording 295
  - symptom record recording 295
- system spooling, cross
  - excluding virtual machines' input spool files
    - from 274
  - excluding virtual machines' output spool files
    - from 276
  - identifying volumes for 73
  - specifying systems participating in 271
  - trace tables
    - allocating space for 273
- system storage
  - configuring 222, 697
- SYSTEM\_IDENTIFIER statement 53, 54, 225, 226
- SYSTEM\_IDENTIFIER\_DEFAULT statement 54, 228
- SYSTEM\_RESIDENCE statement 54
  - description 230
- SYSTEM\_USERIDS statement 54, 232

- system-managed 3800 printers
  - delayed purge queue 721
- system-wide default
  - for logical character-delete symbol 67
  - for logical escape symbol 67
  - for logical line-delete symbol 67
  - for logical line-end symbol 67
  - for logical tab symbol 67
- System/390 architecture 397
- SYSTIME macro 701
- SYSUVOL macro 8, 620, 703

## T

- tab character
  - definition 67
- TAB operand
  - CHARACTER\_DEFAULTS statement 68
- table
  - colors for the CHOOSE\_LOGO and ONLINE\_MESSAGE files 283
  - input area fields 284
  - logo configuration file statements 279
  - map record defaults for DASD 652
  - syntax diagram xvi
  - system configuration file statements 51
- table entry block, command
  - defining new CP 84
- table, trace
  - allocating space for
    - internal 222, 697
    - XSPOOL 273
- table, translate
  - replacing 243
- TAG command 369
- tailoring
  - privilege classes 392, 396
- tape
  - clearing 398
  - drive
    - coding an RDEVICE macro 743
    - configuration guide 47
    - defining in system configuration file 47
  - format
    - stand-alone dump 777
  - library dataserver 566
  - units
    - defining 207
    - unsupported, defining 209
- TAPE operand
  - HOT\_IO\_RATE statement 153
  - RDEVICE statement 210
- TDISK (temporary disk)
  - accounting record 301
  - clearing 136, 384, 398
  - space 338
- TDISK operand
  - DRAIN statement 119
  - START statement 221
- TDSK operand
  - DRAIN statement 119

TDSK operand (*continued*)  
 START statement 221

TELE2\_ADAPTER operand  
 RDEVICE statement 193

Telegraph Terminal Control type 2 193, 738

teleprocessing integrated adapter  
 coding an RDEVICE macro 738

TEMPDISK operand  
 DRAIN statement 119  
 START statement 221

TEMPORARY operand  
 CPXLOAD statement 78

terminal  
 choosing logo 285  
 coding an RDEVICE macro 754  
 input line  
 defining file for 291  
 unsupported  
 defining 209

TERMINAL LINESIZE command 331

TERMINAL operand  
 HOT\_IO\_RATE statement 153  
 RDEVICE statement 210

text  
 of status fields, redefining 293

text deck  
 creating for impact printer 412  
 description 411  
 for impact printer 422  
 input file for image library 411

TEXT operand  
 CPXLOAD statement 77

THROTTLE statement 54, 236

time zone  
 choosing at IPL 237  
 defining 239  
 determining local 237

Time-Of-Day (TOD) clock  
 not changing during IPL 142  
 SYSTIME macro 701

TIMEOUT operand  
 XSPPOOL\_SYSTEM statement 272

TIMEZONE\_BOUNDARY statement 54, 237

TIMEZONE\_DEFINITION statement 54, 239

token ring communication adapter  
 defining in system configuration file 49

Token Ring local area network 738

tolerate  
 errors in system configuration file 241

TOLERATE\_CONFIG\_ERRORS statement 54, 241

TOP operand  
 PRINTER\_TITLE statement 182

topography of CSE complex, describe 653

TPF (Transaction Processing Facility)  
 OPTION user directory control statement 535

trace  
 spool synchronization request 371

TRACE operand  
 STORAGE statement 223  
 SYSSTORE macro 697

trace table  
 allocating space for  
 internal 222, 697  
 XSPPOOL 273

track location format default, change CSE 259, 646

TRACK operand  
 XLINK\_VOLUME\_INCLUDE statement 269

TRANSFER command 369

transfer data to CP spool files 532

transition support, non-disruptive 392

TRANSLATE operand  
 TRANSLATE\_TABLE statement 243

translate table  
 replacing 243

TRANSLATE\_TABLE statement 54, 243

translating guest virtual channel programs 427

TRK operand  
 CSELVOL INCLUDE macro 650

TXTLIB operand  
 CPXLOAD statement 77

TYPE operand for RDEVICE statement  
 3800 printers 212  
 advanced function printers 189  
 card punches 190  
 card readers 192  
 communication controllers 193  
 DASD 195  
 graphic display devices 198  
 impact printers 202  
 line adapters 193  
 special devices 205  
 unsupported devices 209

