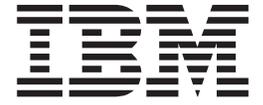
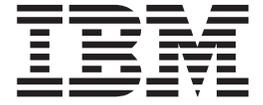


z/OS



MVS Planning: Operations

z/OS



MVS Planning: Operations

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 193.

Third Edition, March 2002

This is a major revision of SA22-7601-01.

This edition applies to Version 1 Release 3 of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405

FAX (Other Countries):

Your International Access Code +1+845+432-9405

IBMLink (United States customers only): IBMUSM10(MHVRCFS)

Internet e-mail: mhvrcfs@us.ibm.com

World Wide Web: <http://www.ibm.com/servers/eserver/zseries/zos/webqs.html>

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

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

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.

© **Copyright International Business Machines Corporation 1988, 2002. All rights reserved.**

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

Contents

Figures	vii
Tables	ix
About This Book	xi
Who Should Use This Book	xi
How To Use This Book	xi
Where to Find More Information.	xii
Using LookAt to look up message explanations	xii
Accessing licensed books on the Web	xiii
Do You Have Problems, Comments, or Suggestions?	xiii
Summary of changes	xv
Chapter 1. Planning Your MVS Operations	1
Operations Goals	1
The Operating Environment.	2
Multiple Console Support and the MVS Environment	2
Sysplex Operating Environment	5
Using MCS and SMCS Consoles in a System or Sysplex.	5
Extended MCS Consoles	6
SDSF and MVS Operations Planning.	10
RMF and MVS Operations Planning	10
Automated Operations and z/OS Operations Planning	11
Remote Operations and MVS Operations Planning.	11
ESCON and Operations Planning	12
Chapter 2. Defining Your Console Configuration	13
Choosing How to Define Your Console Configuration	13
Shared Consoles in Mixed Sysplexes	14
Using CONSOLxx.	14
CONSOLE Statement	15
INIT, DEFAULT, and HARDCOPY Statements	16
CONSOLxx and the Sysplex	20
SMCS Console Considerations	26
Installing SMCS	26
Removing Console Definitions from a Configuration	36
Sample Invocation of IEARELCN	36
Environment	37
Defining Devices as MCS or SMCS Consoles	38
Devices MVS Can Use as MCS Consoles	39
Using Console Names	40
Attaching Consoles to Particular Systems in a Sysplex	42
Planning Console Recovery	43
Recovery Considerations	43
Parmlib Members and Console Recovery	44
Alternate Console Groups and Console Backup.	44
Using CNGRPxx to Define Alternate Console Groups.	44
Activating CNGRPxx.	45
Alternate Console Groups and CONSOLxx	46
Changing the Specification of Alternate Console Groups.	46
Changing Console Alternates without Re-IPLing.	46
Alternate Console Groups and Extended MCS Consoles	46

Using the ALTERNATE Keyword on the CONSOLE Statement	47
Alternate Console of Last Resort	47
Console Switching and Console Recovery	48
The SWITCH CN Command	48
Console Recovery and the RESET CN Command	49
Role of the Master Console During Console Recovery	49
No-Master-Console Condition	49
No-Consoles Condition	50
Selecting a Master Console Using Alternate Console Groups	50
Display of Synchronous Messages	51
Planning Console Security.	52
Controlling Command Authority with the AUTH Attribute	52
Assigning a Console Master Authority	53
Using RACF to Control Command Authority and Operator Logon	54
Defining RACF Profiles	55
MVS Commands, RACF Access Authorities, and Resource Names.	62
Other Ways to Control Command Authority for Consoles	68
Planning Console Functions for Operators	68
How to Control the Use of an MCS Console	69
Defining the USE Attribute.	71
Message Display and the Full-Capability Console Screen	72
Specifying Automatic Message Deletion for MCS or SMCS Consoles	72
Temporarily Suspending the Screen Roll	76
Comparison of Roll, Roll-Deletable, Wrap Modes, and HOLDMODE	77
Manual Deletion of Messages	77
How Operators Specify Message Numbering	78
Using SEG to Delete Groups of Messages from the Screen	79
Status Displays and MCS and SMCS Consoles	79
Setting Up Out-of-Line Display Areas on a Console	80
Where to Route Status Displays	81
Defining the Time Interval for Updating a Dynamic Status Display	82
Controlling the Format of Messages and Status Information on Console Screens	82
Displaying Jobname, Data Set Status, and TSO/E Information	84
Adding Information to Mount Messages	85
Defining PFKs and Other Command Controls for Consoles	85
Setting up PFKs for Consoles	85
Defining the Command Delimiter for Full-Capability Consoles	88
Hardcopy Processing	88
The Hardcopy Message Set	89
The Hardcopy Medium	90
Chapter 3. Managing Messages and Commands	95
General Characteristics of Messages and Commands	95
Message and Command Routing	96
Message Flow in a System	97
Command Flow in a System	97
Command Flooding	98
Message and Command Flow in a Sysplex	98
Messages in a Sysplex	98
Message Recovery Following System Failures	99
Routing Messages	104
Defining Routing Codes	105
Handling Messages without Routing Codes	105
Defining Message Levels for a Console	106
Defining the UD Attribute for Consoles.	107

Directing Messages from Other Systems to a Console in a Sysplex	108
Replying to Messages from Other Systems in a Sysplex	108
Directing Messages that Are Eligible for Automation to Extended MCS Consoles	108
Routing Commands	109
Using CMDSYS on the CONSOLE Statement	109
Using the ROUTE command	109
Using the Command Prefix Facility	110
Using the L=Operand on Certain Commands	111
Sharing System Commands By Using System Symbols	111
MPF and MVS Operations Planning	114
Specifying Message Presentation	115
Suppressing Messages	115
Retaining Messages	116
Selecting Messages for Automation	119
Automation in a Sysplex	120
Installation Exits for Messages and Commands	121
Controlling WTO and WTOR Message Buffers	123
Controlling Reply IDs for WTOR Messages	123
Controlling Automatic Ending of Multi-line WTO Messages	124
Aggregating Messages Returned to the ROUTE Command	125
Controlling Write-to-Log (WTL) Message Buffers	128
Handling Translated Messages	128
Summary of MVS Message and Command Processing Services	134
Chapter 4. Planning for Basic Operation Procedures	135
Initializing the System	136
The System Console and Message Processing	136
Using the System Console	137
Specifying LOAD Information	137
The NIP Console	137
The System Console and CONSOLxx	138
Problem Determination and the System Console	139
Specifying the Time-of-Day Clock and the JES Subsystem	142
CLOCKxx and the Sysplex	142
Handling Wait States	143
Initializing the Master Console	143
Interacting with System Functions	143
Device Allocation	143
Hot I/O Detection	145
Device Boxing	146
Considerations for Operators	146
Controlling Shared DASD	146
Specifying Shared DASD Mount Characteristics	147
Chapter 5. Examples and MVS Planning Aids for Operations	149
Summary of CONSOLxx and Commands to Change Values	149
Controlling Extended MCS Consoles Using RACF	152
Defining the User Profile of an Extended MCS Console	153
Granting the User Access to the RACF OPERCMDS class	153
Allowing a TSO/E User to Issue the CONSOLE Command	154
Changing Console Attributes Using RACF	154
Using RACF to Control APF Lists	154
Command Authorization	155
Defining Command Profiles	155
Controlling How to Add or Delete APF List Entries for a Library	156

Controlling How to Change the APF List Format	157
Using RACF to Control Dynamic Exits	158
Command Authorization	158
Defining Command Profiles	158
Controlling Defining a Dynamic Exit	159
Controlling Adding, Modifying or Deleting Exit Routines	160
Controlling How to Undefine a Dynamic Exit	161
Controlling How to Obtain a List of the Dynamic Exits	161
Controlling Calling of a Dynamic Exit's Routines	162
Controlling Recovering of Dynamic Exit Processing	163
Using RACF to Control LNKLST Concatenations	164
Command Authorization	164
Defining Command Profiles	164
Controlling Defining a LNKLST Set	165
Controlling Adding a Data Set to a LNKLST Set	166
Controlling Deleting a Data Set from a LNKLST Set	166
Controlling Removing the Definition of a LNKLST Set	167
Controlling Testing of a LNKLST Set	168
Controlling Updating of a Job's LNKLST Set	169
Controlling Activation of a LNKLST Set	169
Using RACF to Control Dynamic LPA	170
Command Authorization	170
Defining Command Profiles	171
Controlling Adding A Module to LPA after IPL	171
Controlling Deleting A Module from LPA after IPL	172
Managing Messages with a Console Cluster	173
Setting Up and Using a Master Console Cluster	173
Setting Up Console Recovery for the Consoles	175
Defining Routing Codes for the Consoles	175
Defining the Operating Modes and the Message Levels for the Consoles	176
Setting Up Display Areas	176
Setting Up a TRACK Display	177
Setting Message Roll Rates and Message Deletion Specifications for the Consoles	179
Setting Up a Periodic Display of Outstanding Requests	180
Summary of Contents of CONSOLxx for the Cluster	180
Defining PFKs for the Master Console	181
Summary of the PFK Definitions for the Cluster	184
Activating the PFK Table	184
Using the Master Console Cluster and Setting It Up Again	185
The 3290 as a Console Cluster	185
Defining a Console Configuration for a Sysplex Environment	185
Planning Your Console Configuration for Each System	186
Defining CONSOLxx for Each System	187
Appendix. Accessibility	191
Using assistive technologies	191
Keyboard navigation of the user interface	191
Notices	193
Programming Interface Information	194
Trademarks	194
Glossary	197
Index	203

Figures

1. Console Configuration for an MVS System	4
2. Sysplex Showing Console attachments	5
3. Console Configuration in a Sysplex with Two Systems and Four MCS Consoles	22
4. Console Configuration in a Sysplex with Two Systems and Four MCS Consoles	24
5. Console Configuration in a Sysplex with Four MCS Consoles Attached to One System	25
6. Sample LOGON Mode Table Entry	28
7. SMCS Console Selection Screen	35
8. Screen Formats of a Full-Capability, Status Display, and Message Stream Console	71
9. Example of a Full Wrap Mode Screen	75
10. Example of the Wrap Mode Screen after the Next Wrap	75
11. Sample Screen Showing Two Out-of-Line Display Areas on a Full-Capability Console	81
12. PFKTAB01 Parmlib Member.	88
13. Sample JCL for Creating a Run-Time Message File	130
14. Display Areas on Consoles in the Console Cluster	177
15. Single 3290 Screen As a Console Cluster	185
16. Console Configuration for a Two-System Sysplex	186

Tables

1. Console Attributes for MCS and Extended MCS Consoles	7
2. Summary of CONSOLE statement functions	15
3. Summary of INIT statement functions	17
4. Summary of DEFAULT statement functions	19
5. Summary of HARDCOPY statement functions	19
6. Scope of CONSOLxx Keywords	21
7. Keyword Definitions	28
8. PSERVIC Values for SNA Devices	28
9. Console Switching Using the ALTERNATE or ALTGRP keyword	47
10. MVS Commands, RACF Access Authorities, and Resource Names	62
11. Comparison of Roll, Roll-deletable, and Wrap Mode	77
12. IBM Defaults for PFKs	86
13. Automatic restart of log with JES2	93
14. Summary of Message and Command Processing that MVS Provides	134
15. CONSOLE Statement Summary	149
16. Summary of INIT, HARDCOPY, and DEFAULT Statements	151

About This Book

This book contains planning information for MVS operations. It describes how to define and use multiple console support (MCS) consoles, SNA Multiple console support (SMCS) consoles (as of z/OS V1R1), and extended MCS consoles. It also describes how to manage messages and commands in an MVS single-system or sysplex environment.

Who Should Use This Book

System programmers who plan MVS operations and persons who administer the security procedures for their installations should use this book. The book assumes that the user understands the installation's hardware and software, and also understands the general organization and functions of MVS.

Users should have a good understanding of Parmlib and how to use it.

How To Use This Book

Read the chapters in this book in sequence to obtain a good understanding of MVS operations planning.

The book is organized as follows:

- Chapter 1, "Planning Your MVS Operations" on page 1 describes setting operations goals for an MVS environment. It provides a brief introduction to multiple console support (MCS) consoles, SMCS consoles, and extended MCS consoles.
- Chapter 2, "Defining Your Console Configuration" on page 13 describes how to define an MCS and SMCS console configuration. It describes how to define a device as a console and how to define console functions in CONSOLxx of Parmlib. It also provides information to plan for console recovery, console security, and system logging.
- Chapter 3, "Managing Messages and Commands" on page 95 describes how to manage messages and commands for consoles in an MVS environment. It includes information about the message processing facility (MPF), the action message retention facility (AMRF), installation exits to modify messages and commands, and message translation using the MVS message service (MMS).
- Chapter 4, "Planning for Basic Operation Procedures" on page 135 describes how to plan for basic MVS operator tasks like initializing a system and operating MVS on a day-to-day basis.
- Chapter 5, "Examples and MVS Planning Aids for Operations" on page 149 provides examples of defining a console cluster to handle message traffic in an MVS system and defining a console configuration in a two-system sysplex. It also contains reference information to help you in your planning.

In the back of this book, a glossary defines technical terms used in this book.

Where to Find More Information

Where necessary, this book references information in other books, using shortened versions of the book title. For complete titles and order numbers of the books for all products that are part of z/OS, see *z/OS Information Roadmap*.

The table below lists titles and order numbers for books related to other products.

Short Title Used in This Book	Title	Order Number
<i>NetView Automation: Planning</i>	<i>NetView Automation: Planning</i>	SC31-6141
<i>OPC/ESA General Information</i>	<i>IBM SAA Operations Planning and Control/ESA General Information</i>	GH19-6715
<i>System Automation for OS/390 General Information</i>	<i>System Automation for OS/390 General Information</i>	GC28-1541
<i>System Automation for OS/390 Planning and Installation</i>	<i>System Automation for OS/390 Planning and Installation</i>	GC28-1549

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for z/OS messages, system abends, and some codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can access LookAt from the Internet at:

<http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html>

or from anywhere in z/OS where you can access a TSO command line (for example, TSO prompt, ISPF, z/OS UNIX System Services running OMVS).

To find a message explanation on the Internet, go to the LookAt Web site and simply enter the message identifier (for example, IAT1836 or IAT*). You can select a specific release to narrow your search. You can also download code from the *z/OS Collection*, SK3T-4269 and the LookAt Web site so you can access LookAt from a PalmPilot (Palm VIIx suggested).

To use LookAt as a TSO command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO from a disk on your *z/OS Collection*, SK3T-4269 or from the LookAt Web site. To obtain the code from the LookAt Web site, do the following:

1. Go to <http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html>.
2. Click the **News** button.
3. Scroll to **Download LookAt Code for TSO and VM**.
4. Click the ftp link, which will take you to a list of operating systems. Select the appropriate operating system. Then select the appropriate release.
5. Find the **lookat.me** file and follow its detailed instructions.

To find a message explanation from a TSO command line, simply enter: **lookat message-id**. LookAt will display the message explanation for the message requested.

Note: Some messages have information in more than one book. For example, IEC192I has routing and descriptor codes listed in *z/OS MVS Routing and Descriptor Codes*. For such messages, LookAt prompts you to choose which book to open.

Accessing licensed books on the Web

z/OS licensed documentation in PDF format is available on the Internet at the IBM Resource Link Web site at:

<http://www.ibm.com/servers/resourceLink>

Licensed books are available only to customers with a z/OS license. Access to these books requires an IBM Resource Link Web userid and password, and a key code. With your z/OS order you received a memo that includes this key code.

To obtain your IBM Resource Link Web userid and password log on to:

<http://www.ibm.com/servers/resourceLink>

To register for access to the z/OS licensed books:

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **User Profiles** located on the left-hand navigation bar.
3. Click on **Access Profile**.
4. Click on **Request Access to Licensed books**.
5. Supply your key code where requested and click on the **Submit** button.

If you supplied the correct key code you will receive confirmation that your request is being processed. After your request is processed you will receive an e-mail confirmation.

Note: You cannot access the z/OS licensed books unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

To access the licensed books:

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **Library**.
3. Click on **zSeries**.
4. Click on **Software**.
5. Click on **z/OS**.
6. Access the licensed book by selecting the appropriate element.

Do You Have Problems, Comments, or Suggestions?

Your suggestions and ideas can contribute to the quality and the usability of this book. If you have problems using the book, or if you have suggestions for improving it, complete and mail the Reader's Comment Form found at the back of the book.

Summary of changes

Summary of changes for SA22-7601-02 z/OS Version 1 Release 3

The book contains information previously presented in *z/OS MVS Planning: Operations*, SA22-7601-01, which supports z/OS Version 1 Release 2.

New information

- An appendix with z/OS product accessibility information has been added.

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.

Starting with z/OS V1 R2, you may notice changes in the style and structure of some content in this book—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our books.

Summary of changes for SA22-7601-01 z/OS Version 1 Release 2

The book contains information previously presented in *z/OS MVS Planning: Operations*, SA22-7601-00, which supports z/OS Version 1 Release 1.

New information

- A new parameter, UNCOND, has been added to the VARY devnum,HARDCPY,OFF to temporarily turn off hardcopy.

This book contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of changes for SA22-7601-00 z/OS Version 1 Release 1

The book contains information also presented in *OS/390 MVS Planning: Operations*.

Technical changes include:

- Information is added to describe SMCS consoles.

Chapter 1. Planning Your MVS Operations

Managing the operation of z/OS in today's data processing environment has become increasingly important. Operators need to learn new skills and manage more z/OS functions as installations grow in their computing power. Single MVS systems are becoming part of multisystem environments with new demands on the management of the hardware, software, and people required to run those systems. To monitor MVS and to respond to system changes and problems make operations planning more important than ever before.

In order to make decisions about MVS operations planning, you need to understand:

- The operations goals of your installation
- The operating environment and how it will affect those goals.

Operations Goals

MVS operations planning involves issues like workload management, system performance, I/O device management, console security, and console operations, to name a few. But it also involves the business goals and policies established by the installation to allow the installation to grow and handle work efficiently. These needs, of course, vary from installation to installation, but they are important when you plan your MVS operations.

Managing the complexity of MVS requires you to think about the particular needs of the installation. However, any installation might consider the following goals when planning its MVS operations:

- **Increasing system availability.** Many installations need to ensure that their system and its services are available and operating to meet service level agreements. Installations with 24-hour, 7-day operations need to plan for minimal disruption of their operation activities. In terms of MVS operations, how the installation establishes console recovery or whether an operator must re-IPL a system to change processing options are important planning considerations.
- **Controlling operating activities and functions.** As more installations make use of multisystem environments, the need to coordinate the operating activities of those systems becomes crucial. Even for single MVS systems, an installation needs to think about controlling communication between functional areas (like a tape-pool library and the master console area, for example). In both single and multisystem environments, the commands operators can issue from consoles can be a security concern that requires careful coordination. As planner, you want to make sure that the right people are doing the right tasks when they interact with MVS. If your installation uses remote operations to control target systems, you also need to decide about controlling those activities from the host system.
- **Simplifying operator tasks.** Because the complexity of operating MVS has increased, an installation needs to think about the tasks and skills of its operators. How operators respond to messages at their consoles and how you can reduce or simplify their actions are important to operations planning. Also, your installation needs to plan MVS operator tasks in relation to any automated operations that help simplify those tasks.
- **Streamlining message flow and command processing.** In thinking about operator tasks, an installation needs to consider how to manage messages and commands. Operators need to respond to messages. Routing messages to operator consoles, suppressing messages to help your operators manage

increased message traffic, or selecting messages for automated operations can all help you manage system activity efficiently.

- **Single system image.** Single system image allows the operator, for certain tasks, to interact with several images of a product as though they were one image. For example, the operator can issue a single command to all MVS systems in the sysplex instead of repeating the command for each system.
- **Single point of control.** Single point of control allows the operator to interact with a suite of products from a single workstation. An operator can accomplish a set of tasks from a single workstation, thereby reducing the number of consoles the operator has to manage.

The Operating Environment

The operation of an MVS system involves the following:

- Console operations or how operators interact with MVS to monitor or control the hardware and software.
- Message and command processing that forms the basis of operator interaction with MVS and the basis of MVS automation.

Operating MVS involves managing hardware like processors and peripheral devices (including the consoles where your operators do their work) and software such as the MVS operating control system, the job entry subsystem, subsystems like NetView that can control automated operations, and all the applications that run on MVS.

Planning MVS operations for a system must take into account how operators use consoles to do their work and how you want to manage messages and commands. Because messages are also the basis of automated operations, understanding message processing in an MVS system can help you plan MVS automation.

The MVS environment at an installation can affect how you plan to meet your operations goals. Your MVS operating environment might be a single MVS system or a multisystem environment. Depending on the environment, operating MVS can involve different approaches to your planning tasks. For example, planning console security for a multisystem environment requires more coordination than for a single MVS system. But much of the planning you do for a single system can serve as the basis for planning MVS operations in a multisystem environment.

Single MVS systems can be part of multisystem environments like a sysplex or a JES3 complex. In a sysplex, MVS systems can share work and resources; messages and commands can flow from system to system so that communication among systems is also shared.

Multiple Console Support and the MVS Environment

Generally, operators on an MVS system receive messages and enter commands on MCS and SMCS consoles. (Operators can use other consoles such as NetView consoles, to interact with MVS, but this book primarily describes MCS and SMCS consoles and how to plan for their use. Installations can enhance their MVS operations by using extended MCS consoles. See “Extended MCS Consoles” on page 6.)

MCS consoles are devices that are locally attached to an MVS system and provide the basic communication between operators and MVS. (MCS consoles are attached to control devices that do not support systems network architecture (SNA) protocols.)

SMCS consoles are devices that do not have to be locally attached to an MVS system and provide the basic communication between operators and MVS. SMCS consoles use SecureWay Communications Server to provide communication between operators and MVS instead of direct I/O to the console device. SMCS consoles are available as of z/OS V1R1.

In general, there are small differences in the techniques you use to define and activate MCS consoles and SMCS consoles. Once the consoles are activated, however, MCS consoles and SMCS consoles are very much alike.