## U

U. S. Telegraph Terminal Control type 2 738

UCS (Universal Character Set)  
 buffer images  
 naming convention 416  
 examples of new buffer images  
 3203 printer 416  
 3211 printer 417  
 3262 printer 417

UCSI macroinstruction 415

UCSICCW macroinstruction 415

unit record device  
 configuration guide 48  
 dedicated 11  
 defining for virtual machine  
 spooled unit record device 562  
 defining in system configuration file 48  
 planning 11  
 spooled 11

UNIT RECORD operand  
 HOT\_IO\_RATE statement 153

UNIVERSAL\_CHARSET operand  
 RDEVICE statement 203

UNLOCK operand  
 D8ONECMD statement 483

unlock pages 394, 397

- unsolicited interrupt
  - changing 152
  - setting 152
- unsupported device
  - CCW translation 428
  - coding an RDEVICE macro 763
  - defining 209
  - device tables 428
  - not guaranteed to run properly 46
- UNSUPPORTED operand
  - RDEVICE statement 209
- user
  - accounting record 299
  - administration 5
  - class restructure
    - activating a class override file 441
    - activating an override file 438
    - assigning commands to types of users 435
    - assigning new classes 438
    - associating classes with users and commands 436
    - changing back to IBM-defined user classes 442
    - changing QUERY and SET operand privilege classes 440
    - changing the directory 443
    - CLASS control statement 443, 445
    - class override file 438, 439, 441
    - CLEAR option on OVERRIDE utility 442
    - commands 438
    - COMMANDS command 446
    - creating a class override file 438
    - defining privilege classes for a virtual machine 444
    - defining users' needs 434
    - description 433
    - diagnose codes 438
    - directory entry example 445
    - directory, changing 443
    - displaying commands available to a user 446
    - documentation considerations 438
    - file 441
    - HELP files 437
    - IBM-defined user classes, changing back to 442
    - OVERRIDE control statement 438
    - override file 438
    - OVERRIDE utility 441
    - overview 433
    - planning a new user class structure 433
    - preparing an override file 438
    - sample class override file 439
    - security 383
    - system functions 438
    - system integrity 383
    - USER control statement 443
    - VALIDATE option on OVERRIDE utility 441
    - verifying a class override file 441
  - directory
    - bringing online 31
    - changing 447, 454
    - checking for errors 454
    - checking VM/SP directory statements 455
- user (*continued*)
  - directory (*continued*)
    - continuation comma 452
    - continued across multiple records 452
    - continued directory control statements 452
    - creating 447, 451
    - determining how much space is required for 608
    - quoted string operands 452
    - running the directory creation program 453
    - size constraints 610
    - updating 447
  - directory control statement
    - ACCOUNT 456
    - ACIGROUP 458
    - APPCPASS 459
    - AUTOLOG 461, 578
    - CLASS 462
    - CONSOLE 463
    - CPU 465
    - CRYPTO 468
    - D8ONECMD 483
    - DASDOPT 472
    - DATEFORMAT 475
    - DEDICATE 477
    - DIRECTORY 480
    - GLOBALDEFS 485
    - GLOBALOPTS 486
    - INCLUDE 487
    - introduction 447
    - IOPRIORITY 488
    - IPL 490
    - IUCV 493
    - LINK 500
    - LOAD 504
    - LOGONBY 509
    - MACHINE 510
    - MAXSTORAGE 512
    - MDISK 513
    - MINIOPT 524
    - NAMESAVE 527
    - NICDEF 529
    - NOPDATA 532
    - OPTION 533
    - overview 447
    - POOL 540
    - POSIXGLIST 541
    - POSIXGROUP 543
    - POSIXINFO 545
    - POSIXOPT 548
    - PROFILE 551
    - SCREEN 553
    - SHARE 555
    - SPECIAL 557
    - SPOOL 562
    - SPOOLFILE 565
    - STORAGE 568
    - summary 447
    - SYSAFFIN 569
    - USER 572
    - XAUTOLOG 578
    - XCONFIG 580

- user (*continued*)
  - directory control statement (*continued*)
    - XSTORE 585
  - ID
    - accounting virtual machine 661
    - defining for virtual machine user 572
    - error recording virtual machine 669
    - primary system operator 687
    - system dump virtual machine 667
    - system operator 232
    - to perform special functions, specifying 232
  - identification requirements 391
  - logon limit 686
  - owned DASD volumes, generating a list of 703
  - planning 5
  - supported in CSE 334
  - tasks 5
- user defaults configuration file
  - specifying default attributes 248
- user form name
  - creating list of 247
  - generating default 150
- user owned volume
  - defining with generic volume ID 253, 680
  - excluding DASD from list 671
  - excluding from user volume list 251
  - generating list of 255
- USER user directory control statement 443, 448, 572
- user volume list 26
  - excluding volumes from 251, 671
  - including volumes in 253, 680
- USER\_DEFAULT operand
  - PRIV\_CLASSES statement 185
- USER\_DEFAULTS statement 54, 248
- USER\_VOLUME\_EXCLUDE statement 54, 251
- USER\_VOLUME\_INCLUDE statement 54, 253
- USER\_VOLUME\_LIST statement 54, 255
- user-initiated accounting records 310
- USERFORM statement 54, 247
- USERID operand
  - CHOOSE\_LOGO statement 286
- users, specifying maximum number 136
- using
  - programs 37
  - z/VM stand-alone programs 37
- utilities
  - RETRIEVE
    - retrieving EREP records 316
- utility
  - CCDUMP 321
  - CCLOAD 321
  - CPFMTXA (CMS) 7, 665
  - DIRECTXA 448
    - checking directory for errors 454
    - checking for VM/SP statements 455
    - creating user directory with 451, 453
  - GENIMAGE 411
  - ICHUT100 381
  - IMAGELIB 423
  - IMAGEMOD 424
  - IOCP 184, 673, 705

- utility (*continued*)
  - OVERRIDE 382, 441
  - RETRIEVE 661
  - SALIPL 37
  - sample programs 641
  - SAPL 37
  - stand-alone dump 403
  - VMFHASM 412
  - XLINK 349
- utility program, sample
  - DRAWLOGO 282, 642

## V

- VAFP, virtual advanced functional printer 563
- VALIDATE option on OVERRIDE utility 441
- validation
  - requirements 392
- VCNA (VTAM Communications Network Applications)
  - defining SNA/CCS to 330
- verify
  - accuracy of system configuration file statement syntax 57
  - links 353
  - spool status 354
  - storage access 384
- version
  - creating new for existing CP 84
- virtual
  - and real reserve/release
    - when to use concurrent 626
  - device
    - spooled, defining 562
  - planning 9
  - processor, defining 465
  - reserve/release
    - DASD sharing without using 622
    - when to use 621
  - storage
    - administration 7
    - CP module requirements 589
    - defining for virtual machine user 572
    - planning 7
- virtual console
  - generating default user form name for 150
- virtual disks in storage
  - accounting record 307
  - allocating system storage availability 136
- virtual LAN segment, defining 97
- virtual LAN segment, modifying 175
- virtual machine
  - architecture 510
  - attaching expanded storage to 592
  - authorized for concurrent copy sessions 566
  - authorizing to use IUCV 493
  - characteristics, special 533
  - console
    - defining 180, 766
    - for emergency messages, defining 124
  - defining for
    - command privilege class 572

- virtual machine (*continued*)
  - defining for (*continued*)
    - delete symbol 572
    - escape character 572
    - expanded storage facility 585
    - line delete symbol 572
    - line end symbol 572
    - password 572
    - privilege classes 444
    - storage 572
    - user ID 572
  - defining one to use during outages 372
  - defining set of with same characteristics 540
  - error recording 314
  - file pool server 622
  - host access list size, defining 581
  - log on limit
    - specifying 136
  - multiple, DASD sharing with other systems 624
  - operator's console, define 463
  - resource usage accounting record 299
  - security facilities
    - including in system 160
  - service 295
  - service pool, setting up 325
  - setting up data storage management 327
  - setting up for accounting 295
  - setting up for collecting accounting records 297
  - setting up for collecting EREP records 315
  - setting up for collecting symptom records 319
  - setting up PSF/VM 326
  - setting up symptom record recording 318
  - special characteristics 533
  - specifying for \*SYMPTOM system service 699
  - spool file characteristics 565
- Virtual Machine/ESA
  - defining VSM logos 330
  - defining VSM to 330
  - features 335
  - I/O reconfiguration 12
  - major files 3
  - security 142
  - security facilities 383
- VIRTUAL operand
  - DEFINE ALIAS statement 81
  - DEFINE CMD statement 84
  - DEFINE COMMAND statement 84
  - DISABLE CMD statement 110
  - DISABLE COMMAND statement 110
  - ENABLE CMD statement 126
  - ENABLE COMMAND statement 126
  - MODIFY CMD statement 165
  - MODIFY COMMAND statement 165
- virtual printer
  - generating default user form name for 150
- virtual punch
  - generating default user form name for 150
- virtual switch segment, defining 100
- virtual switch segment, modifying 178
- VM\_LOGO operand
  - JOURNALING statement 162