You can define MCS and SMCS consoles in a console configuration according to different functions. For example, one console can function as a master console for the system. Important messages that require action can be directed to the operator who can act by entering commands on the console. Another console can act as a monitor to display messages to an operator working in a functional area like a tape pool library or to display messages about printers at your installation.

Defining a console configuration is an important part of your MVS operations planning. You define a console configuration by defining the devices you want to use as consoles and their console attributes, in the CONSOLxx Parmlib member. In CONSOLxx, these console attributes control important console functions like the types of commands operators can enter from the console, routing information for messages and commands, and how to use the console. CONSOLxx and the MCS and SMCS console attributes that you can control are described in “Summary of CONSOLxx and Commands to Change Values” on page 149.

Reference

For information about the physical configuration of a master console, see *z/OS MVS Recovery and Reconfiguration Guide*.

Figure 1 shows a console configuration for an MVS system that also includes the system console, an SMCS console, NetView, and TSO/E.

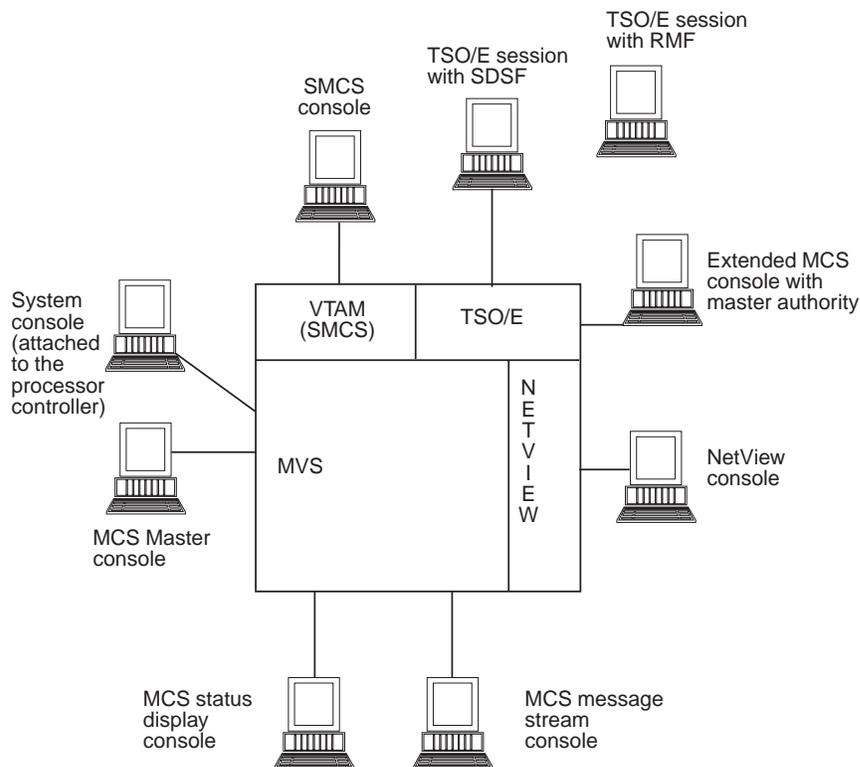


Figure 1. Console Configuration for an MVS System

The system console is attached to the processor controller. An operator can use the system console to initialize MVS and other system software and during recovery situations when other consoles are unavailable.

In addition to MCS and SMCS consoles, the MVS system shown in Figure 1 has a NetView console defined to it. NetView works with system messages and command lists to help you automate MVS operator tasks. You can control many system operations from a NetView console. For information about MVS operations and NetView, see “Automated Operations and z/OS Operations Planning” on page 11.

Users can monitor many MVS system functions from TSO/E terminals. Using the System Display and Search Facility (SDSF) and the Resource Measurement Facility (RMF), TSO/E users can monitor MVS and respond to workload balancing and performance problems. For information about MVS operations and SDSF, see “SDSF and MVS Operations Planning” on page 10. For information about MVS operations and RMF, see “RMF and MVS Operations Planning” on page 10.

An authorized TSO/E user can also initiate an extended MCS console session to interact with MVS. For information on extended MCS consoles, see “Extended MCS Consoles” on page 6.

The MCS consoles in Figure 1 include the following:

- An MCS master console from which an operator can view messages and enter all MVS commands. This console is in full-capability mode because it can receive messages and accept commands. An operator can control the operations for the MVS system from an MCS or SMCS master console.
- An MCS status display console. An operator can view system status information from DEVSERV, DISPLAY, TRACK, or CONFIG commands. However, because

this is a status display console, an operator cannot enter commands from the console. An operator on a full capability console can enter these commands and route the output to a status display console for viewing. An SMCS console cannot be a status display console.

- An MCS message-stream console. A message-stream console can display system messages. An operator can view messages routed to this console. However, because this is a message-stream console, an operator cannot enter commands from the console. You can define routing codes and message level information for the console so that the system can direct relevant messages to the console screen for display. Thus, an operator who is responsible for a functional area like a tape pool library, for example, can view MOUNT messages. An SMCS console cannot be a message stream console.

Sysplex Operating Environment

In a sysplex, you can define an MCS and SMCS console configuration that allows messages and commands to flow from system to system. Figure 2 shows a two-system sysplex, with three consoles attached:

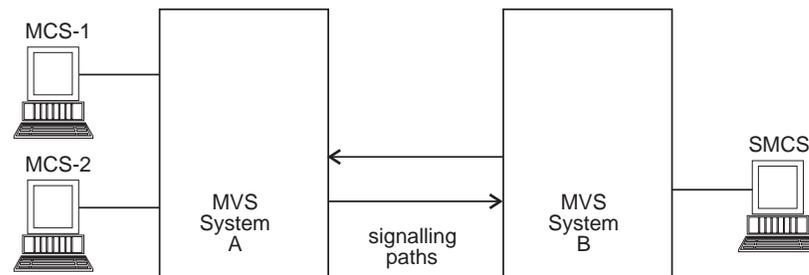


Figure 2. Sysplex Showing Console attachments

In Figure 2, two systems are part of a sysplex with cross-coupling services (XCF) providing signalling paths that allow MCS or SMCS consoles on different systems to communicate with each other. In a sysplex, you can define your MCS or SMCS consoles so that any MCS or SMCS console can receive messages from any system, and commands entered on any MCS or SMCS console can be processed on any system. MCS and SMCS consoles are defined to MVS, and there is no need to define them to JES2, or establish logical associations with other systems.

The sysplex has great flexibility in its console attachments. When you define your MCS or SMCS consoles for a system and IPL the system into a sysplex, your consoles can have a logical association to any system. Any MCS or SMCS console on a system in a sysplex can be the focal point, or MCS and SMCS consoles can share the control they have over systems.

Chapter 5 describes how you can define consoles for a two-system sysplex. For information about defining and tuning the sysplex, see *z/OS MVS Setting Up a Sysplex*.

Using MCS and SMCS Consoles in a System or Sysplex

You can define up to 99 consoles including any subsystem-allocatable consoles for an MVS system. In a sysplex, the limit is 99 consoles for the sysplex. You can exceed this number in a system or sysplex by using extended MCS consoles. (See “Extended MCS Consoles” on page 6.) Therefore, you should examine any product that uses subsystem-allocatable consoles to determine if it could use extended MCS consoles instead.

There is no requirement to have an MCS or SMCS console configured to each system. You can use command and message routing capabilities on one MCS or SMCS console to control multiple systems in the sysplex. MCS or SMCS consoles are not needed on all systems, but you need one MCS or SMCS console to operate the sysplex efficiently.

It is possible to control a sysplex through SMCS consoles alone. In a sysplex with only SMCS consoles, the hardware management console takes on a more important role; it is the console of last resort, for example.

Because SMCS consoles connect through a network, security plays a significant role. For example, you need to require operators to log on, and you must take steps to protect the network connections.

Extended MCS Consoles

To extend the number of consoles on MVS systems or to allow applications and programs to access MVS messages and send commands, an installation can use extended MCS consoles. The use of these consoles can help alleviate the constraint of the 99 MCS console limit. Moving to an extended MCS console base from a subsystem-allocatable console base will allow for easier expansion in a sysplex.

You can define a TSO/E user to operate an extended MCS console from a TSO/E terminal. The user issues the TSO/E CONSOLE command to activate the extended MCS console.

An installation can also write an application program to act as an extended MCS console. An authorized program issues the MVS authorized macro MCSOPER to activate and control the extended MCS console and uses other MVS macros and services to receive messages and send commands.

Using Extended MCS Consoles

Extended MCS consoles provide flexibility in the number of consoles you can use with MVS. Defining extended MCS consoles as part of the MVS console configuration allows you to extend the number of consoles beyond the MCS console limit, which is 99 for an MVS system or sysplex.

Uses for extended MCS consoles include:

- Allow a user through TSO/E to act as an MVS operator and interact with MVS from a TSO/E terminal.
- In an application program, define your own message presentation service, or handle messages and commands that can help automate certain tasks.

For example, you might want to run a program that activates an extended MCS console to control printer operations for a system or sysplex. Because you can direct messages and commands from any system in a sysplex to a specific extended MCS console, you can design programs to control certain automation functions for the entire sysplex.

Both JES2 and JES3 installations can use extended MCS consoles.

Extended MCS Consoles and Console Attributes

An installation can assign to a TSO/E user or to an MVS application program that acts as an extended MCS console many of the same console attributes as an MCS console. These attributes control functions like the types of commands users can issue from the console, console recovery, the routing of messages and commands,

and the format display of messages. “Defining Console Attributes for Extended MCS Consoles” describes how you define these extended MCS console attributes.

Note: The TSO/E CONSOLE command provides only a line-mode interface.

Defining and Protecting Extended MCS Consoles

An installation can define and protect the use of extended MCS consoles through a security product like RACF. To define a user to RACF and control the use of the console, consider the following:

1. Arrange with the RACF security administrator to define a RACF profile for the user of the extended MCS console.

For an interactive TSO/E user, the security or TSO/E administrator can use RACF commands to permit the user to issue the TSO/E CONSOLE command. To customize the use of the TSO/E CONSOLE command, the user can use the TSO/E operator presentation sample defined as a series of Interactive System Productivity Facility (ISPF) panels in SYS1.SAMPLIB. The SYS1.SAMPLIB member name that contains documentation for the TSO/E operator presentation sample is IEATOPSD.

For an MVS application program, the administrator can use RACF commands to protect the use of the MCSOPER macro. In the RACF profile, the administrator defines the name of the extended MCS console that the application must specify on the MCSOPER macro.

2. Ensure that the TSO/E user or application that acts as an extended MCS console has the proper console attributes.

In the RACF profile for the TSO/E user or for the MCSOPER name that the application uses to activate the console, the RACF security administrator can specify the console attributes. An application program can use MCSOPER instead of RACF to specify these console attributes. If both RACF and MCSOPER define console attributes for an extended MCS console, MCSOPER values override the RACF values.

“Controlling Extended MCS Consoles Using RACF” on page 152 describes examples of defining RACF user profiles for an extended MCS console.

Defining Console Attributes for Extended MCS Consoles

If your installation uses RACF to protect extended MCS consoles, RACF maintains information about the console attributes in the OPERPARM segment of each RACF user profile. You can define or alter these attributes using the RACF ADDUSER or ALTUSER commands.

Table 1 shows the console attributes that your installation can control for users of extended MCS consoles. It lists the console attribute, the subkeyword in OPERPARM if you are using RACF, the default value if you do not specify RACF OPERPARM and do not define values through MCSOPER, and the meaning of the default. Notes follow the table:

Table 1. Console Attributes for MCS and Extended MCS Consoles

Console Attribute	RACF OPERPARM Subkeyword	Default value	Meaning of Default
Console group used in recovery	ALTGRP	None	No default; specifies console group for console switching. See “Console Switching and Console Recovery” on page 48.

Table 1. Console Attributes for MCS and Extended MCS Consoles (continued)

Console Attribute	RACF OPERPARM Subkeyword	Default value	Meaning of Default
Command authority for the console	AUTH	AUTH(INFO)	Only informational commands can be issued.
Routing codes for the console	ROUTCODE	ROUTCODE(NONE)	No routing codes established for the console.
Levels of messages directed to the console	LEVEL	LEVEL(ALL)	All levels of messages sent to the console.
Message format for console display	MFORM	MFORM(M)	Display only the message text.
System message scope in the sysplex	MSCOPE	MSCOPE(*ALL)	Display messages from all systems in the sysplex on the console.
Command association in the sysplex	CMDSYS	CMDSYS(*)	Commands are processed on the local system where the console is attached.
Jobname and TSO/E display information	MONITOR	None	No default; monitors jobname and TSO/E information for screen displays. See "Displaying Jobname, Data Set Status, and TSO/E Information" on page 84.
Logging of command responses	LOGCMDRESP	LOGCMDRESP(SYSTEM)	SYSTEM indicates logging is controlled by the value in HARDCOPY CMDLEVEL in CONSOLxx. (NO indicates that the system does not log command responses if the response message was issued by an authorized program).
1-byte migration id assigned to the console	MIGID See Note 1	MIGID(NO)	NO indicates that the console does not require a migration ID. (YES indicates that the system assign a 1-byte migration id to the console.)
Storage limit for message queuing	STORAGE	STORAGE(1)	Storage in megabytes that the system uses for message queuing to the console. The maximum is 2000 megabytes.
Whether the console receives delete-operator-messages (DOMs)	DOM	DOM(NORMAL)	NORMAL indicates that the system direct all appropriate DOMs to the console. (ALL indicates that all systems in a sysplex direct DOMs to the console. NONE indicates that DOMs are not directed to the console.)
Key name for the console	KEY See Note 2	KEY(NONE)	1- to 8-byte character name used in DISPLAY CONSOLES,KEY. A key name allows you to group extended MCS consoles by function and refer to the group using the key name in the DISPLAY command.
Whether the console is to receive action, WTOR, and other important informational messages that the system could not display	UD	UD(N)	Do not display undelivered action and WTOR messages on the console.

Table 1. Console Attributes for MCS and Extended MCS Consoles (continued)

Console Attribute	RACF OPERPARM Subkeyword	Default value	Meaning of Default
Whether the console is to receive messages eligible for automation	AUTO See Note 3	AUTO(NO)	NO indicates that the console does not receive messages specified for automation through MPF. (YES indicates that the console can receive messages eligible for automation.)

Notes:

1. Using the KEY name, operators can display information on the DISPLAY CONSOLES,KEY command for all extended MCS consoles defined with the same key.
2. Using the AUTO keyword, you can define an extended MCS console to receive messages that MPF indicates as eligible for automation. These messages can originate on any system in the sysplex. By specifying AUTO(YES) and MSCOPE(*ALL) or the MCSOPER OPERPARM equivalents, an extended MCS console can receive these messages from all systems in the sysplex.
3. Altering some console attributes might cause a message loss, UD loss, or SYNCHDEST loss. If a loss occurs, MVS issues a DISPLAY CONSOLE,HCONLY command and message IEE889I. You need to understand that this can happen and can affect automation.

The potential for this situation to occur comes from using these commands:

```
VARY CN
VARY CONSOLE
CONTROL V,LEVEL
SWITCH CN
LOGOFF (for SMCS consoles)
```

MCSOPER and OPERPARM

You can use MCSOPER to specify OPERPARM values for the extended MCS console. MCSOPER OPERPARM parameter list fields correspond to the RACF OPERPARM subkeywords in Table 1 on page 7. These MCSOPER values override RACF OPERPARM values for an extended MCS console.

For information on MCSOPER OPERPARM, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

References

For information about using the TSO/E CONSOLE command for TSO/E users of extended MCS consoles, see *z/OS TSO/E System Programming Command Reference*.

For information on writing MVS application programs that use extended MCS consoles, see *z/OS MVS Programming: Authorized Assembler Services Guide* and *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU*. For REXX language programs, see *z/OS TSO/E REXX User's Guide* and *z/OS TSO/E REXX Reference*.

SDSF and MVS Operations Planning

SDSF is a program that runs on TSO/E and uses Interactive System Productivity Facility (ISPF) panels. With SDSF, you can:

- Display immediate, up-to-date information about the jobs submitted to JES2 for processing, including:
 - Jobs on the JES2 input, output, and held queues
 - Job status of a specific job, including the job's priority and input class, the time and date the job was entered in the system, and the time and date the system began processing the job
 - System information about active jobs
 - Spool data sets for a specific job
 - Output from a job
- Monitor and control jobs, output, and resources in a JES2 system without using JES2 command syntax. With the MAS panel, you can display and control members in a JES2 multi-access spool.
- Enter MVS and JES2 system commands from any TSO/E terminal.
- View the system log (SYSLOG), operations log (OPERLOG), or user log (ULOG) online and search for specific information, which can reduce problem management time and eliminate the need for a printed copy of the log.
- View input data sets of jobs that are being processed or waiting to be processed.
- View output data sets online and purge them, which can reduce the system print load.
- Control remote printers and schedule output to be printed at remote printers.
- Get online information: help for panels, commands, and messages; an interactive tutorial for ISPF users; and online documentation through BookManager.

RMF and MVS Operations Planning

Resource Measurement Facility provides data for performance measurements, capacity planning, and trouble shooting. RMF can display information at the touch of a button and provides functions to archive collected data for future reports and analysis.

The functions RMF offers ensure the manageability of parallel microprocessor systems. They assist in performance management without the need to logon to every system where data is collected, and they support the new concept of managing workloads by MVS through service level reporting.

With RMF, you can monitor the performance of the whole system complex from a single point of control, thus increasing user productivity:

- Sysplex performance reports
- Selectable single-system reports in the sysplex
- Sysplex data server to access data across the sysplex

RMF provides performance data about business-oriented workloads and assists in managing service levels efficiently. In addition, you get performance information for CICS and IMS subsystems.

Coupling technology in the sysplex makes high-performance data sharing possible and can increase the manageability of your whole environment. RMF provides the data necessary for planning of the coupling facility configuration.

For information on RMF, see *z/OS RMF User's Guide*.

Automated Operations and z/OS Operations Planning

As part of planning z/OS operations, consider using automated operations at your installation. Automated operations help simplify operator tasks.

NetView

NetView selects messages that you can specify through the MVS message processing facility (MPF) and uses its own message automation functions to help automate operations tasks. Using MPF, you can suppress large numbers of messages that operators do not need to see or select messages that NetView can use to automate MVS tasks. (For information about using MPF to process messages, see “MPF and MVS Operations Planning” on page 114.)

The NetView console, which is attached to NetView on an MVS system, allows operators to perform many tasks that they ordinarily perform on MCS or SMCS consoles. On the NetView console, you can display MVS messages, highlight and hold important messages as on an MCS or SMCS console, and enter MVS commands. The NetView console also allows operators to define NetView command lists. These command lists can respond to messages selected through MPF on MVS and perform a series of command operations that simplify operator console actions. You can also route messages to a NetView console. You can select certain messages to be directed to a specific console for operator action.

NetView consoles allow your operators to enter MVS commands to do work on behalf of MVS. Your operators can also use MCS or SMCS consoles to enter NetView commands. Thus, operators can invoke NetView command lists from MCS or SMCS consoles to accomplish NetView tasks.

NetView Automation: Planning can help you coordinate activities for your MCS or SMCS consoles, NetView, and MPF.

Automated Operations Control/MVS

System Automation for OS/390 is a NetView application that automates console operations in a z/OS or OS/390 environment. System Automation for OS/390 uses the message handling capabilities of MVS and NetView to initiate automation procedures. These automation procedures perform operator functions that manage MVS, JES2 or JES3, and program products like VTAM, RMF, and TSO/E.

System Automation for OS/390 General Information can help you plan console automation using NetView.

Operations Planning and Control/ESA

NetView and System Automation for OS/390 can help you plan automated operations for z/OS or OS/390 systems and networks and can simplify the tasks operators need to perform. Automating production workload processing, including batch processing, can also simplify operations and improve the workload management at your installation. IBM's program product IBM SAA Operations Planning and Control/ESA (OPC/ESA) can help you plan your MVS production workload. It plans and schedules workload processing and monitors and controls the flow of work through your MVS environment.

OPC/ESA General Information can help you plan the automation of your production workload processing using OPC/ESA.

Remote Operations and MVS Operations Planning

If your installation is managing target systems from host systems, you need to consider how these remote operations tasks can affect your operations planning.

The Target System Control Facility (TSCF) is now part of System Automation for OS/390 and uses NetView to allow a host z/OS or OS/390 system to automate operations at target systems. Using System Automation for OS/390, you can automate console functions remotely, like IPLing or power-on restarting a processor. You can initialize or monitor target systems, or use a passthrough facility to enter processor commands for target systems, and let NetView operators manage several target systems simultaneously from a host system.

In multisystem environments where remote operations is a goal, System Automation for OS/390 and NetView provide a good way to manage operations. *System Automation for OS/390* describes the planning tasks for managing remote operations of systems using System Automation for OS/390 and NetView.

ESCON and Operations Planning

ESCON Manager (Enterprise Systems Connection Manager) is now part of System Automation for OS/390 and permits implementation of configuration changes among channels, ESCON Directors, control units, and devices. System Automation for OS/390 can be used to control and display the entire I/O configuration, whether it be ESCON or non-ESCON or switched (via ESCON Directors) or non-switched. System Automation for OS/390 ensures that a change to the I/O configuration will not unexpectedly cause system or application outages due to the loss of a connection path that is in use.

System Automation for OS/390 runs in z/OS or OS/390 environments providing:

- A single, logical point of control of I/O for multiple systems
- A unified multisystem view of I/O configuration and resource information
- Ability to vary online and vary offline devices attached to ESCON or parallel channels
- Support for coupling facilities

These enhancements significantly increase the effectiveness of managing and controlling I/O resources resulting in improved availability of computing resources and increased efficiency in doing problem determination.

Reference

For more information see *System Automation for OS/390 General Information*.

Chapter 2. Defining Your Console Configuration

An MVS console configuration consists of the various consoles that operators use to communicate with MVS. Your installation first defines the I/O devices it can use as MCS consoles with the hardware configuration definition (HCD). HCD manages the I/O configuration for the MVS system. You do not use HCD to define an SMCS console. To indicate to MVS which devices to use as MCS consoles, you specify the appropriate devices in the CONSOLxx parmlib member.

Choosing How to Define Your Console Configuration

CONSOLxx lets you define MCS consoles, SMCS consoles, or subsystem-allocatable consoles.

Subsystem-allocatable consoles are defined to a subsystem such as NetView, which manages the console for the system. For an MCS or SMCS console, CONSOLxx allows you to define various console attributes that control how operators can use the console and also control message routing and command processing for the console. For subsystem-allocatable consoles, you control console functions through the subsystem. It is beneficial to use an extended MCS console interface (when available) instead of a subsystem-allocatable console interface to relieve the 99 console constraint.

How you define your console configuration depends on the MVS system environment at your installation. For a single MVS system, you might want to consolidate console functions using NetView. A single NetView console instead of several MCS consoles can serve as the focal point for MVS operator actions and for NetView automation tasks. An operator can handle many operational needs of the system from this one NetView console. For information on using NetView consoles, see *NetView Automation: Planning*.

For an MVS system that manages many system resources or subsystems, you might want to use several MCS consoles, each assigned with different functions. For example, defining a console cluster for a system can help your installation divide its console functions more efficiently. A console cluster is a group of several MCS or SMCS consoles located together that you can use in place of a single console to divide up the functions and message traffic of the single console. "Managing Messages with a Console Cluster" on page 173 shows how to set up a console cluster for an MVS system.

If your MVS system requires increased security, your installation can use RACF to control console logon and the commands that an operator can enter from a specific console. It is especially important to use RACF to control access to SMCS consoles and the commands they can issue. Using RACF with MCS or SMCS consoles in an MVS system or sysplex can ensure that operators enter only the commands they are authorized to use.

In a sysplex, centralizing and coordinating console functions among different systems is an important operations goal. Message traffic and command routing are two considerations when you define consoles for a sysplex. In a sysplex, operators can receive messages from different systems on a single console, or can enter commands from a console to affect the processing of another system. How you define console functions for each MVS system can affect the operations of the sysplex as a whole. As a result, you need to understand the operations of the sysplex and plan the console configuration for each MVS system accordingly.

This chapter describes how to set up an MCS and SMCS console configuration for an MVS system using the CONSOLxx parmlib member. It describes how to define devices as consoles to MVS and how to define console functions to plan for console recovery and security. It also describes how to define console functions that help operators manage messages on their console screens and enter commands from their keyboards. Finally, it describes how you can define hardcopy processing to handle your MVS system recording. Because consoles in a sysplex present special cases, the chapter also includes planning considerations for defining and using consoles in a sysplex environment.

Shared Consoles in Mixed Sysplexes

If your sysplex has systems of different levels, the systems can share consoles.

Using CONSOLxx

Reference

For complete information about CONSOLxx and any Parmlib member in this book, see *z/OS MVS Initialization and Tuning Reference*. It provides reference information, options, and values that you can specify for CONSOLxx and other Parmlib members. When you define your console configuration for MVS, use that book as reference to code your members.

To define your MCS and SMCS console configuration, you use the following Parmlib members:

- CONSOLxx, which defines console characteristics for each MCS or SMCS console.
- CNGRPxx, which defines alternate console groups to use for recovery. (For information on CNGRPxx, see “Planning Console Recovery” on page 43.)
- PFKTABxx, which contains the program function key PFK tables for all MCS and SMCS consoles. (For information on PFKTABxx, see “Defining PFKs and Other Command Controls for Consoles” on page 85.)
- MPFLSTxx, which defines message processing to retain, suppress, or modify messages and commands. (For information on MPFLSTxx, see “MPF and MVS Operations Planning” on page 114.)

CONSOLxx lets you define certain devices as consoles and specify attributes that determine how your operators can use MCS or SMCS consoles.

CONSOLxx contains four statements that define and control consoles for an MVS system:

- CONSOLE
- INIT
- DEFAULT
- HARDCOPY

See “SMCS Console Considerations” on page 26 for specific information about defining SMCS consoles.

CONSOLE Statement

You use the CONSOLE statement to define a device as a console. You define each console device with one CONSOLE statement. CONSOLE also lets you specify console attributes that control the following for an MCS or SMCS console:

- Console recovery by assigning an alternate console or a group of consoles to act as alternates to the console specified on the statement. (You define groups of alternate consoles in the CNGRPxx Parmlib member.)
- Console security by assigning command authority levels
- Certain console screen functions (console mode, methods for deleting messages from the screen, ways to control display areas on the screen, and how to set up the PFKs for the console)
- Message routing and message formatting
- Console operation in a sysplex

Table 2 summarizes the console functions that you control using the CONSOLE statement. It includes the CONSOLE keyword and the MVS command to change the keyword value. If an MVS command cannot be used, the table indicates that you must re-IPL to change the value. The table also includes a page reference in the book for a description of each keyword.

Table 2. Summary of CONSOLE statement functions

Task	CONSOLE statement keyword	MVS command to change value	See page
Defining a device as a console:			
• Device number or system console name (SYSCONS)	DEVNUM	Must re-IPL	38
• Console name	NAME	Must re-IPL	38
• Kind of device	UNIT	Must re-IPL	38
• The VTAM logical unit (LU) name (SMCS console only)	LU	VARY CN,LU	31
Planning console recovery:			
• Device number or name for a single alternate or backup console	ALTGRP	VARY CN,ALTGRP	43
• Name of the alternate console group	ALTERNATE	VARY devnum, CONSOLE,ALTCONS	43
Planning console security:			
Command authority level for the console	AUTH	VARY CN,AUTH	52
Override LOGON value on default statement	LOGON	VARY CN,LOGON	54
Controlling the console screen function:			
• Input/output capability or console mode	USE	CONTROL V,USE	71
• Message deletion mode of the console	DEL	CONTROL S,DEL	72
• Number of message lines that roll on the console screen	RNUM	CONTROL S,RNUM	73

Table 2. Summary of CONSOLE statement functions (continued)

Task	CONSOLE statement keyword	MVS command to change value	See page
• Number of seconds between message rolls or wraps	RTME	CONTROL S,RTME	73
• Conversational/ nonconversational message deletion	CON	CONTROL S,CON	77
• Number of lines to be deleted from the console screen using CONTROL E,SEG	SEG	CONTROL S,SEG	79
• Defining status display areas of the console screen	AREA	CONTROL A	80
• Time interval for updating dynamic status display areas of the console screen	UTME	CONTROL T,UTME	82
• Routing instructions for status displays	MSGRT	MSGRT	81
• Monitoring selected events	MONITOR	MONITOR	84
• Defining a PFK table for the console	PFKTAB	CONTROL N,PFK	85
Controlling message routing and message formatting:			
• Routing codes for the console	ROUTCODE	VARY CN,ROUT VARY CN,AROUT VARY CN,DROUT	105
• Message levels for the console	LEVEL	CONTROL V,LEVEL	106
• Routing of undelivered messages	UD	VARY CN,UD	82
• Message formats for console display	MFORM	CONTROL S,MFORM	107
Controlling console operation in a sysplex:			
• System scope for messages that the console receives	MSCOPE	VARY CN,MSCOPE VARY CN,AMSCOPE VARY CN,DMSCOPE	108
• System association for commands entered	CMDSYS	CONTROL V,CMDSYS	109
• Specifying the system where you want the console to be active	SYSTEM	VARY CN,SYSTEM	42

INIT, DEFAULT, and HARDCOPY Statements

INIT, DEFAULT, and HARDCOPY statements define general characteristics for all MCS and SMCS consoles in the system or sysplex.

The INIT statement

You use the INIT statement to control basic initialization values for all MCS or SMCS consoles in the configuration. INIT lets you control the following:

- Console recovery (activating the CNGRPxx member that contains alternate console group definitions and specifying a console group of master console candidates)
- Certain console screen functions for all consoles (activating the PFKTABxx member to control the PFK tables for MCS and SMCS consoles, displaying certain information for mount messages, and specifying the command delimiter for operator input of multiple commands)
- Message processing (such as activating MPF, AMRF, and the IEAVMXIT message processing exit; and controlling WTO and WTOR messages, the hardcopy message set, and MMS for message translation).
- The SMCS VTAM application for controlling SMCS consoles.

After IPL, operators can use the SET command to change some values defined on the INIT statement. See Table 3, which summarizes console functions that you control on the INIT statement:

Table 3. Summary of INIT statement functions

Task			
Console function or attribute	INIT statement keyword	MVS command to change value	See page
Planning console recovery:			
• Activating the CNGRPxx member that contains alternate console group definitions	CNGRP	SET CNGRP	44
• Specifying the name of an alternate console group of master console candidates	NOCCGRP	Activate another CNGRPxx member (SET CNGRP) that defines the same alternate console group but with different console members. See “Changing the Specification of Alternate Console Groups” on page 46.	49
Controlling the console screen function:			
• Display of certain information for mount messages	MONITOR	MONITOR	85
• PFKTABxx member that contains PFK tables for consoles	PFK	SET PFK	85
• Defining the command delimiter for multiple command input	CMDDELIM	Must re-IPL	88
• Specifying the VTAM APPLID that SMCS is to use on this system	APPLID	CONTROL M,APPLID	30
• Specifying the VTAM GENERIC resource name that SMCS is to use for the sysplex	GENERIC	CONTROL M,GENERIC	30
Controlling message processing:			
• Activating the message processing facility (MPF)	MPF	SET MPF	114
• Activating the action message retention facility	AMRF	CONTROL M,AMRF	116

Table 3. Summary of INIT statement functions (continued)

Task			
Console function or attribute	INIT statement keyword	MVS command to change value	See page
• Activating the IEAVMXIT message processing exit	UEXIT	CONTROL M,UEXIT	121
• Maximum number of WTO buffers	MLIM	CONTROL M,MLIM	123
• Maximum number of WTOR buffers	RLIM	CONTROL M,RLIM	123
• Maximum number of write-to-log (WTL) buffers	LOGLIM	CONTROL M,LOGLIM	128
• Activating the MVS message service (MMS) for message translation	MMS	SET MMS	128
• In a sysplex, controlling the aggregation of messages returned by the ROUTE *ALL or ROUTE <i>systemgroupname</i> command	ROUETIME	CONTROL M,ROUETIME	125
Controlling component tracing options			
Specifying the Parmlib member that contains tracing options for the operations services (OPS) component	CTRACE	TRACE CT	OS/390 MVS Initialization and Tuning Reference

The DEFAULT statement

You use the DEFAULT statement to control certain default values for MCS and SMCS consoles in the configuration. DEFAULT lets you specify console attributes that control the following for an MCS and SMCS console configuration:

- Console security by specifying operator logon options
- Certain console screen functions for all consoles (ability for operators to hold messages on the screen)
- Routing for messages without routing codes or other message queuing information, and routing for synchronous messages that bypass normal message queuing
- Determining the maximum value for operator REPLY ids.

Unlike values in CONSOLE and INIT, operators cannot change individual DEFAULT statement values. Operators must re-IPL the system, or, in some cases, the sysplex, with the CONSOLxx member that contains the new DEFAULT statement.

Table 4 on page 19 summarizes console functions that you can control using the DEFAULT statement:

Table 4. Summary of DEFAULT statement functions

Task	DEFAULT statement keyword	MVS command to change value	See page
Controlling console security:			
• Operator logon to MCS and SMCS consoles	LOGON	Must re-IPL	54
Controlling the console screen function:			
• Freezing the display of messages on MCS or SMCS console screens	HOLDMODE	Must re-IPL	76
Controlling message routing:			
• Assigning routing codes for messages without any specified target	ROUTCODE	Must re-IPL	105
• Assigning the name of an alternate console group to receive synchronous messages	SYNCHDEST	Activate another CNGRPxx member (SET CNGRP) that defines the same alternate console group but with different console members. See "Changing the Specification of Alternate Console Groups" on page 46.	51
Controlling message processing:			
• Maximum number of REPLY ids	RMAX	CONTROL M,RMAX	123

The HARDCOPY statement

You can use the optional HARDCOPY statement to define the characteristics of the hardcopy message set and specify the hardcopy medium. You can control how to record messages and commands for the system. After IPL, operators can use the VARY command to do the following:

- Change the set of messages included in the hardcopy message set
- Assign either SYSLOG, OPERLOG, or an MCS printer as the hardcopy medium
- Stop a specific hardcopy medium provided another one is active.

For information about using the VARY command, see *z/OS MVS System Commands*.

For information about hardcopy processing, see "Hardcopy Processing" on page 88.

Table 5 summarizes console functions you can control using the HARDCOPY statement:

Table 5. Summary of HARDCOPY statement functions

Task	HARDCOPY statement keyword	MVS command to change value	See page
Controlling logging and system recording:			
• Device number for hardcopy device, OPERLOG, or SYSLOG	DEVNUM	VARY devnum,HARDCPY	88

Table 5. Summary of HARDCOPY statement functions (continued)

Task	HARDCOPY statement keyword	MVS command to change value	See page
• Routing codes for the hardcopy message set	ROUTCODE	VARY devnum,HARDCPY,AROUT VARY devnum,HARDCPY,DROUT	88
• Hardcopy of commands by level	CMDLEVEL	VARY devnum,HARDCPY,cmdlevel	88
• Assigning an alternate console group of devices as backup logging devices	HCPYGRP	Activate another CNGRPxx member (SET CNGRP) that defines the same alternate console group but with different console members.	46
• Controlling the recording of undelivered action messages, WTOR messages, and important informational messages	UD	VARY ,HARDCPY,UD	88
• Defining 4-digit year	HCFORMAT	None	88

CONSOLxx and the Sysplex

When the operator initializes an MVS system with CONSOLxx, the console definitions and attributes are in effect for the system. MCS and SMCS consoles defined by CONSOLE statements are active, and the values specified for INIT, DEFAULT, and HARDCOPY control console operations for the system. Operators can use the CONTROL, MONITOR, MSGRT, SET, and VARY commands to change many of the definitions after the system is active; the effect of the command, however, is temporary. In a single system, changes made with commands last only for the duration of the IPL. In a sysplex, changes made with commands last for the life of the sysplex; that is, the changes remain in effect until every system in the sysplex is shut down at the same time.

In a sysplex, certain CONSOLxx keywords have **sysplex scope**. When a system with those keywords is first IPLed into a sysplex, the keyword values are in effect for the entire sysplex.

For example, NAME and ROUTCODE on the CONSOLE statement have sysplex scope. NAME specifies a unique name that identifies the console within the sysplex; ROUTCODE defines the routing codes for messages that the console is able to receive from all systems in the sysplex.

For INIT and DEFAULT keywords that have sysplex scope, CONSOLxx for the first system that joins the sysplex determines the values in effect for all systems in the sysplex. When other systems join the sysplex, MVS ignores changes to keyword values with sysplex scope defined in CONSOLxx for those systems. For example, if the action message retention facility (AMRF) is active in CONSOLxx for the first system that joins the sysplex, the sysplex ignores the AMRF keyword specified for other systems that join, and the action message retention facility is active for all systems in the sysplex.

CONSOLxx keywords that have **system scope** apply only to the system on which they are defined. For example, DEVNUM, UNIT, and PFKTAB for CONSOLE and all keywords for HARDCOPY have system scope. The device number (DEVNUM),

device type (UNIT), and PFK table (PFKTAB) for the console apply only to the system where the console is attached. Similarly, the hardcopy log specifications for HARDCOPY apply only to the local system where CONSOLxx is defined.

Table 6 summarizes which keywords on each CONSOLxx statement are system or sysplex in scope:

Table 6. Scope of CONSOLxx Keywords

CONSOLxx statement	System scope	Sysplex scope
CONSOLE	DEVNUM UNIT PFKTAB	NAME ALTERNATE ALTGRP AUTH USE CON SEG DEL RNUM RTME AREA UTME MSGRT MONITOR ROUTCODE LEVEL MFORM UD MSCOPE CMDSYS SYSTEM LOGON LU
INIT	PFK MONITOR CMDDELIM MPF UEXIT MMS MLIM LOGLIM NOCCGRP APPLID	AMRF RLIM CNGRP ROUETIME GENERIC
DEFAULT	LOGON HOLDMODE ROUTCODE SYNCHDEST	RMAX
HARDCOPY	All keywords	

Understanding the scope of CONSOLxx keywords is important when you plan your console configuration for a sysplex. Depending on the needs of your installation and the scope of CONSOLxx keywords, you can specify CONSOLxx for systems in a sysplex in different ways. Consider the following ways to define CONSOLxx in a sysplex:

1. Share a single CONSOLxx member for all systems.
2. Use unique CONSOLxx members for each system.

- Use unique CONSOLxx members for each system, but define all consoles in the CONSOLxx member of the first system to join the sysplex.

The method you choose depends on how you want to use the console device numbers. If you want to define a console with the same device number on two different systems, the consoles must have different names. Therefore, if you use the same device numbers for consoles across the sysplex, you must use option 2 on page 21, or option 1 with symbolics. If the sysplex requires unique console device numbers, you can use any of the methods.

The following sections explain the ways to define CONSOLxx in a sysplex in detail.

Sharing a Single CONSOLxx Member for All Systems

Sharing the same CONSOLxx for all systems in the sysplex provides a single, consistent set of console definitions, as if you are defining all your consoles for a single system.

In Figure 3, systems SYA and SYB share the same CONSOLxx member. SYA has three physically attached consoles (CON1, CON2, and CON3); SYB has two physically attached consoles (CON3 and CON4).

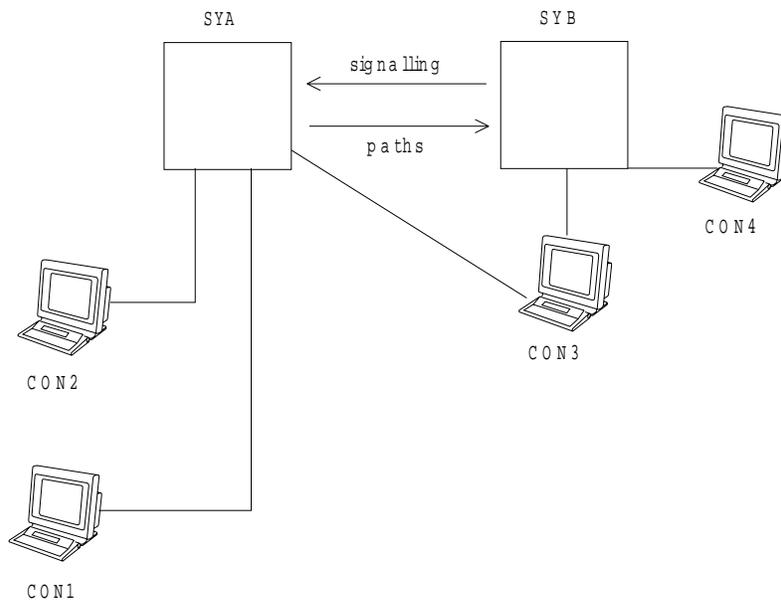


Figure 3. Console Configuration in a Sysplex with Two Systems and Four MCS Consoles

The following are statements from a CONSOLxx parmlib member that is shared by both SYA and SYB:

```

CONSOLxx for Both Systems
CONSOLE ... NAME(CON1) AUTH(MASTER)
CONSOLE ... NAME(CON2) AUTH(MASTER)
CONSOLE ... NAME(CON3) MSCOPE(SYA)
                    SYSTEM(SYB)
CONSOLE ... NAME(CON4) MSCOPE(SYB)
INIT AMRF(Y)

```

In this example:

- Values for INIT, DEFAULT, and HARDCOPY are the same across systems, and the order in which systems join the sysplex does not affect the sysplex environment.
- With names specified for all consoles in the configuration, MVS uses only four IDs and saves on the maximum number of console IDs (99) that MVS can assign in a sysplex. If you did not name the consoles in this example, MVS would use eight separate IDs (one ID for each console statement in the sysplex).
- A console can be active on only one system at a time. In Figure 3 on page 22, CON3 is physically attached to both SYA and SYB. Without specifying SYSTEM(SYB) for CON3, CON3 would become active on either SYA or SYB, whichever system joins the sysplex first. Specifying SYSTEM(SYB) ensures that CON3 is activated only on SYB.
- Because CON4 is not physically attached to SYA, it becomes active only when SYB joins the sysplex.

When two or more systems require unique values in a shared CONSOLxx member, you can use system symbols to represent those values. When each system processes CONSOLxx, the system replaces the system symbols with the substitution texts that it has defined to the system symbols.

For example, suppose you want to define names for two consoles on two different systems, and that the consoles are both at address X'3E0'. If both consoles to be active at the same time, they require different names. If you plan to use one CONSOLxx member for both systems, you can use system symbols to generate unique console names while retaining the same device number, as follows:

```
CONSOLE DEVNUM(3E0)
        NAME(C3E0S&SYSCLONE.) /* CONSOLE NAME "C3E0Snn" */
        ... /* Remaining CONSOLE keywords */
```

The console definition can then specify different names on different systems: For example, if your installation accepts the default substitution text for &SYSCLONE (the last two characters of the system name), the following console names result:

- C3E0SS1 on system SYS1
- C3E0SS2 on system SYS2
- C3E0SS3 on system SYS2

For more information about using system symbols in parmlib members, including lists of valid system symbols, see the section on sharing parmlib members in *z/OS MVS Initialization and Tuning Reference*.

Using Unique CONSOLxx Members for Each System

You can define separate CONSOLxx members for each system in the sysplex. Like Figure 3 on page 22, Figure 4 on page 24, shows SYA with three physically attached consoles (CON1, CON2, and CON3) and SYB with one physically attached console (CON4). Console statements are defined in two CONSOLxx members, one for each system in the sysplex.