- VM\_READ operand
  - STATUS statement 294
- VM/ESA commands, restriction of 393
- VMAC operand
  - DEFINE CMD statement 87
  - DEFINE COMMAND statement 87
  - DEFINE DIAGNOSE statement 91
- VMFHASH utility 412, 422
- VMFHLASM utility 422
- VMLAN statement 54, 257
- VMLINK CMS command 620
- VOLID operand
  - DRAIN statement 118
  - START statement 220
  - SYSTEM\_RESIDENCE statement 230
- volume identifier
  - excluding from user volume list 671
  - using generic to define user volumes 253, 680
  - using to exclude volumes from user volume list 251
- volume list
  - CP-owned, defining 73
  - excluding user volumes from 251, 671
  - including user volumes in 253, 680
- VPROT operand
  - DEFINE CMD statement 87
  - DEFINE COMMAND statement 87
  - DEFINE DIAGNOSE statement 92
- VSCS
  - defining SNA/CCS to 331
- VSM (VTAM Service Machine)
  - choosing logo 285
  - defining logos 330
  - defining to z/VM 330
  - starting 332
  - termination 332
- VSM\_VMID operand
  - CHOOSE\_LOGO statement 287

## W

- wait state if parm disk is not CMS-formatted 51
- warm start
  - data
    - allocating DASD space for 606
    - specifying whether CP should attempt 136
- warning
  - 3525 card punch 190
  - 3705, 3720, 3725, and 3745 BSC line adapters 193
  - 3850 Mass Storage System 195
  - about migration incompatibilities 656, 678
  - against using 2250 display 124
  - against using 3250 display 124
  - calling entry point after loading CP routines 78
  - changing defaults on CSELVOL INCLUDE
    - macro 651
  - CONTROL operand of CPXLOAD 78
  - data integrity loss 264
  - defining advanced function printers 189
  - moving volumes containing spool space 74
  - removed
    - issuing multiple CP commands 483

- warning (*continued*)
  - removed (*continued*)
    - specifying real storage available for CP 697
  - sharing system residence and paging volumes 73
  - system security 234
  - turning off hot I/O rate 153
  - users migrating from VM/XA SP 451, 454
- WARNING command, altering output 373
- WEST operand
  - TIMEZONE\_DEFINITION statement 239
- WNG command 276
- write-through minidisk 597

## X

- X.25 network (high-level data link control) 738
- X\$DRWL\$X SAMPXEDI file 641
- X\$DRWL\$X XEDIT file 641
- X'15', preventing use of 483
- XAUTOLOG command
  - journaling 385
  - specifying invalid password 299
  - using during initialization 332
- XAUTOLOG user directory control statement 578
- XCONFIG user directory control statement 580
- XLINK CHECK command 353
- XLINK FORMAT command
  - using to format volume 350, 652
  - using to specify CSE area 340
- XLINK RESET command 359
- XLINK utility program 349
- XLINK\_DEVICE\_DEFAULTS statement 54, 259
- XLINK\_SYSTEM\_EXCLUDE statement 54, 263
- XLINK\_SYSTEM\_INCLUDE statement 54, 264
- XLINK\_VOLUME\_EXCLUDE statement 54, 266, 338
- XLINK\_VOLUME\_INCLUDE statement 54, 268, 338
- XSPPOOL QUERY command 354
- XSPPOOL trace table
  - allocating space for 273
- XSPPOOL\_SYSTEM statement 54, 271
- XSPPOOL\_TRACE statement 54, 273
- XSPPOOL\_XLIST\_INPUT statement 54, 274
- XSPPOOL\_XLIST\_OUTPUT statement 55, 276
- XSTORE user directory control statement 585, 592

## Z

- zone, time
  - choosing at IPL 237
  - defining 239
  - determining local 237



---

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

z/VM  
CP Planning and Administration  
Version 5 Release 1

Publication No. SC24-6083-01

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you?  Yes  No

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

---

Name

---

Address

---

Company or Organization

---

Phone No.



Fold and Tape

Please do not staple

Fold and Tape



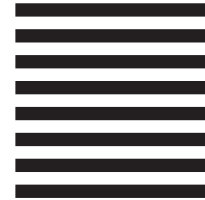
NO POSTAGE  
NECESSARY  
IF MAILED IN THE  
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation  
Department 55JA, Mail Station P384  
2455 South Road  
Poughkeepsie, New York 12601-5400



Fold and Tape

Please do not staple

Fold and Tape





Program Number: 5741-A05

Printed in USA

SC24-6083-01



Spine information:



z/VM

CP Planning and Administration

Version 5 Release 1