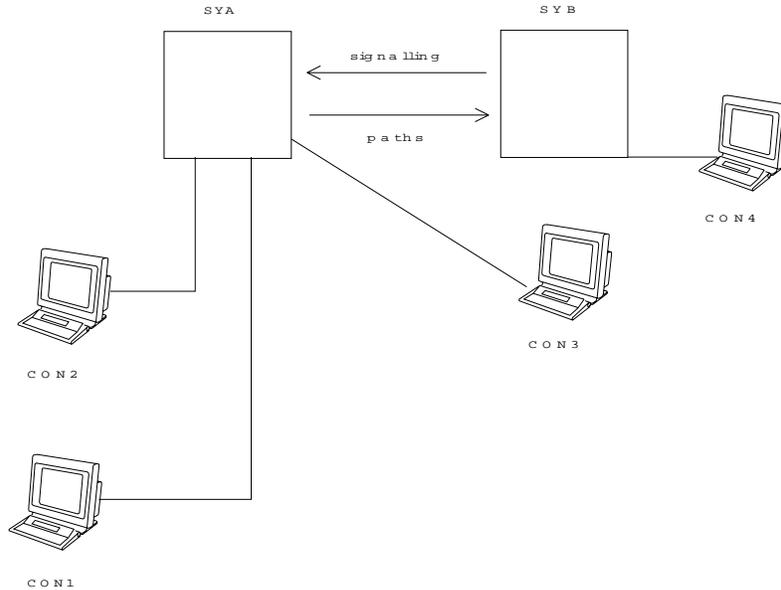


Figure 4. Console Configuration in a Sysplex with Two Systems and Four MCS Consoles

Note: In the examples that follow, the required CONSOLE keyword DEVNUM has been omitted.

The following are the CONSOLxx statements for each system in this configuration:

CONSOLxx for SYA	CONSOLxx for SYB
<pre> CONSOLE. . .NAME(CON1) AUTH(MASTER) CONSOLE. . .NAME(CON2) AUTH(MASTER) CONSOLE. . .NAME(CON3) MSCOPE(SYA) </pre>	<pre> CONSOLE. . .NAME(CON4) MSCOPE(SYB) INIT MPF(01) </pre>

This configuration provides great flexibility for consoles in the sysplex. You can define consoles based on the needs of each system. However, depending on when the systems join the sysplex, the scope of the CONSOLxx keywords can affect how the consoles operate in the sysplex.

For example, if an operator initializes SYA as the first system in the sysplex, CON1 is initialized as the master console. CON1 is the first console defined with master authority, specified as AUTH(MASTER) on the CONSOLE statement. The AUTH keyword has sysplex scope; depending on the order of the CONSOLE statements in CONSOLxx, the first system to join the sysplex determines how the master console is selected. When the first system joins the sysplex, the console defined with AUTH(MASTER) on the first CONSOLE statement becomes the master console.

In this configuration, CON2 has master authority. It can issue all MVS commands, but because it is the second CONSOLE defined in CONSOLxx for SYA, it is not the master console. Other consoles with master authority can join the sysplex, but as long as CON1 is active, it remains the master console.

Understanding how the master console is defined in a sysplex is important particularly in recovery situations, where the master console can serve as the

alternate for other consoles. “Planning Console Recovery” on page 43 describes recovery situations in a sysplex using the master console.

For CONSOLxx keywords with system scope, keyword values apply only to the system where the consoles are physically attached. For example, the MPF keyword in CONSOLxx for SYB indicates that MPFLST01 is active when SYB is initialized. However, because MPF has system scope, the default for MPF used on SYA indicates that SYA does not perform MPF message processing. In a sysplex that uses unique CONSOLxx members, it is therefore important to understand the scope of CONSOLxx keywords for each system.

Defining All Consoles in the CONSOLxx Member of the First System to Join the Sysplex

In Figure 5, all consoles are physically attached to SYA, and all consoles are defined in CONSOLxx for the first system that is to join the sysplex (which is SYA):

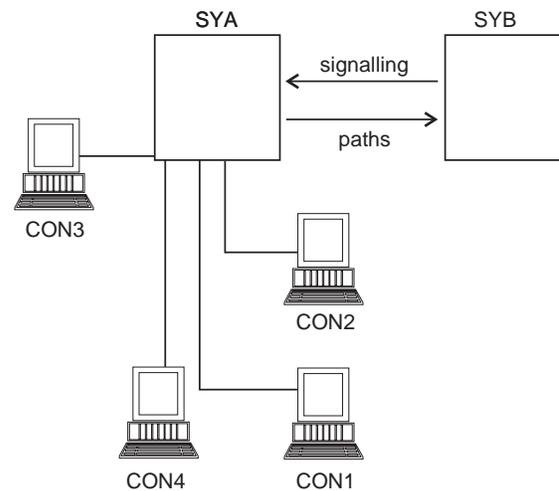


Figure 5. Console Configuration in a Sysplex with Four MCS Consoles Attached to One System

Although SYB joins the sysplex with a different INIT statement, its CONSOLxx member does not define additional MCS consoles.

The following are the CONSOLxx statements for each system in the configuration:

CONSOLxx for SYA	CONSOLxx for SYB
<pre> CONSOLE. . .NAME(CON1) AUTH(MASTER) CONSOLE. . .NAME(CON2) AUTH(MASTER) CONSOLE. . .NAME(CON3) MSCOPE(SYA) CONSOLE. . .NAME(CON4) MSCOPE(SYB) INIT AMRF(Y) DEFAULT HOLDMODE(YES) </pre>	<pre> INIT MPF(01) </pre>

The first system to join the sysplex (SYA) is the focal point of console operations for the sysplex configuration in Figure 5 on page 25. Thus, you are able to define all your MCS CONSOLE statements for the entire sysplex in one place, in this example CONSOLxx for SYA.

SYB uses an INIT statement with a specific MPF value that applies to that system. Because MPF has system scope, the value applies only to SYB.

If SYA in Figure 3 on page 22 fails, the sysplex is unable to use any MCS consoles because SYB does not have any CONSOLE statements defined in its CONSOLxx member. Using the system console and defining alternative consoles, such as SMCS consoles or extended MCS consoles, are ways to resolve the problem.

SMCS Console Considerations

SMCS consoles are MCS consoles that use VTAM services for input and output. SMCS consoles provide most of the same functions as MCS consoles with the following exceptions:

- Synchronous WTO/R, also known as disabled console communication facility (DCCF), is not supported for SMCS consoles. The system console or an MCS console must be used instead.
- SMCS consoles are not available during NIP. The system console or an MCS console must be used instead.
- VTAM must be active for SMCS to be active. The system console and MCS consoles do not rely on VTAM, and these can be used before VTAM is active.
- SMCS consoles must be activated differently than MCS consoles. The activation process depends on the console definitions, but in all cases, VARY CONSOLE and VARY CN, ONLINE do not work for SMCS.
- SMCS does not support output-only (message stream and status display) consoles. SMCS consoles must always be full-capability consoles.
- SMCS does not support printer consoles, and cannot be used as hardcopy devices.

Because an SMCS console is connected through a network and uses VTAM services, the VTAM commands VARY NET and HALT NET, as well as network problems, can affect console operations.

Installing SMCS

An SMCS console can be a real 3270 type device, but usually it will be a 3270 emulator such as IBM Personal Communications. SMCS supports VTAM LU Type 0 or Type 2, and SMCS consoles must support Extended Data Stream and the Read Partition Query function.

Installing SMCS consoles requires some VTAM Definitions:

- Define the SMCS application.
- Create a LOGON mode table (optional).
- Indicate that certain LUs are always to be used for SMCS (optional).

CONSOLxx also requires some changes:

- Specify that the SMCS application is to be started.
- Define some SMCS consoles.

Finally, RACF requires some definitions:

- Userids for operators.
- Command authority.

Defining SMCS to VTAM

To define the SMCS application to VTAM, you must update the ATCCONxx member of SYS1.VTAMLST to point to a member of SYS1.VTAMLST that defines the SMCS application id (APPLID). You could write the SMCS application definition as:

```
SMCS      VBUILD TYPE=APPL
SMCS&SYSCLONE. APPL
```

You can also choose to specify DLOGMOD and MODETAB, but you should take defaults for all other keywords. Each system within the sysplex that will run SMCS must have a unique application name. See *z/OS Communications Server: SNA Resource Definition Reference* for more details.

A LOGON mode table can be provided to define the session protocols for devices that will be used as SMCS consoles. Each LOGON mode table is assembled and link-edited into SYS1.VTAMLIB. In most cases, the same LOGON mode table that is used for TSO will be suitable for SMCS.

The DLOGMOD and/or MODETAB specifications indicate which LOGON mode table to use. The specifications can be made on:

- The APPL statement pointed to by ATCCONxx.
- The LOCAL statement when defining local non-SNA major nodes.
- The LU statement when defining SNA major nodes.

See *z/OS Communications Server: SNA Resource Definition Reference* for details.

If certain devices are always used for SMCS, they can be defined to automatically log on to the SMCS application when the device becomes active using the LOGAPPL keyword on the LOCAL or LU statements:

```
LOCALDEV LBUILD
S&SYSCLONE.D3E0 LOCAL CUADDR=3E0
                TERM=3277,
                FEATUR2=(MODEL2),
                ISTATUS=ACTIVE,
                USSTAB=USSCNH,
                DLOGMOD=S3270,
                LOGAPPL=SMCS&SYSCLONE
```

Figure 6 on page 28 shows a sample LOGON mode table entry. Table 7 on page 28 defines the keywords in the LOGON mode table entry, and the values that SMCS expects. Table 8 on page 28 provides details about the values to specify for the PSERVIC keyword.

```

*****
*
*      DYNAMIC LOGMODE ENTRY FOR SNA
*      3270 DISPLAYS (APPLIES TO QUERABLE TERMINALS)
*
*****
DYN SNA  MODEENT LOGMODE=DYN SNA,COS=INTERACT,APPNCOS=#INTER,
          FM PROF=X'03',
          TS PROF=X'03',
          PRIPROT=X'B1',
          SEC PROT=X'90',
          COM PROT=X'3080',
          RUSIZES=X'87F8', * OUTBOUND 3840    INBOUND 1024
          TYPE=1,
          PSERVIC=X'028000000000000000000000300'

```

Figure 6. Sample LOGON Mode Table Entry

Table 7. Keyword Definitions

Keyword	Definition	Local Non-SNA Value	SNA Value
FM PROF	Function Management Profile	X'02'	X'03'
TS PROF	Transmission Services Profile	X'02'	X'03'
PRIPROT	Primary LU Protocol	X'71'	X'B1'
SEC PROT	Secondary LU Protocol	X'40'	X'90'
COM PROT	Common LU Protocol	X'2000'	X'3080'
RUSIZES	Maximum length of data in a request unit	X'0000'	X'87F8' The X'87' indicates a 1024-byte maximum secondary logical unit RU send size and the X'F8' indicates a 3840-byte maximum primary logical unit RU send size.
TYPE	Bind type	1	1
PSERVIC	LU Presentation Services Profile	X'0080000000000185000000300'	Value depends on the device type. See Table 8 for values.

Table 8. PSERVIC Values for SNA Devices

Byte	Value	Definition
1	X'00' or X'02'	LU type 0 or LU type 2. LU0 indicates that the session protocol is determined by the application. SMCS will use LU0 for non-SNA locally attached 3270 data stream devices. LU2 indicates that the session protocol is for an SNA 3270 data stream device. SMCS will use this for SNA locally or remotely attached devices.

Table 8. PSERVIC Values for SNA Devices (continued)

Byte	Value	Definition
2	X'80'	Indicates that query is supported. This is the recommended value for byte 2 whenever possible. If X'00' is specified, the alternate screen size may be required depending on the presentation space size indication.
3,4,5,6	0	These must be zero.
7,8	X'0000'	Screen size when in default presentation space size (24 rows x 80 columns).
9,10	Possible values: <ul style="list-style-type: none"> • X'0000' • X'1850' • X'1B84' • X'2050' • X'2B50' 	Screen size when in alternate presentation space size. This value depends on the specification in byte 11 and the device type to be used as an SMCS console. Byte 9 (number of rows) is limited to 8 (X'08') through 255 (X'FF'). Byte 10 (number of columns) is limited to 80 (X'50') through 255 (X'FF'). The product of bytes 9 and 10 (rows * columns) must be less than or equal to 16,383. If both bytes 9 and 10 are zero, the screen size is determined by querying the device. Possible values are: <ul style="list-style-type: none"> • X'0000' Screen size determined by querying the device. Byte 11 contains an X'03'. This is the recommended value for bytes 9 and 10. • X'1850' 24 rows by 80 columns. Byte 11 contains a X'02', X'7E' or X'7F'. • X'1B84' 27 rows by 132 columns. Byte 11 contains a X'7F'. • X'2050' 32 rows by 80 columns. Byte 11 contains a X'7F'. • X'2B50' 43 rows by 80 columns. Byte 11 contains a X'7F'.
11		Indicates which screen size should be used. Supported values are: <ul style="list-style-type: none"> • X'02' - Screen size is always 24 rows by 80 columns • X'03' - Default presentation space is 24 x 80 and the alternate presentation space is specified in the Query Reply. This is the recommended value. If this value is specified, byte 2 must contain a X'80'. • X'7E' - The default screen size is to be used. • X'7F' - The alternate screen size is to be used.
12	0	Must be zero.

For more information, see *z/OS Communications Server: SNA Resource Definition Reference*.

Updating CONSOLxx

To indicate that the SMCS application is to be started, you must define the SMCS APPLID on the INIT statement of CONSOLxx:

INIT APPLID(SMCS01)

If you omit APPLID, SMCS will not be available for the life of the system. You can change the APPLID after the system is active, but only when an APPLID was specified in CONSOLxx during IPL.

SMCS also supports the use of VTAM generic resource. VTAM generic resource names allow an operator who logs on to be connected to the system that VTAM selects rather than being connected to a specific system. Specifying GENERIC in CONSOLxx provides flexibility and promotes effective recovery from problems. Specifying a specific system when logging on, in contrast, is sometimes necessary when a particular operator requires affinity to facilities available on a specific system. When you identify a specific system, make sure that the message scope you define in CONSOLxx matches the system you identify.

VTAM has the following requirements for using generic resource names:

- The system must be part of a Parallel Sysplex (PLEXCFG=MULTISYSTEM), and it must have a coupling facility.
- The coupling facility must have the generic resource structure defined. The default name of the structure is ISTGENERIC.
- VTAM must be an APPN node.

SMCS consoles must be defined in CONSOLxx, using the CONSOLE statement. With a few exceptions, any keywords and values that you can specify for a MCS console can also be specified for a SMCS console.

SMCS adds a value for the DEVNUM keyword and the LU and LOGON keywords.

SMCS and MCS console definitions can be mixed in the same CONSOLxx. Both types of consoles can coexist within the same system, as well as within a sysplex. An example of a SMCS console definition follows:

```
CONSOLE DEVNUM(SMCS)
        NAME(CON1)
        ALTGRP(GROUP1)
        AUTH(MASTER)
        LOGON(REQUIRED)
        LU(S01LU24)
        RNUM(20)
        RTME(1/4)
```

SMCS consoles are not associated with a particular system. A SMCS console defined on one system can be activated on another system, provided that the SMCS application is active on both systems.

Starting the SMCS Application

The SMCS application is designed to start, and restart, automatically. The SMCS application will attempt to connect to VTAM using the SMCS APPLID every 15 seconds. If the APPLID is deactivated, the SMCS application will attempt to restart (reconnect to VTAM using the SMCS APPLID) every 15 seconds.

The SMCS APPLID must be active before SMCS can use it. Normally, the APPLID will be defined to be active once VTAM starts. If there is a need to deactivate the SMCS APPLID, enter the following:

```
VARY NET,INACT,ID=applid[,I or ,F]
```

This command will cause the SMCS application to stop, deactivate all consoles connected to the specified APPLID, and cause the SMCS application to try to reconnect every 15 seconds.

There are some functions that require you to deactivate and reactivate the SMCS APPLID, called 'recycling the APPLID'.

Changing APPLIDs: It may be necessary to change the SMCS APPLID for a system. The following command will change the APPLID.

```
K M,APPLID=applid
```

SMCS will continue to use the old APPLID until it is deactivated with the VARY NET,INACT command. Once the old APPLID is deactivated, the new one may need to be activated using the V NET,ACT command. During the time that the old APPLID is still in use, message IEE821E will be issued as a reminder that SMCS needs to be recycled on that system. You can issue D C,SMCS to verify your actions.

The new APPLID is only in effect for the life of the system. CONSOLxx will need to be updated to use the new APPLID on the next IPL.

Using VTAM Generic Resource Names: Use of generic resources is optional. If you use generic resources, specify GENERIC on the INIT statement. When you specify GENERIC, you supply one generic name for the entire sysplex. You specify the name on the INIT statement:

```
INIT APPLID(SMCS01) GENERIC(SMCSGENR)
```

Like APPLID, GENERIC can be changed after the system is active. Unlike APPLID, if GENERIC is not specified in CONSOLxx, you can add GENERIC later.

For more information about VTAM generic resources, see *z/OS Communications Server: SNA Resource Definition Reference*.

Changing GENERICS: The operator can change the SMCS GENERIC that is in use by the sysplex using the following command:

```
K M,GENERIC=generic
```

The operator can also turn off the SMCS GENERIC by using:

```
K M,GENERIC=*NONE*
```

Each SMCS application in the sysplex will continue to use the old GENERIC until that SMCS application is recycled, using the V NET,INACT and V NET,ACT commands. Each SMCS can be recycled separately. Once each SMCS application is recycled, it will use the new GENERIC value, but any SMCS application that has not yet been recycled will continue to use the old GENERIC value. Therefore, it is possible to have some SMCS applications using the old GENERIC value and some using the new GENERIC value. You can issue D C,SMCS to verify your actions.

Message IEE820E will be issued as a reminder that an SMCS needs to be recycled and will remain outstanding until all SMCS applications are using the new GENERIC value.

Defining SMCS Consoles

The first parameter on the CONSOLE statement must be the DEVNUM parameter. SMCS consoles must specify DEVNUM (SMCS). All other parameters on the CONSOLE statement may be specified in any order. Do not specify the UNIT,

ALTERNATE, or SYSTEM parameters on the CONSOLE statement. Also, the only acceptable value for the USE keyword is FC.

SMCS consoles require the NAME parameter. If NAME is not specified, or is not valid, the CONSOLE statement is rejected. NAME is recommended for every CONSOLE statement in CONSOLxx, including subsystem consoles, but only SMCS consoles require it. Each console in the sysplex must have a unique name. System symbolics can be used in the name and throughout CONSOLxx so that one CONSOLxx member can be used for the entire sysplex.

SMCS does not support the use of alternate consoles; ALTGRP should be used instead, although it is optional. All consoles should have at least one console to switch to. For more information on defining an ALTGRP, see *z/OS MVS Initialization and Tuning Reference*.

SMCS consoles support the AUTH keyword in the same way that MCS consoles do. For a sysplex with MCS consoles or both MCS and SMCS consoles, the first MCS console listed in CONSOLxx with MASTER authority becomes the sysplex master console. For a sysplex with only SMCS consoles, the first console with MASTER authority that becomes active. Because it is impossible to predict which console will activate first, do not give MASTER authority to any console that cannot be the sysplex master console and use RACF to prevent the console from becoming the master console.

Predefined LU and LOGON: With predefined LU and LOGON, you can bypass the SMCS selection screen by indicating that a particular console name is always associated with a particular LU. Once the LU is logged on to the SMCS application, the console becomes active.

The LOGAPPL VTAM function indicates that a particular LU automatically logs on to a particular application when the LU becomes active. By indicating that a particular LU automatically logs on to the SMCS application with LOGAPPL, and indicating that the LU is associated with a particular console name with a predefined LU, a console can be activated automatically once VTAM is active, in much the same way that MCS consoles activate automatically during IPL.

The predefined LU allows a SMCS console to activate at one particular LU. To specify a predefined LU, specify the LU keyword on the CONSOLE statement. If a predefined LU is specified for a console, only that console can be activated at that LU. No other console can be activated at that LU, and that console can only be activated at that LU. The predefined LU can be changed later with the VARY CN command.

SMCS consoles also support the LOGON keyword on the CONSOLE statement. This keyword allows the console to override the LOGON value on the DEFAULT statement. However, some of the definition and operation of LOGON for SMCS is different than MCS and also depends on whether or not a predefined LU is specified.

If a predefined LU is specified, the LOGON definitions are the same as for MCS consoles:

- LOGON (OPTIONAL) indicates that the console does not need to be logged on.
- LOGON (AUTO) indicates that the console is automatically logged on.
- LOGON (REQUIRED) indicates that the console must be logged on before commands can be issued.

- LOGON (DEFAULT) indicates that the console is to use the LOGON value specified on the DEFAULT statement.
- If LOGON is not specified, the console also uses the LOGON value specified on the DEFAULT statement.

If a predefined LU is not specified:

- LOGON(OPTIONAL), LOGON(AUTO), LOGON(REQUIRED), and LOGON(DEFAULT) work the same as if a predefined LU was specified. LOGON(REQUIRED) is, however, strongly recommended.
- If LOGON is not specified, the console default is LOGON(REQUIRED). The console does not use the LOGON value specified on the DEFAULT statement.
- Regardless of whether a predefined LU is specified or not, LOGON is different for MCS and SMCS consoles. An MCS console always displays all messages that it receives; the console does not have to be logged on by an operator to receive messages. An SMCS console, in contrast, always displays messages queued directly to it. However, to display all messages that it normally receives, the console must be defined with LOGON(OPTIONAL), either by default or because it was specifically indicated, or it must be logged on by an operator.

Changing LOGON: The VARY CN command allows you to change the LOGON value of an MCS or SMCS console after the system is active:

```
VARY CN(consname),LOGON=OPTIONAL
                                AUTO
                                REQUIRED
                                DEFAULT
```

The change will take effect immediately.

This command requires MASTER console authority. It may be protected with the RACF MVS.VARYLOGON.CN profile in the OPERCMDS class, and it requires CONTROL authority.

LOGON can be combined with other parameters on the VARY command. The console can be active or inactive when LOGON is changed.

Changing the Predefined LU: The VARY CN command can also change the predefined LU of a SMCS console:

```
VARY CN(consname),LU=1uname
```

The same command can also turn off the predefined LU of a SMCS console::

```
VARY CN(consname),LU=*NONE*
```

However, if the console is not LOGON(REQUIRED), then an error message is issued to prevent a change that could accidentally create a security exposure. In that case, the LOGON keyword must be specified along with the LU keyword. The change will take effect immediately.

This command requires MASTER console authority. It may be protected with the RACF MVS.VARYLU.CN profile in the OPERCMDS class, and it requires CONTROL authority.

LU can be combined with other parameters on the VARY command.

Providing Security for SMCS Consoles

Now that operator consoles can be located anywhere, each installation must ensure proper security controls of operator access. There are many security issues to address, and these issues are installation-dependent.

Userids: The first thing to consider are userids. Each operator needs an individual userid that appropriately restricts access to controlled functions. Most security products control access based on the userid that is logged on to the console, not the console itself. Controlling access is very difficult unless LOGON(REQUIRED) is in effect.

Commands: Certain commands should be restricted only to users who need to issue those commands. This book lists all of the MVS commands that can be issued and the resource names that you can use to protect them. See Table 10 on page 62 for more information.

SMCS has introduced some new functions on the VARY command that could allow operators to create security exposures. SMCS options on the VARY command need particular consideration; VARY CN,LOGON and VARY CN,LU are examples. These commands require MASTER console authority, and it is a very good idea to use a security product to limit access to the commands. See Table 10 on page 62.

Application ID: Access to the SMCS APPLID can be protected through the RACF APPL class. You can use the APPL class to restrict certain users from accessing certain SMCS applications while allowing access to others, which means that certain users can activate consoles on some systems but not others. See “Planning Console Security” on page 52 for more information.

Console: The CONSOLE class of the security product can be used to restrict users from certain consoles. See “Planning Console Security” on page 52 for more information.

Network: There are security considerations for SMCS consoles at the network level. An SMCS console may display sensitive data, and since this data is flowing across the network, it must be protected. Ways to protect this data include:

- For TCP/IP networks, Secure Sockets Layer (SSL) security can be implemented to protect the IP session.
- Session level encryption can be used to protect a SNA session.
- Dedicated IP ports can be assigned to restrict access to SMCS.

See IBM SecureWay Communications Server and IBM SecureWay Security Server publications for more information.

Activating an SMCS Console

After the installation and definitions are complete, you can IPL the system. The system console or an MCS console must be used to do perform the IPL. Once MVS command processing is available, VTAM must be started in one of the following ways:

- A START VTAM command in COMMNDxx could start VTAM.
- Automation could START VTAM.
- An operator could START VTAM manually from the system console or an MCS console.

Once VTAM is initialized and the VTAM functions are available, the SMCS application will start automatically. SMCS consoles can then be activated.

Assuming the SMCS is installed on a system and some SMCS consoles are defined, there are several ways to activate an SMCS console. For example, an operator or system programmer can:

1. Walk up to a terminal, or telnet to the system, to get to an active VTAM logon screen
2. Log on to the SMCS application, which displays an SMCS Console Selection screen. See Figure 7.
3. On the SMCS Console Selection screen, enter a valid SMCS console name.
4. If the name is valid, the next screen is an SMCS console screen that displays messages unless logon is required. If logon is required, the messages appear after the operator logs on.

```
SMCS CONSOLE SELECTION

Enter the Console Name you want to access and press ENTER.

CONSOLE NAME ==>          (Required. This name must have been defined as an
                           SMCS console in CONSOLxx at IPL).

You are attempting to access:

  SYSPLEX: plexname SYSTEM: sysname

Licensed Materials - Property of IBM
"Restricted Materials of IBM"
5694-A01 (C) Copyright IBM Corp. 2001
All rights reserved.
```

Figure 7. SMCS Console Selection Screen

Specifying a predefined LU can bypass the SMCS Console Selection screen, and the LOGAPPL VTAM function allows automatic logon.

Deactivating an SMCS Console

Once an SMCS console is active, you might need to deactivate it. There are several ways to deactivate an SMCS console:

- The operator can issue the LOGOFF command at the console to deactivate the console.
- VARY consname,OFFLINE can deactivate the console.
- VARY CN (consname),OFFLINE can also deactivate the console.

SMCS consoles will also be deactivated by the system when VTAM or the SMCS application is deactivated.

Removing Console Definitions from a Configuration

You can delete the definition of any MCS, SMCS, or Subsystem console defined in CONSOLxx. In a sysplex, deleting a console definition releases the console id associated with the console and makes it available for other console definitions. Thus, you have flexibility controlling the number of console ids you need in an active console configuration. You cannot, however, delete the system console or an extended MCS console.

For example, if you define 10 consoles in CONSOLxx and you have used the VARY CONSOLE OFFLINE command for one of the consoles (it is inactive), the system still associates the console id with the inactive console. Using the console service, you can delete the console definition making the console id available for reuse. When you add a new console, the system reassigns the console id.

To remove a console definition, use the sample JCL for program IEARELCN in SYS1.SAMPLIB. "Sample Invocation of IEARELCN" describes the sample job, the programming environment, and the return and reason codes for invoking the console service.

Before you remove a console definition, issue DISPLAY CONSOLES to determine if the console is defined as an alternate. If you delete the definition for a console that is defined as an alternate for another console, you remove the alternate console as back up. If a console switch to the alternate is required, the system cannot switch because the alternate has already been removed. Use VARY CN to change the list of potential alternates for a console (ALTGRP) and VARY CONSOLE to change the definition for ALTERNATE.

The following restrictions for removing a console definition apply:

- Dynamic I/O reconfiguration can be performed for a device that has been defined as an MCS console in CONSOLxx. If you want to change the I/O configuration of a device which is defined as a console, you must first delete the console definition. The sample program IEARELCN can be used to do this. After the definition has been removed, the device can be dynamically reconfigured, but it cannot be used as a console again, until a re-IPL. In a mixed-level sysplex, dynamic I/O reconfiguration is not supported on the systems below OS/390 Release 6. For more information about dynamic I/O reconfiguration, see *z/OS HCD Planning*.
- The console must be defined in CONSOLxx.
- The console must not be active.
- A subsystem console that is in use must first be released. (See *z/OS MVS Using the Subsystem Interface*.)
- You cannot remove the console definition for a console when a no-consoles condition exists.
- When a no master console condition exists, you cannot remove a console that is the last active master console.

Sample Invocation of IEARELCN

SYS1.SAMPLIB provides a sample program in member IEARELCN to remove a console definition.

```
//jjj      JOB
//sss      EXEC PGM=IEARELCN,
//          PARM='CONSNAME(xxxxxxxx)'
//SYSPRINT DD  SYSOUT=A
```

xxxxxxx: is the name of the console whose definition is to be removed.

Environment

You can also invoke the console definition removal service (IEAVG730) from an authorized program. IEAVG730 receives control with the following environment:

```

Minimum authorization:  Supervisor state and key zero.
Dispatchable unit mode: Task
Cross Memory mode:    PASN=HASN=SASN
ASC mode:              Primary
Interrupt Status:     Enabled for I/O and external interrupts
Locks:                No locks held
Control parameters:   Control parameters must be in the primary
                       address space
  
```

Before you invoke IEAVG730 from your program, ensure that the following general purpose register (GPR) contains the specified information:

Register Contents

1 Address of a fullword containing the address of a field with the console name.

Return and Reason Codes

When control returns from the console definition removal service (module IEAVG730), the return code appears in register 15, and the reason code in register 0:

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning and Action
00	00	Successful processing.
04	00	Caller is not authorized. Ensure that caller is in supervisor state.
04	04	Caller is not authorized. Ensure that caller is in key zero.
04	08	Caller is in cross memory mode. Ensure that PASN = HASN = SASN.
04	0C	System level is not valid. In a sysplex, all the systems in the sysplex must be at OS/390 (R6 or later) or z/OS.
04	10	A no-consoles condition exists in the system or sysplex. Invoke this service again when the no-consoles condition is relieved.
08	00	Recovery cannot be established. Report error to the appropriate IBM support personnel.
08	04	Retry from an abend. Report error to the appropriate IBM support personnel.
08	08	This reason code is for IBM internal diagnostic purposes only. Record it and supply it to the appropriate IBM support personnel.
08	0C	This reason code is for IBM internal diagnostic purposes only. Record it and supply it to the appropriate IBM support personnel.
08	10	Secondary recovery cannot be established. Report error to the appropriate IBM support personnel.

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning and Action
08	14	Retry from an abend for the secondary recovery routine. Report error to the appropriate IBM support personnel.
08	18	This reason code is for IBM internal diagnostic purposes only. Record it and supply it to the appropriate IBM support personnel.
08	1C	This reason code is for IBM internal diagnostic purposes only. Record it and supply it to the appropriate IBM support personnel.
08	20	This reason code is for IBM internal diagnostic purposes only. Record it and supply it to the appropriate IBM support personnel.
0C	00	Console is active. If the console is an MCS console, deactivate the console. If the console is a subsystem console, the console is currently allocated to a subsystem. Release the subsystem console, and try the service again to remove the console.
0C	04	Console is not an MCS or SMCS console. Ensure that the console to be removed is for an MCS or SMCS console defined in CONSOLxx.
0C	08	Console is not defined in CONSOLxx. Ensure that the active CONSOLxx member contains a CONSOLE definition statement for an MCS or SMCS or subsystem allocatable console.
0C	0C	The console definition to be removed is for the master console and a no master console condition exists. Ensure that you have a new active master console and try the service again to remove the old master console.

Defining Devices as MCS or SMCS Consoles

The first step in planning an MVS console configuration is to define the I/O devices to MVS. Ensure that you define each I/O device that you plan to use as an MCS console with the hardware configuration definition (HCD) program for each MVS system at the installation. Use the HCD Add Device panel to define the device number and other information that identifies the device to MVS.

Note: MCS consoles are locally attached to the system through control devices that do not support Systems Network Architecture (SNA) protocols. SMCS consoles are not defined to HCD.

Use the following keywords on the CONSOLE statement to define a device as an MCS console.

DEVNUM Defines the console device number.
NAME Defines the console name.
UNIT Specifies the type of device to be used as an MCS console.

The device number you specify for each console on a CONSOLE statement - CONSOLE DEVNUM - must correspond to the device number specified through HCD on the Add Device panel. Except for DEVNUM, which must be first, you can

specify the keywords in any order. For MCS consoles that are managed by a subsystem (subsystem-allocatable consoles like NetView), you can specify:
CONSOLE DEVNUM(SUBSYSTEM) NAME(**name**)

where **name** is the name of the subsystem console.

You can specify DEVNUM(SYSCONS) to define the system console in CONSOLxx. See “The System Console and CONSOLxx” on page 138.

Note: The system pins UCBs for console devices defined in CONSOLxx at IPL time. Therefore, you must IPL or remove the console definition with IEARELCN if you delete console devices via HCD.

Use the following keywords on the CONSOLE statement to define the device number and name of an SMCS console.

DEVNUM

Specify SMCS. This must be the first parameter on the CONSOLE statement.

NAME Specify the name of the SMCS console. If the name is not valid, the system rejects the CONSOLE statement.

Do not specify the UNIT, ALTERNATE, or SYSTEM keywords on the CONSOLE statement for SMCS consoles.

Devices MVS Can Use as MCS Consoles

MCS consoles are either output-only devices like printers or input/output devices like a 3279 display console. You can define printers as hardcopy devices and specify the printer device on the HARDCOPY statement.

Input/output devices are also called display consoles. An MCS display console can be a combination input (operator-to system) and output (system-to-operator) device whose function you can control. You control how to use the display console with the USE attribute so that it can be a full-capability console (send commands and receive messages), or an output-only console like a message stream console or status display console, from which an operator cannot enter commands. For information on USE, see “Defining the USE Attribute” on page 71.

If you use 3270-X devices as display consoles, consider the following:

- If the console device you plan to use is attached to a control unit that supports the Read Partition Query Feature and the device also supports the feature, specify 3270-X for UNIT.
- Only 3270-X devices can display synchronous messages issued during certain recovery procedures. (For information about synchronous messages, see “Display of Synchronous Messages” on page 51.)

In this book, references to devices often do not mention model numbers. When you see a device referenced without a model number, assume the reference applies to all models of the device.

Reference

For a list of devices that MVS can use as MCS consoles (including eligible 3270-X devices), see *z/OS MVS Initialization and Tuning Reference*.

For information about using HCD to define console devices, see *z/OS HCD Planning*.

With z/OS V1R1 and higher, MVS can also use SMCS consoles. See “Multiple Console Support and the MVS Environment” on page 2

Using Console Names

Define each console by device number and device unit on the CONSOLE statement and name each MCS console. Console names are recommended for all CONSOLE statements in CONSOLxx. Console names are required for SMCS consoles and optional for other types of consoles.

If you do not define an MCS console by name, the console ID that the system assigns becomes the name of the console. The MCS console ID is a two-digit number from 1 to 99 (01 - 99) based on the order in which the CONSOLE statements appear in CONSOLxx.

Using console names for MCS consoles has several advantages. Console names are generally easier to remember and use than console IDs. Also, you can choose a console name that indicates a specific function for the console. (For example, you could define a console name of TAPE for a console that receives messages about tape mounts.) Operators and system programmers can use console names instead of console IDs on MVS commands and macros. You also specify console names for alternate consoles that you define as backup consoles and in alternate console group definitions. For consoles in a sysplex, assigning console names allows you to define an alternate or backup console that is attached to another system. (See “Planning Console Recovery” on page 43.)

Using Console Names in a Sysplex

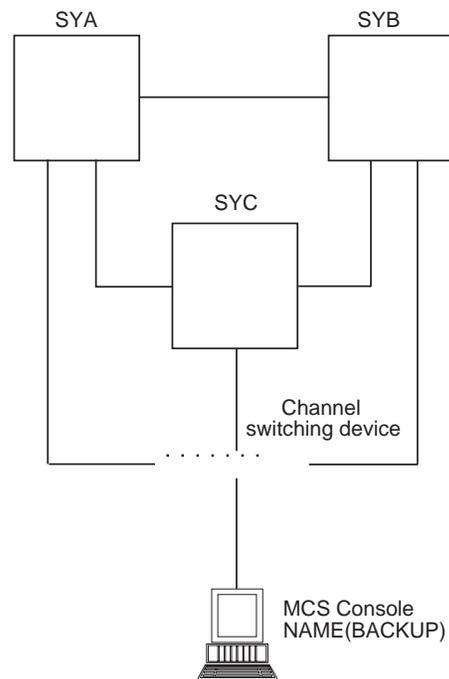
When defining consoles for a sysplex, plan to use names for MCS and SMCS consoles and subsystem-allocatable consoles. A good way to specify unique names and establish a consistent naming convention for all the consoles in a sysplex is to use *system symbols* in console definitions, as described in “Sharing a Single CONSOLxx Member for All Systems” on page 22.

In a sysplex, the console name uniquely identifies the console to the sysplex for the life of the sysplex. The console ID identifies a console only for the life of the IPL. Because MVS can assign a different console ID to a console on a system that re-IPLs, using console IDs is unpredictable.

For a subsystem-allocatable console if you do not use console names, MVS assigns the next available console ID for the console whenever it rejoins the sysplex. For example, an unnamed subsystem-allocatable console, is assigned a console ID of 05 in a sysplex with four active consoles (IDs 01, 02, 03, and 04). If the system with the subsystem-allocatable console, leaves the sysplex and rejoins later, MVS does not reassign the console ID 05 to the console, but instead assigns the next available ID 06. Even if you have only five consoles in the sysplex, MVS assigns the next available ID. In this example, that increases the number of console IDs in use for the sysplex from 5 to 6 even though only five consoles are active.

Using console names avoids using more console IDs in a sysplex than the sysplex needs. As long as you do not change the name in CONSOLxx, the console name always identifies the console in a system or sysplex. MVS always associates the console name to a specific ID that does not change from IPL to IPL.

You can also define the **same** console to different systems in the sysplex by using the console name. In the following example, a console named BACKUP is defined in CONSOLxx for three systems in a sysplex (SYA, SYB, and SYC). A channel switching device allows an operator to switch the console from system to system:



CONSOLxx for each system contains the following statements:

For SYA:

```
CONSOLE DEVNUM(3F5) NAME(BACKUP)
```

For SYB:

```
CONSOLE DEVNUM(3D0) NAME(BACKUP)
```

For SYC:

```
CONSOLE DEVNUM(3E0) NAME(BACKUP)
```

BACKUP can be active on only one system in the sysplex at a time. If BACKUP is active on SYA and SYA fails, the operator can switch BACKUP to SYB or SYC and activate the console using the console name on the VARY CN command. The attributes that were assigned to BACKUP before SYA failed are now inherited by the newly activated BACKUP console on either SYB or SYC.

In a sysplex, you can use the console name to define an alternate to a console that resides on a different system. If a console fails and its alternate is defined on a different system, the sysplex can switch to the alternate if it is active. For information on console recovery, see “Planning Console Recovery” on page 43.

Restrictions for Console Names

Console names must be from 2 to 8 characters and cannot start with a digit. Characters are alphanumeric and can also include the characters #, \$, and @. When naming MCS consoles, do not use the following names:

- HC
- INSTREAM
- INTERNAL
- OPERLOG
- SYSIOSRS

Note about SYSIOSRS

Console name SYSIOSRS is reserved for system use.

The IOSAS address space must have an associated “trusted” userid define in the RACF started procedures table (ICHRIN03). This will permit the IOSAS address space to issue commands to the SYSIOSRS extended console. For more information, see *z/OS Security Server RACF Security Administrator's Guide*

- SYSLOG
- UNKNOWN

Also, do not use console names that might be confused with device numbers. For example, the following name is not a good choice:

NAME(BAD)

For information on the system console and naming restrictions, see “Naming the System Console” on page 138.

Attaching Consoles to Particular Systems in a Sysplex

Use the optional SYSTEM keyword parameter in the CONSOLxx parmlib member to specify the system in the sysplex to which MVS should attempt to activate the console. This parameter will primarily refer to consoles that are physically attached to multiple systems and managed by a physical switch. In this case, the SYSTEM parameter determines which system should attempt to activate the console.

If SYSTEM is specified and the SYSTEM value names the current system being initialized, then MVS will activate the console device if the device is attached and in ready status. If the SYSTEM value names a system other than the one currently being initialized, then MVS will not activate the console even if it is attached and ready on the system being initialized. If SYSTEM is not specified, MVS activates the console on the first system to join the sysplex (to which the console is attached and ready).

Attention: Use the SYSTEM parameter with great care whenever there is more than one CONSOLxx parmlib member for the sysplex. If you define multiple CONSOLE statements with the same DEVNUM and specify a SYSTEM differently on different statements, the system will activate the device as a console on the first system where it (a) is online and ready, and (b) has a SYSTEM parameter value equal to the name of the IPLing system, or has no SYSTEM keyword.

It is possible that a device will not be ready (not turned on) when the system or sysplex is being initialized. The device might even be attached to another system as the sysplex is initialized (for example, during an error recovery situation). When

you decide to use the device, first turn it on or re-attach it to the proper system, then issue a VARY CN,ONLINE command for the console.

During VARY CN,ONLINE command processing, the CONSOLE statement SYSTEM value is used to determine where to process the VARY CN,ONLINE command (unless the console was previously active or SYSTEM is specified on the VARY CN,ONLINE command).

Note: SMCS consoles are not associated with a particular system and so the SYSTEM keyword is not valid for these types of consoles. An SMCS console defined on one system can be activated on another system (provided both systems have SMCS active).

Planning Console Recovery

Planning recovery for consoles ensures that your operators are able to respond to system problems and continue to monitor the MVS environment. You can specify:

- A group of consoles from which MVS can select an eligible alternate for a console
- A single alternate for a console

You can specify a group of consoles that can serve as candidates:

- For alternates to a console
- For the master console when no master console is available
- For the hardcopy device that the installation specifies as the hardcopy log.
- To receive synchronous messages that bypass regular message queuing

If a console fails or the system issues a synchronous message, MVS can switch to the first available console in the group.

Planning console recovery by using alternate console groups allows an installation to define console recovery according to group function. For example, you might define a group of consoles that display messages about various printers at an installation. Each console might specify an alternate console group that includes these consoles so that if a console fails, MVS can switch to one of the functionally related consoles in the group. In a sysplex, you can define alternate consoles that reside on different systems as members of an alternate console group, but ensure that you define the consoles by console name.

Console recovery using alternate console groups is available to MCS consoles, SMCS consoles, and extended MCS consoles. You use the CONSOLE statement of CONSOLxx to define the alternate console group for an MCS or SMCS console and the RACF ADDUSER or ALTUSER command to define an alternate console group for an extended MCS console.

You cannot specify a single alternate console for an SMCS console; you must specify an alternate group. It is possible to specify a single console as an alternate for an MCS or extended MCS console, but using an alternate group provides greater flexibility in terms of console recovery than using a single alternate console as backup. IBM recommends that you provide an alternate group as backup for all consoles.

Recovery Considerations

When you plan console recovery for a system or a sysplex, first consider all the MCS consoles, SMCS consoles, and extended MCS consoles you have defined for

your console configuration. (For MCS and SMCS consoles, you must consider the console definitions in CONSOLxx for each system. For extended MCS consoles, you must consider the TSO/E userids that the RACF or TSO/E administrator defines for each system.) You should consider backing up each of these consoles.

Also consider the following:

- “Parmlib Members and Console Recovery”
- “Alternate Console Groups and Console Backup”
- “Console Switching and Console Recovery” on page 48
- “Role of the Master Console During Console Recovery” on page 49
- “No-Master-Console Condition” on page 49
- “No-Consoles Condition” on page 50
- “Display of Synchronous Messages” on page 51

Parmlib Members and Console Recovery

You can use the following Parmlib members to define consoles and their alternates:

- CONSOLxx
- CNGRPxx

CONSOLxx allows you

- To define an MCS console and the name of its alternate console group or alternate console.
- To define an SMCS console and the name of its alternate console group.
- To activate the CNGRPxx member that contains alternate console group definitions.
- To define a group of MCS consoles to use as the master console when no full-capability consoles are available.
- To define a group of MCS consoles to handle the display of synchronous messages.
- To define a group of MCS console devices to back up the hardcopy log.

CNGRPxx allows you to define console groups with MCS consoles, SMCS consoles, and extended MCS consoles as members.

Alternate Console Groups and Console Backup

You can use the CNGRPxx Parmlib member to specify alternate console groups whose members can be used as alternate consoles. In CNGRPxx, you define the group name and the MCS consoles, SMCS consoles or extended MCS consoles that are members of the group.

During a console switch (when a console fails, for example, or when an operator issues the SWITCH command to switch to a console’s alternate), MVS searches for an alternate console based on the order of the console members defined in the group.

Using CNGRPxx to Define Alternate Console Groups

The GROUP statement of CNGRPxx allows you to define an alternate console group and its members. On the GROUP statement, you can specify

- The name of the alternate console group.
- The names of the MCS consoles, SMCS consoles, or extended MCS consoles that serve as candidates in the alternate console group.

You can specify the name of the alternate console group with its console group members on

- **CONSOLE ALTGRP** to define alternates for MCS consoles, and SMCS consoles. (See “Alternate Console Groups and CONSOLxx” on page 46.)
- **INIT NOCCGRP** to define alternates for the master console. (See “No-Master-Console Condition” on page 49.)
- **DEFAULT SYNCHDEST** to define a group of consoles able to display synchronous messages. (See “Display of Synchronous Messages” on page 51.)
- **HARDCOPY HCPYGRP** to define alternates for the hardcopy log device
- **RACF ADDUSER ALTGRP** or **ALTUSER ALTGRP** commands to define alternates for an extended MCS console. (ALTGRP is a subkeyword of the OPERPARM segment that defines attributes for extended MCS consoles.)

When you define group names in CNGRPxx, do not duplicate either the names of MCS or SMCS consoles defined in CONSOLxx or the console key names specified for the DISPLAY KEY command.

Activating CNGRPxx

To activate the CNGRPxx member or members at IPL time, use the following keyword on the INIT statement of CONSOLxx:

CNGRP Specifies the member or members of CNGRPxx that you want active.

NO indicates that you do not want to specify an alternate console group and is the default.

You can also activate CNGRPxx members at IPL time by placing the SET CNGRP command in the COMMNDxx of Parmlib member. Operators can use SET CNGRP to change the specifications after IPL.

You can activate more than one CNGRPxx member at a time. If you activate two or more CNGRPxx members for a system or sysplex and define the same group name in different members, MVS uses the group definition of the first member you specify.

INIT CNGRP has sysplex scope. The first system IPLed into a sysplex defines alternate console groups for the entire sysplex through the CNGRPxx member specified on its INIT statement. MVS ignores the INIT CNGRP values of other systems that subsequently join the sysplex. To change the CNGRPxx member after IPL, operators can use the SET CNGRP command, which affects all systems in a sysplex for the life of the IPL.

To display information about the CNGRPxx members in effect for a system or sysplex, operators can use DISPLAY CNGRP.

Reference

For complete information on CNGRPxx, see *z/OS MVS Initialization and Tuning Reference*.

Alternate Console Groups and CONSOLxx

You can specify the group names that you have defined in CNGRPxx for each console in CONSOLxx. On the CONSOLE statement for the console, use the following keyword to assign an alternate console group name:

ALTGRP Defines the name of the alternate console group for the console.

If the console fails, MVS can switch to the first available console defined in the group. MVS searches for the first available console based on the order of the console members defined for the alternate console group.

For example, the following console group TAPEGR defines three members in CNGRPxx:

```
GROUP NAME(TAPEGR)
    MEMBERS(TAPE2,TAPE3,EXTAPE)
```

TAPE2 and TAPE3 are consoles defined in CONSOLxx; EXTAPE is an extended MCS console defined by RACF.

You define the following CONSOLE statement for TAPE1:

```
CONSOLE NAME(TAPE1) ALTGRP(TAPEGR)
```

If TAPE1 fails, MVS first tries to switch to TAPE2, then TAPE3, and finally to EXTAPE, depending on which console is available. If EXTAPE (the last defined alternate for TAPE1) is not available, MVS can send action messages, WTOR messages, or important informational messages destined for TAPE1, to any MCS console, SMCS console, or extended MCS console with the UD attribute, as well as the master console. For information about the UD attribute, see “Defining the UD Attribute for Consoles” on page 107. If the master console is unavailable or no console in the configuration has the UD attribute, MVS can use the system console. See “Alternate Console of Last Resort” on page 47.

Changing the Specification of Alternate Console Groups

Operators can use the VARY CN,ALTGRP command to reassign alternate console groups for MCS consoles, SMCS consoles, and extended MCS consoles.

Changing Console Alternates without Re-IPLing

You can define the same alternate console group name in different CNGRPxx members and specify different alternate consoles as members of the group. For example, you can define the group name TAPEGR in CNGRP01 and CNGRP02

```
(CNGRP01) GROUP NAME(TAPEGR)
    MEMBERS(TAPE1, TAPE2)
```

```
(CNGRP02) GROUP NAME(TAPEGR)
    MEMBERS(TAPE3, TAPE4)
```

When you define alternate groups in CONSOLxx, you can specify the alternate console group name TAPEGR. If CNGRP01 is active and you want to change the alternate console specifications to those in CNGRP02, the operator can issue SET CNGRP02 to change the alternates without having to re-IPL the system.

Alternate Console Groups and Extended MCS Consoles

To define alternate console groups for extended MCS consoles, your RACF security administrator can use the RACF ADDUSER ALTGRP command or change specifications using the RACF ALTUSER ALTGRP command. For an example of

how to use these commands to define extended MCS consoles and their attributes, see “Controlling Extended MCS Consoles Using RACF” on page 152.

Using the ALTERNATE Keyword on the CONSOLE Statement

If you want to specify a single alternate for a console, you can use the following keyword on the CONSOLE statement:

ALTERNATE Defines the console name or device number of another MCS console to act as backup

Note, however, that ALTERNATE is not the recommended way to define backup consoles; console groups and the ALTGRP keyword provide more flexible recovery. You can define separate console statements using ALTERNATE and ALTGRP in the same CONSOLxx parmlib member:

```
CONSOLE . . . NAME(DISK) ALTERNATE(DISK2)
```

```
CONSOLE . . . NAME(TAPE1) ALTGRP(TAPEGR)
```

You cannot specify an extended MCS console on ALTERNATE. You can specify an MCS console or an SMCS console on ALTERNATE for an MCS console. You cannot specify ALTERNATE for an SMCS console.

If you specify ALTERNATE, MVS switches only to the console specified if it is available; MVS does not recognize console chaining through the use of the ALTERNATE keyword.

Note: In the examples that follow, the required CONSOLE keyword DEVNUM has been omitted.

For example, if you specify this:

```
CONSOLE . . . NAME(TAPE) ALTERNATE(DISK1)
```

```
CONSOLE . . . NAME(DISK1) ALTERNATE(DISK2)
```

```
CONSOLE . . . NAME(DISK2) ALTGRP(DISKGR)
```

and TAPE fails, MVS tries to switch to DISK1. If DISK1 is not available, MVS does NOT try to switch to DISK2. The following chart summarizes how ALTERNATE and ALTGRP work in this example.

Table 9. Console Switching Using the ALTERNATE or ALTGRP keyword

Console	Alternate console or group	Event
TAPE	ALTERNATE(DISK1)	If TAPE fails, switches to DISK1. If DISK1 fails, does not switch.
DISK1	ALTERNATE(DISK2)	If DISK1 fails, switches to DISK2. If DISK2 fails, does not switch.
DISK2	ALTGRP(DISKGR)	If DISK2 fails, switches to first member of DISKGR. If that member fails, switches to next member of DISKGR, and so forth. If the last member specified in the group fails, does not switch.

Alternate Console of Last Resort

If the console on ALTERNATE is not available, or none of the consoles specified for ALTGRP is available, MVS uses

- Any console with the UD attribute (including the master console by default) to send action messages, WTOR messages, or important informational messages destined for the failing console. “Defining the UD Attribute for Consoles” on page 107 describes assigning the UD attribute to a console.
- The system console to send undelivered messages when no other console with the UD attribute is available.

Console Switching and Console Recovery

When the system switches a failing console to an alternate (or when an operator switches a console as a result of the SWITCH command), MVS merges the following console attributes with those of the alternate:

- Routing codes (ROUTCODE)
- Message levels (LEVEL)
- Command authority of MCS consoles and SMCS consoles (AUTH)
- Message scope in a sysplex (MSCOPE)
- Ability for the console to receive undelivered messages (UD)

Console switching occurs for MCS consoles, SMCS consoles, and extended MCS consoles.

Note that the attributes are added to those of the alternate console and do not replace the existing attributes. Thus, the command authority, message scope, and UD status of the alternate console are not permanently affected by the addition of the failing console’s attributes.

The following example illustrates how console switching works with routing codes. If console TAPE1 with routing codes 1, 2, and 7 switches to its alternate console TAPE2, MVS merges the routing codes of TAPE1 with those of TAPE2 and redirects TAPE1’s messages to TAPE2. If TAPE2 has routing codes 7-10, the following routing codes are in effect for TAPE2 after the switch:

```
TAPE2 ROUTCODE(1,2,7-10)
```

When an operator reactivates the console (TAPE1 in this example), MVS restores the console attributes of TAPE1 and removes TAPE1’s attributes from TAPE2.

```
TAPE1 ROUTCODE(1,2,7)
```

```
TAPE2 ROUTCODE(7-10)
```

In a sysplex, if a system with attached consoles fails, MVS can switch the attributes of the failed consoles (including extended MCS consoles) to consoles on an active system.

Note: In a JES3 complex, JES3 can make use of the dynamic system interchange (DSI) function to ensure that messages destined for a console on a failing system are redirected to an active system. See *z/OS JES3 Initialization and Tuning Guide*.

The SWITCH CN Command

Operators can use the SWITCH CN command to switch console attributes between consoles. Using the switch command might be useful

- To handle message traffic during changes to operator shifts
- To redistribute operator workload by rerouting messages

Consoles can be MCS consoles, SMCS consoles, or extended MCS consoles. The operator can use SWITCH to do the following:

- Switch the console attributes of an active or inactive console to another console.
- Switch the console attributes of an active or inactive console to its first available alternate in the alternate console group or the console specified on ALTERNATE.
- Restore the console attributes of an inactive console that has switched to an active alternate.

When an operator uses the SWITCH command to switch from an MCS or SMCS console to another console or its alternate, MVS first merges the console attributes with the other console and then deactivates the switched console.

Operators cannot use the SWITCH CN command to switch the master console, the hardcopy log device, or the system console.

Note:

To switch the console attributes of an inactive extended MCS console, it is the responsibility of the program that displays messages for an extended MCS console session to first deactivate the console session using the MCSOPER DEACTIVATE macro. For information on MCSOPER, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Reference

z/OS MVS System Commands describes the full syntax and use of the SWITCH command.

Console Recovery and the RESET CN Command

If a problem occurs that causes a console to become unusable and attempts to restore the console fail (for example, in response to the VARY command, the system issues message IEE339I indicating that the console is changing status), the operator does not have to re-IPL the system to recover the console. From another console, the operator can first issue the RESET CN command and then either issue the VARY CN,ONLINE command for the inactive console, or, if the inaction console is an SMCS console, log the console on again.

For information on using RESET CN, see *z/OS MVS System Commands*.

Role of the Master Console During Console Recovery

The master console is a full-capability console from which the operator has the authority to enter any MVS commands. As a result, the master console is often the focal point of MVS operations. In a system or sysplex, the master console becomes the alternate for any console without a defined alternate or alternate console group.

No-Master-Console Condition

If the master console fails and is unable to switch, a no-master-console condition exists. If the master console fails, you should restore the master console as soon as possible.

If there are no active alternates for the master console but other full-capability consoles are active in the system or sysplex, the operator can issue VARY MSTCONS on one of the consoles. VARY MSTCONS allows an active full-capability console to become the master console.

In a no-master-consoles condition, the system accepts the VARY MSTCONS command from any console. Thus, any MCS or SMCS console that is not a MASTER authority console can issue the VARY MSTCONS command to make itself the master console unless the installation has used RACF to prevent the console from making itself the master console. IBM strongly recommends that you use RACF to prevent any user that should not be allowed to have access to the master console from issuing the VARY MSTCONS command. Provide an MVS.VARY.MSTCONS profile in the OPERCMDS class and ensure that the user does not have READ or greater authority. Note that if you do not require operator LOGON, it is extremely difficult, if not impossible, to control authorization.

If the master console failed due to a system partition, bringing the failing system back into the sysplex will also relieve the no-master-console condition.

No-Consoles Condition

If no full-capability consoles are active in a system or sysplex, a no-consoles condition exists. For a no-consoles condition, the operator can:

- Issue VARY CN(),ONLINE (from an extended MCS or subsystem console) to activate a full-capability console that is offline and make it the master.
- Issue CONTROL V,USE=FC (from an extended MCS or subsystem console) to change an active message stream or status display MCS console to a full-capability MCS console and make it the master.
- Press the attention interrupt key on an MCS console device that is to become the master console, then press the external interrupt key on the system console to activate the device as the master console.
- In a sysplex, an operator can use the system console to IPL a system with a full-capability console (defined with AUTH=MASTER) into the sysplex.
- Activate an SMCS console to relieve the no-consoles condition. If the console was defined AUTH=MASTER, it will become the master console. Otherwise, it will not become the master console, and the system or sysplex will enter a no-master-console condition as described in “No-Master-Console Condition” on page 49. In a no-master-console condition, the operator using the console can make the console the master console using the VARY MSTCONS command, unless the installation takes steps to prevent it as described in “No-Master-Console Condition” on page 49.

Selecting a Master Console Using Alternate Console Groups

When no full-capability consoles are available in a system or sysplex, an operator can select an MCS console from an alternate console group specified on the INIT statement and activate it as the master console. When the operator presses the attention interrupt key on any MCS console device that is a member of the alternate console group and then presses the external interrupt key of the system console, MVS can activate the MCS console as the master console. You cannot select an SMCS console by pressing the ATTN key. Pressing the external interrupt key while the master console is an SMCS console causes the console to switch to its alternate.

Use the following keyword on the INIT statement to specify a master console group:

NOCCGRP Specifies the name of the alternate console group defined in CNGRPxx from whose members the system or sysplex can select a master console during a no consoles condition.

In a system or sysplex, the group specified on NOCCGRP must be defined in an active CNGRPxx member.

Console members in the group should be defined as full-capability MCS consoles. In a sysplex, the master console that an operator selects and the system console that the operator uses to activate the console must be attached to the same system.

Devices defined as master console alternates on INIT NOCCGRP allow MVS to select only the MCS console devices specified. Thus, when a no-consoles condition occurs, subsystem consoles like NetView are not affected and the subsystems can continue to use the consoles.

Display of Synchronous Messages

Synchronous messages are WTO or WTOR messages that can be issued during initialization or recovery situations, or by programs that want messages to bypass normal message queuing. In a sysplex, a console can display a synchronous message only if it is attached to the system that issues the message.

You can define the MCS master console, the system console, or other MCS consoles as members of an alternate console group in CNGRPxx to receive synchronous messages.

Note: You cannot define an SMCS console to receive synchronous messages.

Use the following keyword on the DEFAULT statement of CONSOLxx, to handle the display of a synchronous message:

SYNCHDEST Specifies the name of the alternate console group whose members can receive a synchronous message.

MVS searches for an eligible console based on the order of the console members specified in the group. You can specify valid MCS console names as members of the group. You can also specify *MSTCON*, the master console in the system or sysplex, or *SYSCON*, the system console. To receive the synchronous message, the console must be attached to the system that issues the message.

Considerations Using Consoles to Display Synchronous Messages

If you do not specify an alternate console group on SYNCHDEST or none of the consoles on SYNCHDEST are active, the system that issues the message tries to select the following console:

1. The master console, if it is not an SMCS console but is active and physically attached to the system that issues the message; otherwise,
2. The system console on the system that issues a wait-state message or WTOR message; normally MVS tries to display only WTOR messages on the system console.

For a sysplex environment, you should understand and plan where your synchronous messages will be displayed.

Synchronous messages can be displayed only on the system where they originated. They can be displayed on any MCS console attached to the system, but you must specify the console(s) to be used, in the SYNCHDEST console group. Systems with no attached MCS consoles will use the system console for these messages.

The SYNCHDEST console group is an ordered list of consoles where MVS is to attempt to display synchronous messages. The system console can be specified in the list. If an MCS console in the list is not attached to the system where the message is issued, it is skipped. So, the same SYNCHDEST group can be used for all systems, if you wish. If a console in the list is an SMCS console, it is skipped, and *MSTCON* is ignored if the master console is an SMCS console.

If the system attempts to use a console for a synchronous message and fails, the next console in the SYNCHDEST group, which is attached to this system, will be used. The system console can be specified in the group, and will also be used as a last resort, if all other console attempts have failed.

If MCS consoles share a control unit and an operator tries to respond to a synchronous message on one of the consoles, interruptions from the other consoles can make it impossible for the operator to reply to a synchronous message. When you plan your sysplex recovery, you should attach the MCS console that is to display synchronous messages to its own control unit without any other attached console. If it shares a control unit, there is a higher probability of failure on the console; the message will then be attempted on the next suitable console in the SYNCHDEST group, or on the system console.

Planning Console Security

Console security means controlling which commands operators can enter on their consoles to monitor and control MVS. How you define command authorities for your consoles or control logon for operators allows you to plan the operations security of your MVS system or sysplex. In a sysplex, because an operator on one system can enter commands that affect the processing on another system, your security measures become more complicated and you need to plan accordingly.

If your installation plans to use extended MCS consoles, you should consider ways to control what an authorized TSO/E user can do during a console session. Because an extended MCS console can be associated with a TSO/E userid and not a physical console, you might want to use RACF to limit not only the MVS commands a user can enter but from which TSO/E terminals the user can enter the commands.

You can control whether an operator can enter commands from a console:

- Through the AUTH keyword on the CONSOLE statement of CONSOLxx
- Through the LOGON keyword of the DEFAULT statement and RACF commands and profiles.

“Controlling Command Authority with the AUTH Attribute” describes the AUTH attribute and command groups. “Using RACF to Control Command Authority and Operator Logon” on page 54 describes RACF and the LOGON keyword for the DEFAULT statement. Special security considerations for SMCS consoles appear in “Providing Security for SMCS Consoles” on page 34.

Controlling Command Authority with the AUTH Attribute

The AUTH keyword on the CONSOLE statement of CONSOLxx allows you to control the command authority of your full-capability consoles so that the system accepts commands defined by command group that you assign for the console. For example, consoles with master authority need to issue **all** commands, including those that affect other consoles (including extended MCS consoles). On the other hand, a console used only to issue I/O commands, such as PURGE, MOUNT, and

UNLOAD, needs the authority to issue only certain commands. For this reason, MVS commands are grouped into system command groups that allow you to control which commands operators can issue from any given console.

MVS commands are assigned to one of five command groups according to command function. The command groups are:

- Informational commands (INFO)
- System control commands (SYS)
- I/O control commands (IO)
- Console control commands (CONS)
- Master console commands (MASTER)

For a list of the commands in each group see system command group information in *z/OS MVS System Commands*. (For information about JES2 commands, see *z/OS JES2 Commands* or *z/OS JES3 Commands*.)

To authorize which of the command groups an operator can enter on an MCS or SMCS console, use the following keyword on the `CONSOLE` statement.

AUTH Defines the command authority for an MCS or SMCS console

Options you can specify for AUTH include the following:

MASTER Specifies that the console has master authority. You can enter all MVS operator commands from the master console.

INFO Specifies that the console can issue any informational commands and is the default value

SYS Specifies that the console can issue system control commands and informational commands

IO Specifies that the console can issue I/O control commands and informational commands

CONS Specifies that the console can issue console control commands and informational commands

ALL Specifies that the console can issue informational, system control, I/O control, and console control commands

Operators can use the `VARY CN` command to change AUTH.

An operator can enter informational commands from any full-capability console. You can specify any combination of SYS, IO, and CONS together on the AUTH keyword so that an operator can enter these commands (along with informational commands) from the console. If an operator enters a command at a console where it is not authorized, MVS rejects the command and sends an error message to the issuing console.

Because consoles can receive messages based on assigned routing codes and message levels, ensure that the console has the proper authority for the operator to be able to respond to the message. For a description of message routing codes and levels, see “Message and Command Routing” on page 96.

Assigning a Console Master Authority

By assigning the system command groups for a console in a system, you can establish a console with master authority.

For example, to assign master authority to a console named MSTR (device number 031), code the following CONSOLE statement in CONSOLxx:

```
CONSOLE DEVNUM(031) NAME(MSTR) AUTH(MASTER)
```

If no console is defined with master authority in the system, the first full-capability MCS console will be made to have master authority.

In a sysplex, the first active console with master authority in the first system that joins the sysplex becomes the master console. You can define AUTH(MASTER) for other consoles in that system or for other systems that subsequently join the sysplex. These consoles have master authority, but there can be only one master console in the sysplex.

In a sysplex with only SMCS consoles, the first active console with MASTER authority becomes the master console. Because it is impossible to know which console will activate first, any console that should not be the sysplex master should not have MASTER authority.

Operators can assign the master authority of a console by using the following command:

```
VARY CN(name),AUTH=MASTER
```

This command authorizes the console with master authority and establishes the commands that the console can receive. If a console with master authority is operating properly, an operator can switch to another console without disrupting normal operations. The operator must enter these commands through the console currently defined with master authority. The effect of the VARY command lasts only for the duration of the IPL.

Using RACF to Control Command Authority and Operator Logon

CONSOLxx provides a way to limit command authority for MCS and SMCS consoles. However, to control operator logon, limit the use of specific commands to specific MCS and SMCS consoles, or control command use for extended MCS consoles, your RACF security administrator can help you plan your console security. When you use RACF, you need to educate operators about the security policy at the installation and the changes to their jobs that the security policy requires.

An installation can audit the use of commands and limit the use of commands by operator as well as by console:

- Based on the identity of the issuer of the command — who issued the command. Using this method, the installation can verify that the operator who issues a command is authorized to do so and optionally produce audit records that log command activity. The installation can control who can issue what commands at several different levels. For example, all operators might be allowed to issue all commands, some operators might be allowed to enter only a subset of the allowable commands, or some commands might be restricted to just one or two individual operators.
- Based on the MCS console device number or the console name used to enter the command — where the command was issued. Using this method, the installation can verify that the command has been issued from a console that is authorized to issue the command and optionally produce audit records that log command activity.

- Based on both the identity of the command issuer and the console device number or console name used to enter the command — both who issued the command and where the command was issued. Using this method, the installation can verify that the operator who issues a command is authorized to do so and that the command has been issued from a console that is authorized to issue the command. Audit records can log command activity.

Your installation can use RACF and CONSOLxx to provide restrictions on the use of system commands to meet the security policy at your installation. If console definition (through the AUTH keyword) provides adequate control of command use, you need take no action. Simply ensure that the LOGON parameter on the CONSOLE or DEFAULT statement in the CONSOLxx Parmlib member is set to OPTIONAL, which is the default.

Using RACF to Authorize Console Operators and Command Use

If your installation requires additional security controls on the use of system commands, you must first determine what controls are required. For example, do you want to require all your operators to logon to MCS or SMCS consoles, or do you want certain operators with special authority to be able to enter commands that require a higher authority than the console allows? Do you want to audit logon activity? If so, do you want to log all command activity or only unauthorized, or unsuccessful, attempts to issue system commands? Using RACF and the LOGON keyword in CONSOLxx can help you achieve the kind of added security you might need.

If your installation uses SMCS consoles, IBM strongly recommends that you use RACF to prevent an SMCS console from making itself the master console. That is, provide an MVS.VARY.MSTCONS profile in the OPERCMDS class and ensure the user does not have READ or greater authority. Note that, if you do not require operator LOGON, it is extremely difficult, if not impossible, to control authorization.

If your installation uses extended MCS consoles, you need to plan for their security. Your TSO or RACF security administrator can help you authorize TSO/E users and control the console attributes (defined in the OPERPARM segment) for those users. For examples, see “Controlling Extended MCS Consoles Using RACF” on page 152.

Note that using RACF to authorize commands can increase the path length the system requires to process a command, and auditing command activity can increase the number of security-related SMF records your system generates.

Defining RACF Profiles

To determine whether a particular user (an operator) is allowed to access a particular resource (a command or a console), RACF uses the RACF profiles. The security administrator can define a RACF profile for:

- Each user of a console
- Each console that is to be automatically logged on
- Each MVS command issued from a console
- Each user of the SMCS application is able to enter a command.

SMCS will support the protecting of the SMCS application via the APPL class of a security product. If the user is defined and authorized by the security product and the APPL class is not active or the APPL class is active but no profile matches the SMCS APPLID, access will be granted. If the APPL class is active and a profile matching the SMCS APPLID exists, the name the user is logging on with must be

defined in the profile's access list with at least READ authority for access to be granted. If the console has been defined with LOGON(AUTO), the console name must be in the access list.

Using RACF to authorize commands means that each operator requires an individual user profile. (TSO/E users of extended MCS consoles should already have a RACF profile in order for them to log on to TSO.) This user profile establishes the userid of the individual operator, and the userid identifies the operator when the operator logs on to the system. You can define the operator's or TSO/E user's authority to access resources by userid, but you can also establish access authority through a RACF group. For example, if you have several operators or TSO/E users with identical access requirements, you can have the security administrator create a RACF group and define the access for the individual operators or TSO/E users through the group. For more information, see "Defining Users with RACF" on page 57.

If you want an MCS console to be automatically logged on when you specify LOGON(AUTO), you must ensure that each console has a user profile established for it. Your RACF security administrator can define a user profile by console name. When LOGON(AUTO) is in effect, the console is automatically logged on when it is activated. For more information, see "Automatic LOGON" on page 60.

Resources, such as commands, MCS or SMCS consoles, and TSO terminals, also require RACF profiles. These profiles establish the access requirements for the resource — such as who can issue the command or use the console or terminal — and the level of security auditing your installation requires. For example, you might need to audit all uses of commands or want to audit only unauthorized uses of commands. For specific information, see "Defining Commands with RACF" on page 57 and "Defining Consoles with RACF" on page 59. For an example of defining a TSO/E terminal as a resource, see "Controlling Extended MCS Consoles Using RACF" on page 152.

You need to work with the RACF security administrator to set up the RACF profiles and options to implement your installation's security goals. *z/OS Security Server RACF Security Administrator's Guide* includes RACF-related information about securing access to system commands and consoles.

RACF Access Authorities

In RACF profiles that protect resources, the MCS authority "translates" to a RACF access authority. This RACF access authority is specified for a user or console in an access list of the resource profile and determines the command authority of the user or console.

MCS Authority	RACF Access Authority
MASTER	CONTROL
ALL(SYS,IO,CONS)	UPDATE
INFO	READ

These access authorities are the same for extended MCS console users. The security administrator can define resource profiles for MCS, SMCS and extended MCS consoles using RACF commands. (See "Controlling Extended MCS Consoles Using RACF" on page 152.)

Defining Users with RACF

Your installation's security policy determines how you define the operators, MCS consoles, or SMCS consoles for automatic logon. If your installation's security policy requires you to audit all operator commands according to the identity of the user, then all operators as individual users must be defined. If your installation uses the LOGON(AUTO) option in CONSOLxx to automatically log on MCS and SMCS consoles when they are activated, you must ensure that a user profile exists for each console to be logged on.

You can also grant access to commands to groups of operators. A RACF group defines a set of related individuals who have similar security requirements. Defining access authority by group minimizes changes to the RACF profiles when individual users change job responsibilities or leave a particular job.

To create profiles for operators, the RACF security administrator needs to know

- Who the operators are
- Which operators fall into groups with identical access requirements.

To create profiles for consoles to be automatically logged on, the RACF security administrator needs to know the names of the consoles defined in CONSOLxx.

Changes made to the access authority while a system is running may not take effect until the security data for the console(s) is reset in MVS. This occurs during LOGON for MCS or SMCS consoles and during MCSOPER ACTIVATE for EMCS consoles. For instance, if an active user is connected to a new group, the user must log off and then log back on again to have the authority associated with that new group.

Defining TSO/E Users of Extended MCS Consoles with RACF

Your TSO or RACF security administrator should define user profiles for all TSO/E users of extended MCS consoles. TSO/E logon can be controlled through TSO/E or RACF, and like operators, you can define TSO/E users by individual or group profiles. Your installation authorizes the TSO/E user to be able to issue the TSO/E CONSOLE command. This command initiates an extended MCS console session. For an example of how to define a TSO/E user to initiate an extended MCS console, see "Controlling Extended MCS Consoles Using RACF" on page 152.

Defining Commands with RACF

Your installation's security policy determines which commands you must protect. A RACF profile for the command in the OPERCMDS class protects the command. When an operator logs on to a console and issues an MVS command that requires a higher authority than the console allows, RACF can check the access list of the command profile to determine if the user is authorized to issue the command.

To link the command the operator issues with the profile that protects the command, MVS provides a construct, or structure, called a resource-name for each command.

The resource-name for an MVS command has the following parts:

```
MVS.command.command-qualifier.command object
```

where:

MVS

Is the high-level qualifier that defines the command as a system command. MVS is a required part of the resource-name. Subsystem commands use a different high-level qualifier, such as JES2 or JES3.

command

Specifies the command or a specific variation of the command. To protect an individual command, this part of the resource-name is required. It also allows you to control significant variations of a command separately. For example, FORCE without the ARM operand has a different effect than does FORCE with the ARM operand; you can thus specify either FORCE or FORCEARM to control the two uses separately.

command-qualifier

Specifies a subfunction of the command. This part of the resource-name is optional. It allows you to protect specific command subfunctions separately. For example, the following resource-name protects all functions of the TRACE command:

```
MVS.TRACE.**
```

In contrast, the following resource-names protect each function of the TRACE command separately:

```
MVS.TRACE.ST  
MVS.TRACE.MT  
MVS.TRACE.CT  
MVS.TRACE.STATUS
```

command-object

Specifies the object or target of the command. This part of the resource-name is optional. Examples of objects or targets include:

- The device on a CANCEL command
- The jobname on a MODIFY command
- The membername on a START command

Table 10 on page 62 defines the MVS commands and their corresponding resource-names. It also shows the RACF access authority associated with each command. To define resource profiles for system commands, the RACF security administrator can use the resource-names exactly as shown in Table 10 on page 62, or replace the optional fields with asterisks or, for *command-object*, specific values. In the command profile, the security administrator also defines the auditing requirements and the users or groups allowed to issue the command in the profile's access list.

When an operator issues an MVS command with a RACF profile, MVS determines the resource-name that matches the command and passes that resource-name to RACF. RACF uses the resource-name to locate the profile for the command and verifies that the operator is allowed to issue the command by checking the access list in the profile. If RACF authorizes the access, MVS processes the command; if RACF denies the access, MVS rejects the command. If your installation has user-written commands that you must protect, use the CMDAUTH macro; see *z/OS MVS Programming: Authorized Assembler Services Guide* and *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*.

To create profiles for MVS system commands that you do not have to change frequently, it is a good idea to end each name with two asterisks, which indicate that the profile protects all commands that match the specified portion of the resource-name, regardless of whether there are additional qualifiers or how many additional qualifiers there are. For example, use:

```
MVS.SET.**
```

to protect all SET commands with a single profile.

Defining Consoles with RACF

You can use a RACF profile in the CONSOLE class to determine which userids are authorized to log on to a particular console. The commands in the following example define a RACF profile for console CON1 and authorize userid CONSID1 to log on to that console.

```
RDEF CONSOLE CON1 UACC(NONE)
PERMIT CON1 CLASS(CONSOLE) ID(CONSID1) ACCESS(READ)
SETROPTS CLASSACT(CONSOLE)
```

Setting DEFAULT LOGON Requirements for MCS and SMCS Consoles

Once you have established the RACF profiles your installation requires, you use the LOGON keyword on the DEFAULT statement in CONSOLxx to establish your MCS console operator LOGON requirements. You can:

- Have the system automatically log each console on as the console is activated. Operators can log on but are not required to do so. See “Automatic LOGON” on page 60.
- Require each operator to log on to the system before issuing commands. See “Required LOGON” on page 61.
- Allow MCS console command authorization to control access to commands. See “Optional LOGON” on page 61.

To control how operators can log on to MCS consoles, use the following keyword on the DEFAULT statement in CONSOLxx:

LOGON Controls the logon for operators of MCS or SMCS consoles

Options you can specify for LOGON are as follows:

AUTO Specifies that the console is automatically logged on by its console name. In addition, operators can optionally log on to the console.

REQUIRED Specifies that operators must log on before the system allows them to enter commands. If a system includes SMCS consoles, LOGON(REQUIRED) is recommended.

OPTIONAL Specifies that operators can optionally log on to the console; otherwise, MCS console authority is in effect.

The LOGON keyword affects only full-capability display consoles.

It does not prevent the operator from receiving synchronous messages. Regardless of the LOGON value set on the DEFAULT statement, individual consoles can override the value. For more information, see “Setting LOGON Requirements for Individual MCS or SMCS Consoles”.

Setting LOGON Requirements for Individual MCS or SMCS Consoles

With z/OS, the LOGON keyword on the CONSOLE statement in CONSOLxx can override the system console LOGON default on the DEFAULT statement.

To control how operators can log on to specific MCS or SMCS consoles, specify the following keywords on the CONSOLE statement in CONSOLExx:

LOGON Controls the logon for operators of MCS and SMCS consoles

Options you can specify for LOGON are as follows:

AUTO Specifies that the console is automatically logged on.

- REQUIRED** Specifies that the console must be logged on before commands can be issued.
- OPTIONAL** Specifies that the console does not need to be logged on.
- DEFAULT** Specifies that the console is to use the LOGON value on the DEFAULT statement. If you specify DEFAULT and the DEFAULT statement does not contain a LOGON value, the system issues an error message and uses LOGON(OPTIONAL) for an MCS console and LOGON(REQUIRED) for SMCS.

For an SMCS console, see “Defining SMCS Consoles” on page 31.

Automatic LOGON

To control and audit command activity by console, specify LOGON (AUTO). When LOGON (AUTO) is in effect and RACF is active, the system automatically issues a LOGON for each MCS or SMCS console as the console is activated. The automatic LOGON uses the console name as the logon userid.

To ensure that the console is automatically logged on, the security administrator must define a user profile for each console by console name.

Your installation must define the name of the system console as a valid USERID to RACF. IBM recommends that if you plan to use LOGON (AUTO) for your installation, you define the system console in CONSOLxx and do not use the system default name as the name of the system console.

To define access requirements for the console, the security administrator defines a resource profile for the console in the RACF CONSOLE class. The CONSOLE class must be active when console resource profiles are used.

When automatic LOGON is in effect, operators can log on to the system but are not required to do so. The system issues an automatic LOGON for the console whenever RACF is active and the following conditions occur:

- The console is activated either during system initialization, as a result of the VARY command or if an SMCS console is logged on.
- The console is switched from message-stream or status display mode to full capability mode.
- An operator who had logged on issues the LOGOFF command.

Once the console is logged on, operators can use it to issue commands at the level defined for the userid. This could be the level defined in the OPERCMDS class for the userid, or lacking an OPERCMDS definition matching the command, the authority of the console (originally defined in CONSOLxx). If you have some consoles, perhaps those not in secure areas, that you want to require LOGONs, LOGON (AUTO) and RACF profiles allow you to control operator logon. If an operator wishes to issue a command requiring a higher level of authorization, and the operator (through RACF checking of OPERCMDS profiles) has the required level of authorization, the operator must log on to the console to be able to issue the command successfully. The operator authority (defined in the OPERCMDS class) then replaces the console authority. When the operator logs off, the system automatically issues the LOGON for the console name, thus reverting back to the original console authority.

When using LOGON(AUTO), you should ensure that at least one operator is logged on with master authority to be able to communicate with the system.

Required LOGON

To audit all command activity by operator userid or to control which commands individual operators may issue, specify LOGON(REQUIRED) on either the CONSOLE statement or the DEFAULT statement. Specifying LOGON(REQUIRED) is especially important for SMCS consoles. Before setting LOGON(REQUIRED), your installation must define RACF profiles for all operators and for the commands and consoles you want to protect. When protecting commands and consoles with RACF resource profiles, both the OPERCMDS and CONSOLE class must be active. Also, before setting LOGON(REQUIRED), your installation must define the name of the system console as a valid USERID to RACF. IBM recommends that, if you plan to use LOGON(REQUIRED) for your installation, you define the system console in CONSOLxx and do not use the system default name as the name of the system console.

When LOGON(REQUIRED) is in effect, all operators must log on before issuing commands, and your installation can limit the commands they can issue. If an operator tries to issue a command without logging on, the system rejects the command and issues a message. The system also rejects any command the operator is not authorized to issue. To change LOGON(REQUIRED) on the DEFAULT statement, you must re-IPL the system. You can use the VARY CN command to change LOGON(REQUIRED) on the CONSOLE statement.

During system initialization, the system accepts commands only from the master console (or the system console) until RACF is fully initialized and able to process LOGON requests, with one exception. If there is no master console present in the system before RACF is initialized, the system accepts the VARY MSTCONS command from any full-capability console to establish a master console. Allowing commands from the master console before RACF is fully initialized allows an operator to intervene if required to complete RACF initialization.

Once RACF is initialized, the LOGON prompt appears on all MCS display consoles. The LOGON prompt requires the operator to log on by supplying at least a userid and password. The LOGON prompt also appears:

- When a console changes from status display or message stream to full capability
- When the console is brought on line by a VARY command
- When an SMCS console is activated
- When the current operator logs off

When LOGON(REQUIRED) is in effect, no operator should leave the console unattended without first issuing the LOGOFF command. Issuing LOGOFF leaves the console in a secure, unattended state. For an MCS console, messages continue to appear on the console, but the system does not accept any command from that console until an operator logs on to the console. For SMCS consoles, the console session is terminated.

When using LOGON(REQUIRED), you should also ensure that at least one operator is logged on with master authority to be able to communicate with the system.

Optional LOGON

If you do not need special command auditing, you can specify LOGON(OPTIONAL). LOGON(OPTIONAL) allows console command authorization (defined by AUTH on the CONSOLE statement) to determine whether the system is to accept the command being issued on the console.

MVS Commands, RACF Access Authorities, and Resource Names

Table 10 lists all MVS commands, the RACF access authority associated with them, the RACF resource name for the profile, and any explanatory notes:

Table 10. MVS Commands, RACF Access Authorities, and Resource Names

Command/Keyword	Authority	Resource-Name
ACTIVATE	UPDATE	MVS.ACTIVATE
CANCEL device	UPDATE	MVS.CANCEL.DEV.device
CANCEL jobname	UPDATE	MVS.CANCEL.JOB.jobname
The previous command is for a job that is not a started task.		
CANCEL jobname.id CANCEL id	UPDATE	MVS.CANCEL.STC.mbrname.id
The previous command is for a started task for which an identifier is provided.		
CANCEL jobname	UPDATE	MVS.CANCEL.STC.mbrname.jobname
The previous command is for a started task for which an identifier was not provided. <i>mbrname</i> is the name of the member containing the JCL source.		
CANCEL jobname	UPDATE	MVS.CANCEL.ATX.jobname
The previous command is for APPC transaction programs.		
CANCEL U=userid	UPDATE	MVS.CANCEL.TSU.userid
CHNGDUMP	UPDATE	MVS.CHNGDUMP
CMDS DISPLAY	READ	MVS.CMDS.DISPLAY
CMDS SHOW	READ	MVS.CMDS.SHOW
CMDS REMOVE	CONTROL	MVS.CMDS.REMOVE
CMDS ABEND	CONTROL	MVS.CMDS.ABEND
CONFIG	CONTROL	MVS.CONFIG
CONTROL A	READ	MVS.CONTROL.A
Note: For CONTROL A, the access authority for all CONTROL commands except CONTROL M is normally READ, but the <i>L=cc</i> (console id) or <i>L=name</i> (console name) operand can change the access level. When <i>L=cc</i> or <i>L=name</i> specifies a console that is not full-capability and is not the issuing console, the access authority is UPDATE. When <i>L=cc</i> or <i>L=name</i> specifies a console that is full-capability and is not the issuing console, the access authority is CONTROL.		
For Control C, D, E, N, Q, S, T and V, see Control A.		
CONTROL C	READ	MVS.CONTROL.C
CONTROL D	READ	MVS.CONTROL.D
CONTROL E	READ	MVS.CONTROL.E
CONTROL M	CONTROL	MVS.CONTROL.M
CONTROL N	READ	MVS.CONTROL.N
CONTROL Q	READ	MVS.CONTROL.Q
CONTROL S	READ	MVS.CONTROL.S
CONTROL T	READ	MVS.CONTROL.T
CONTROL V	READ	MVS.CONTROL.V
DEVSERV	READ	MVS.DEVSERV
DISPLAY A	READ	MVS.DISPLAY.JOB
DISPLAY APPC	READ	MVS.DISPLAY.APPC

Table 10. MVS Commands, RACF Access Authorities, and Resource Names (continued)

Command/Keyword	Authority	Resource-Name
DISPLAY ASCH	READ	MVS.DISPLAY.ASCH
DISPLAY ASM	READ	MVS.DISPLAY.ASM
DISPLAY CNGRP	READ	MVS.DISPLAY.CNGRP
DISPLAY CONSOLES	READ	MVS.DISPLAY.CONSOLES
DISPLAY DMN	READ	MVS.DISPLAY.DMN
DISPLAY DLF	READ	MVS.DISPLAY.DLF
DISPLAY DUMP	READ	MVS.DISPLAY.DUMP
DISPLAY EMCS	READ	MVS.DISPLAY.EMCS
DISPLAY ETR	READ	MVS.DISPLAY.ETR
DISPLAY GRS	READ	MVS.DISPLAY.GRS
DISPLAY IOS	READ	MVS.DISPLAY.IOS
DISPLAY IPLINFO	READ	MVS.DISPLAY.IPLINFO
DISPLAY JOBS	READ	MVS.DISPLAY.JOB
DISPLAY LOGREC	READ	MVS.DISPLAY.LOGREC
DISPLAY MMS	READ	MVS.DISPLAY.MMS
DISPLAY M	READ	MVS.DISPLAY.M
DISPLAY MPF	READ	MVS.DISPLAY.MPF
DISPLAY NET	READ	MVS.DISPLAY.NET
DISPLAY OPDATA	READ	MVS.DISPLAY.OPDATA
DISPLAY PARMLIB	READ	MVS.DISPLAY.PARMLIB
DISPLAY PFK	READ	MVS.DISPLAY.PFK
DISPLAY PROD	READ	MVS.DISPLAY.PROD
DISPLAY PROG	READ	MVS.DISPLAY.PROG
DISPLAY R	READ	MVS.DISPLAY.R
DISPLAY RTLS	READ	MVS.DISPLAY.RTLS
DISPLAY SLIP	READ	MVS.DISPLAY.SLIP
DISPLAY SMF	READ	MVS.DISPLAY.SMF
DISPLAY SMS	READ	MVS.DISPLAY.SMS
DISPLAY SSI	READ	MVS.DISPLAY.SSI
DISPLAY SYMBOLS	READ	MVS.DISPLAY.SYMBOLS
DISPLAY T	READ	MVS.DISPLAY.TIMEDATE
DISPLAY TP	READ	MVS.DISPLAY.TCAM
DISPLAY TRACE	READ	MVS.DISPLAY.TRACE
DISPLAY TS	READ	MVS.DISPLAY.JOB
DISPLAY U	READ	MVS.DISPLAY.U
DISPLAY WLM	READ	MVS.DISPLAY.WLM
DISPLAY XCF	READ	MVS.DISPLAY.XCF
DUMP	CONTROL	MVS.DUMP
DUMPDS	UPDATE	MVS.DUMPDS
FORCE device	CONTROL	MVS.FORCE.DEV.device

Table 10. MVS Commands, RACF Access Authorities, and Resource Names (continued)

Command/Keyword	Authority	Resource-Name
FORCE jobname	CONTROL	MVS.FORCE.JOB.jobname
The previous command is for a job that is not a started task.		
FORCE jobname.id FORCE id	CONTROL	MVS.FORCE.STC.mbrname.id
The previous command is for a started task for which an identifier was provided.		
FORCE jobname	CONTROL	MVS.FORCE.STC.mbrname.jobname
The previous command is for a started task for which an identifier was not provided. <i>mbrname</i> is the name of the member containing the JCL source.		
FORCE U=userid	CONTROL	MVS.FORCE.TSU.userid
FORCE device,ARM	CONTROL	MVS.FORCEARM.DEV.device
FORCE jobname,ARM	CONTROL	MVS.FORCEARM.JOB.jobname
The previous command is for a job that is not a started task.		
FORCE [jobname.]identifier,ARM	CONTROL	MVS.FORCEARM.STC.mbrname.id
The previous command is for a started task for which an identifier was provided.		
FORCE jobname,ARM	CONTROL	MVS.FORCEARM.STC.mbrname.jobname
The previous command is for a started task for which an identifier was not provided. <i>mbrname</i> is the name of the member containing the JCL source.		
FORCE U=userid,ARM	CONTROL	MVS.FORCEARM.TSU.userid
HALT EOD	UPDATE	MVS.HALT.EOD
HALT NET	UPDATE	MVS.HALT.NET
HALT TP	UPDATE	MVS.HALT.TCAM
HOLD	UPDATE	MVS.HOLD.TCAM
IOACTION	CONTROL	MVS.IOACTION
LIBRARY	UPDATE	MVS.LIBRARY
LOG	READ	MVS.LOG
MODE	UPDATE	MVS.MODE
MODIFY jobname	UPDATE	MVS.MODIFY.JOB.jobname
The previous command is for a job that is not a started task.		
MODIFY userid	UPDATE	MVS.MODIFY.JOB.userid
MODIFY jobname MODIFY jobname.id MODIFY id	UPDATE	MVS.MODIFY.STC.mbrname.id
The previous command is for a started task for which an identifier was provided.		
MODIFY jobname	UPDATE	MVS.MODIFY.STC.mbrname.jobname
The previous command is for a started task for which an identifier was not provided. <i>mbrname</i> is the name of the member containing the JCL source.		
Note: MODIFY might actually affect more than one job. For example:		
<ul style="list-style-type: none"> • If START ABC.DEF and START ABC.GHI are issued, MODIFY ABC.* affects both jobs, and one authorization request is issued for each. • If the START ABC command is issued twice, two started tasks named ABC start running on the system. MODIFY ABC affects both jobs, and one authorization request is issued for each. 		
MONITOR	READ	MVS.MONITOR
MOUNT	UPDATE	MVS.MOUNT

Table 10. MVS Commands, RACF Access Authorities, and Resource Names (continued)

Command/Keyword	Authority	Resource-Name
MSGRT	READ	MVS.MSGRT
PAGEADD	UPDATE	MVS.PAGEADD
PAGEDEL	UPDATE	MVS.PAGEDEL
QUIESCE	CONTROL	MVS.QUIESCE
RELEASE	UPDATE	MVS.RELEASE.TCAM
REPLY	READ	MVS.REPLY
RESET	UPDATE	MVS.RESET
RESET CN	CONTROL	MVS.RESET.CN
ROUTE system	READ	MVS.ROUTE.CMD.system
Note: When a system name is specified on the ROUTE command, <i>system</i> is the name of the system that is the target of the command.		
ROUTE *ALL	READ	MVS.ROUTE.CMD.ALLSYSTEMS
ROUTE *OTHER	READ	MVS.ROUTE.CMD.OTHERSYSTEMS
ROUTE sysgrpname	READ	MVS.ROUTE.CMD.sysgrpname
ROUTE (sys1,....,sysN)	READ	MVS.ROUTE.CMD.sys1 : MVS.ROUTE.CMD.sysN
ROUTE (group1,....,groupN)	READ	MVS.ROUTE.CMD.group1 : MVS.ROUTE.CMD.groupN
SEND	READ	MVS.SEND
SET APPC	UPDATE	MVS.SET.APPC
SET ASCH	UPDATE	MVS.SET.ASCH
SET CLOCK	UPDATE	MVS.SET.TIMEDATE
SET CNGRP	UPDATE	MVS.SET.CNGRP
SET DAE	UPDATE	MVS.SET.DAE
SET DATE	UPDATE	MVS.SET.TIMEDATE
SET GRSRNL	UPDATE	MVS.SET.GRSRNL
SET ICS	UPDATE	MVS.SET.ICS
SET IOS	UPDATE	MVS.SET.IOS
SET IPS	UPDATE	MVS.SET.IPS
SET MMS	UPDATE	MVS.SET.MMS
SET MPF	UPDATE	MVS.SET.MPF
SET OPT	UPDATE	MVS.SET.OPT
SET PFK	UPDATE	MVS.SET.PFK
SET PROG	UPDATE	MVS.SET.PROG
Note: For examples of how to define RACF profiles for this command, see “Using RACF to Control APF Lists” on page 154.		

Table 10. MVS Commands, RACF Access Authorities, and Resource Names (continued)

Command/Keyword	Authority	Resource-Name
SET RESET	UPDATE	MVS.SET.TIMEDATE
SET RTLS	UPDATE	MVS.SET.RTLS
SET SCH	UPDATE	MVS.SET.SCH
SET SLIP	UPDATE	MVS.SET.SLIP
SET SMF	UPDATE	MVS.SET.SMF
SET SMS	UPDATE	MVS.SET.SMS
SETDMN	UPDATE	MVS.SETDMN.DMN
SETETR	UPDATE	MVS.SETETR.ETR
SETGRS MODE=STAR	UPDATE	MVS.SETGRS.MODE.STAR
SETIOS	UPDATE	MVS.SETIOS.IOS
SETLOAD	UPDATE	MVS.SETLOAD.LOAD
SETLOGRC	CONTROL	MVS.SETLOGRC.LOGRC
SETPROG	UPDATE	MVS.SETPROG
<p>Note: For examples of how to define RACF profiles for this command, see the following topics in Chapter 5, “Examples and MVS Planning Aids for Operations” on page 149; for SETPROG APF , see “Using RACF to Control APF Lists” on page 154; for SETPROG EXIT, see “Using RACF to Control Dynamic Exits” on page 158; for SETPROG LNKLST, see “Using RACF to Control LNKLST Concatenations” on page 164; and for SETPROG LPA, see “Using RACF to Control Dynamic LPA” on page 170.</p>		
SETSMF	UPDATE	MVS.SETSMF.SMF
SETSMS	UPDATE	MVS.SETSMS.SMS
SETSSI ADD	CONTROL	MVS.SETSSI.ADD.subname
SETSSI ACTIVATE	CONTROL	MVS.SETSSI.ACTIVATE.subname
SETSSI DEACTIVATE	CONTROL	MVS.SETSSI.DEACTIVATE.subname
SETXCF	UPDATE	MVS.SETXCF.XCF
SLIP	UPDATE	MVS.SLIP
START mbrname[.identifier]	UPDATE	MVS.START.STC.mbrname[.id]
<p>The previous command is for a started task for which an identifier was provided. <i>mbrname</i> is the name of the member containing the JCL source.</p>		
START mbrname,JOBNAME=jobname	UPDATE	MVS.START.STC.mbrname.jobname
<p>The previous command is for a started task for which an identifier was not provided. <i>mbrname</i> is the name of the member containing the JCL source.</p>		
START commands that use one or more of the following keywords: <ul style="list-style-type: none"> • DSN or DSNAME • DISP • PROTECT 	UPDATE	<p>The resource name substitutes DDALERT for one or more of the keywords.</p> <p>MVS.START.jobname.qualifier.DDALERT</p>
<p>An example of the previous MVS START command is as follows:</p>		
START jobname.qualifier,DSN=dsname.qualifier,DISP=SHR		
STOP jobname	UPDATE	MVS.STOP.JOB.jobname
<p>The previous command is for a job that is not a started task.</p>		
STOP userid	UPDATE	MVS.STOP.JOB.userid

Table 10. MVS Commands, RACF Access Authorities, and Resource Names (continued)

Command/Keyword	Authority	Resource-Name
STOP jobname STOP jobname.id STOP id	UPDATE	MVS.STOP.STC.mbrname.id
The previous command is for a started task for which an identifier was provided. <i>mbrname</i> is the name of the member containing the JCL source.		
STOP jobname	UPDATE	MVS.STOP.STC.mbrname.jobname
The previous command is for a started task for which an identifier was not provided. <i>mbrname</i> is the name of the member containing the JCL source. Note: STOP might actually affect more than one started task if more than one unit of work with the same name is active at the same time. If so, there is one call to RACF for command authorization for each unit of work.		
STOPMN	READ	MVS.STOPMN
STOPTR	READ	MVS.STOPTR
SWAP	UPDATE	MVS.SWAP
SWITCH CN	CONTROL	MVS.SWITCH.CN.cnme1.cnme2
Note: For SWITCH CN, <i>cnme1</i> is the name of the console whose attributes are being switched. <i>cnme2</i> is the name of the console that receives the attributes during the switch.		
SWITCH SMF	UPDATE	MVS.SWITCH.SMF
TRACE CT	UPDATE	MVS.TRACE.CT
TRACE MT	CONTROL	MVS.TRACE.MT
TRACE ST	UPDATE	MVS.TRACE.ST
TRACE STATUS	UPDATE	MVS.TRACE.STATUS
TRACK	READ	MVS.TRACK
UNLOAD	UPDATE	MVS.UNLOAD
VARY CN	UPDATE	MVS.VARY.CN
VARY CN,ACTIVATE	READ	MVS.VARY.CN
Note: Issue VARY CN,ACTIVATE only from the system console.		
VARY CN,AUTH	CONTROL	MVS.VARYAUTH.CN
VARY CN,DEACTIVATE	READ UPDATE	MVS.VARY.CN
Note: For the VARY CN,DEACTIVATE command, READ applies only when that command is issued from the system console; otherwise, UPDATE applies.		
VARY CN,LOGON	CONTROL	MVS.VARYLOGON.CN
VARY CN,LU	CONTROL	MVS.VARYLU.CN
VARY CONSOLE	UPDATE	MVS.VARY.CONSOLE
VARY CONSOLE,AUTH	CONTROL	MVS.VARYAUTH.CONSOLE
VARY GRS	CONTROL	MVS.VARY.GRS
VARY HARDCPY	CONTROL	MVS.VARY.HARDCPY
VARY MSTCONS	CONTROL READ	MVS.VARY.MSTCONS
Note: For VARY MSTCONS, CONTROL applies after RACF is initialized. READ applies during a no master console condition to allow an operator to establish a master console if necessary.		
VARY NET	UPDATE	MVS.VARY.NET
VARY OFFLINE	UPDATE	MVS.VARY.DEV

Table 10. MVS Commands, RACF Access Authorities, and Resource Names (continued)

Command/Keyword	Authority	Resource-Name
Note: If VARY CN,OFFLINE is specified, the rules for VARY CN apply (the system checks for UPDATE access to MVS.VARY.CN, not MVS.VARY.DEV).		
VARY OFFLINE,FORCE	CONTROL	MVS.VARYFORCE.DEV
VARY ONLINE	UPDATE	MVS.VARY.DEV
Note: If VARY CN,ONLINE is specified, the rules for VARY CN apply (the system checks for UPDATE access to MVS.VARY.CN, not MVS.VARY.DEV).		
VARY ONTP	UPDATE	MVS.VARY.TCAM
VARY OFFTP	UPDATE	MVS.VARY.TCAM
VARY PATH	UPDATE	MVS.VARY.PATH
VARY SMS	UPDATE	MVS.VARY.SMS
VARY WLM	CONTROL	MVS.VARY.WLM
VARY XCF	CONTROL	MVS.VARY.XCF
WRITELOG	READ	MVS.WRITELOG

Handling Unrecognized Commands

To handle MVS system commands that operators might enter but which the system does not recognize, create an MVS.UNKNOWN profile for RACF auditing, and define a universal access authority of READ.

Some command processors, including CANCEL, FORCE, MODIFY, and STOP, will also use this resource when auditing unknown tasks or address spaces.

When you specify auditing, the auditing records contain the full text of the commands as entered.

Other Ways to Control Command Authority for Consoles

If you do not use RACF to override MCS or SMCS console authority, you can authorize specific commands issued from an MCS or SMCS console through the command installation exit. You can specify command installation exits in MPFLSTxx. See “Command Installation Exits” on page 122.

Planning Console Functions for Operators

CONSOLxx allows you to plan MCS and SMCS console screen functions for your operators. How operators do work on consoles is affected by the following factors:

- The capability of the console to send commands and receive messages or status displays
- The volume of messages on the console screen
- How messages roll or wrap on the screen and how quickly they move or are overlaid
- How easily operators can delete unnecessary messages from the screen
- Out-of-line display areas for system status displays
- Message format and the information that appears in message displays
- How easily and efficiently operators can enter commands

In CONSOLxx, you can establish the use of an MCS display console. The USE attribute for the console controls whether an operator can send commands as well as receive messages and status. Your master console must be a full-capability console able to accept commands and receive messages. On the other hand, an output-only console is useful for an operator who only needs to monitor messages or status displays.

How operators delete unwanted messages from the console screen has a direct effect on the work they do. Message traffic, especially large numbers of unsolicited messages or certain kinds of informational messages, can be controlled through MPF message suppression. Important action messages that require a specific operator response can be retained for later viewing by operators if AMRF is active. However, operators need to be able to respond quickly to action messages and remove unnecessary messages from the screen. Setting up screen functions to help operators handle messages efficiently is an important part of console planning.

In CONSOLxx, you can establish whether messages roll or wrap on the console screen, whether action messages are to be isolated from other types of messages, or how operators can manually delete messages by letting them verify deletion requests. You can also specify a hold-mode function for consoles in roll, roll-deletable, or wrap mode so that the operator can “freeze” the screen to view an important message.

You can define out-of-line display areas for an MCS or SMCS console. An out-of-line display area is a specified part of the screen that can receive status displays separate from the messages that appear. Output from certain commands like TRACK, CONFIG, DISPLAY, or DEVSERV can be directed to these specific console areas on the screen for operators to view.

You can control the message format so that certain information can appear or be suppressed. Examples: you can control whether the jobname or system name should accompany a message, and whether status displays contain or suppress information about certain events like job starts or stops, when a data set is freed, or information about TSO/E users.

Finally, you can define PFKs or control the multiple entry of commands for MCS or SMCS consoles. Establishing PFKs for your MCS or SMCS consoles allows you to control command functions for operators so that they can enter frequent commands quickly and easily from their consoles. You can also define a command delimiter for MCS or SMCS consoles so that operators can enter multiple commands on the command line.

How to Control the Use of an MCS Console

The devices that you have defined as MCS consoles can function as:

- Full-capability consoles
- Status-display consoles
- Message stream consoles

The devices that you have defined as SMCS consoles can function only as full-capability consoles.

Full capability consoles

A full-capability console has both input and output capability; the console can be used both to enter commands and to receive status displays and messages. You

can also control how messages move on the screen of a full capability console and how operators can delete those messages as they fill the screen. For example, you can specify that messages roll off the screen as the screen fills (roll or roll-deletable mode) or that messages wrap, that is, overlay old existing messages on the screen (wrap mode). For consoles in roll, roll-deletable, or wrap mode, you can define HOLDMODE that allows operators to freeze the screen to view messages.

With all modes but wrap mode, you can divide the screen of a full-capability console so that part of it receives general messages and the other part receives status displays. When a status display is not on the screen, MCS uses the status display area for general messages.

Status Display Consoles

A status display console has output capability only; it cannot be used to enter commands. The system uses the screen to receive status displays.

Message Stream Consoles

A message stream console also has output capability only; it cannot be used to enter commands. The system uses the screen to present general messages.

Note: In this book, the term output-only mode refers to status display mode and message stream mode.

An operator can change a full-capability console to message stream or status display. When the change occurs, the PFK display line, the instruction line, and the entry area are incorporated into the message area or the display area. Once a display console enters message stream or status display mode, it can accept no more input; you must use another console to enter commands.

Examples of MCS Console Screens

Figure 8 shows screens of a full-capability, status display, and message-stream console:

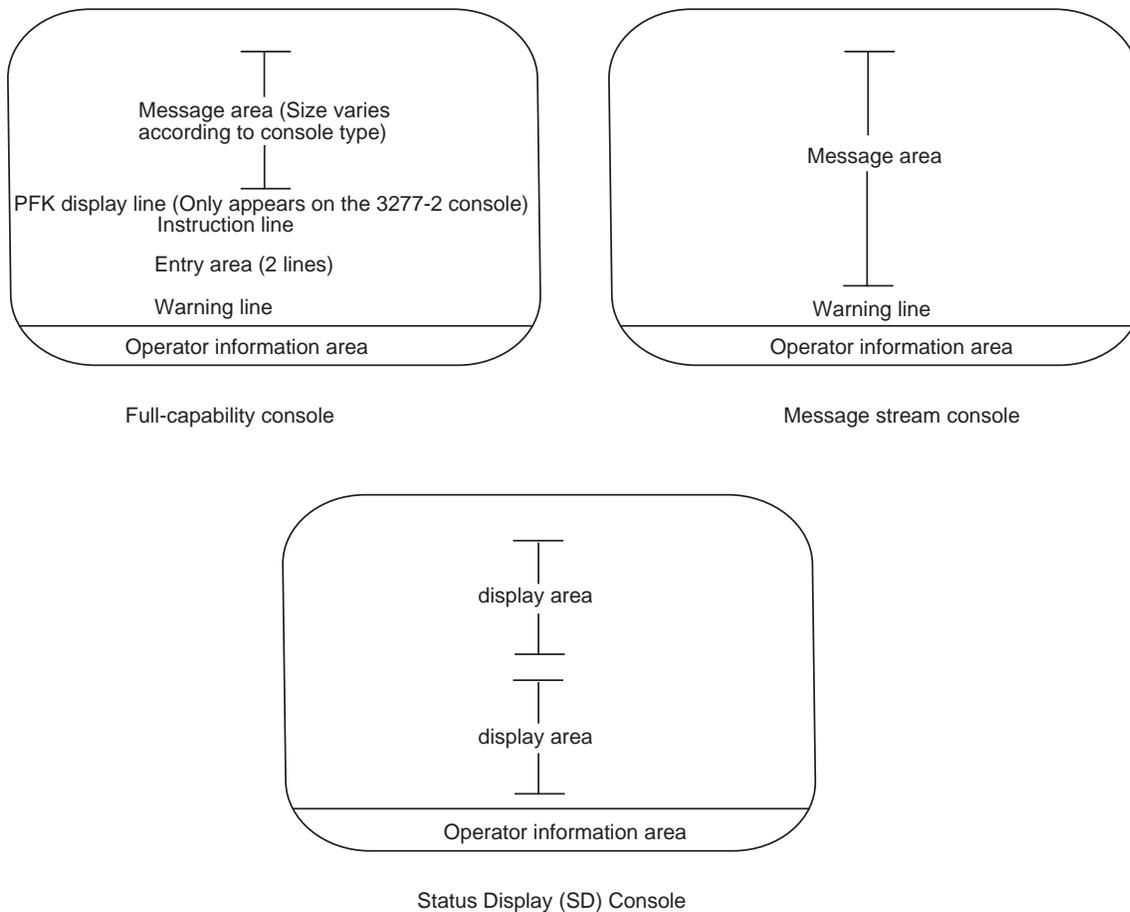


Figure 8. Screen Formats of a Full-Capability, Status Display, and Message Stream Console

Message area is that part of the display where messages appear. Display area is that part of the screen where status displays appear.

On full-capability console screens of 3277-2 models, the PFK display line displays the numbers of the PFKs to select with the selector pen.

On all full-capability console screens, the instruction line displays console screen control messages in response to certain actions (for example, if the operator makes a CONTROL command error). The entry area (1 or 2 lines) allows operators to enter commands on full-capability console screens.

The warning line on full-capability and message stream console screens warns the operator of conditions that could require action (for example, when the message area is full and one or more messages is waiting to appear.)

Operator information on the status of the console appears on some console screens in the operator information area.

Defining the USE Attribute

Use the following keyword on the CONSOLE statement to define how to use a display console:

USE Controls how the display console is used:

The following are options for USE:

- FC** Defines a full-capability console able to enter commands and receive status displays and messages
- MS** Defines a message stream console
- SD** Defines a status display console.

If a console is an input/output device, the default operating mode is full-capability (FC).

SMCS consoles must specify the FC option.

Message Display and the Full-Capability Console Screen

As programs execute during system operation, the message area of a full-capability console screen fills with messages that operators might need to delete. The system can automatically remove messages from a console screen, or operators can make room for more messages by manually deleting non-action messages and messages for which action has been taken.

You can define automatic message deletion mode for an MCS or SMCS console. With automatic message deletion, the system removes old messages without operator assistance as the screen fills. "Specifying Automatic Message Deletion for MCS or SMCS Consoles" describes how you can control the automatic message deletion mode for a console. It describes automatic mode, roll mode, roll-deletable mode, and wrap mode. To handle frequent messages that appear on full-capability MCS or SMCS consoles, it is a good idea to use roll, roll-deletable, or wrap mode. Specifying one of these automatic message deletion modes prevents messages from backing up on system queues while the system waits for screen space.

With roll, roll-deletable, and wrap modes, you can also specify that the system freeze the console screen for easier viewing of messages. "Temporarily Suspending the Screen Roll" on page 76 describes HOLDMODE, the console option that allows the operator to freeze the console screen to view messages.

Operators can also manually delete non-action messages from a full-capability console screen. You can control whether an operator must verify a manual deletion request to make changes or corrections. "Manual Deletion of Messages" on page 77 describes how operators can manually remove messages from a console screen.

You can also activate the action message retention facility (AMRF) so operators can retrieve messages that have disappeared from the console screen. The action message retention facility helps operators deal with the heavy volume of message traffic in a system or sysplex. "Retaining Messages" on page 116 describes message retention of action messages.

Specifying Automatic Message Deletion for MCS or SMCS Consoles

Use the following keyword on the CONSOLE statement in CONSOLxx to control automatic message deletion:

- DEL** Specifies the mode for message deletion

The following are options for DEL:

- Y** Specifies automatic deletion mode

- N** Specifies that messages can only be manually deleted from the console screen
- RD** Specifies roll-deletable mode; roll-deletable mode is the default.
- R** Specifies roll mode
- W** Specifies wrap mode

Automatic mode

In ***automatic mode***, messages are removed whenever the message area becomes full, or when a status display is overlaying messages in the bottom portion of the message area. Flagged messages are the only messages removed under automatic mode. These messages include:

- Action messages for which the action has been taken
- System or problem program messages that are marked deletable by the issuer
- Messages that are indicated as deletable at job step end
- WTOR messages that have been answered
- WTOR messages that have not been answered but that are associated with a job step that has ended

Roll and roll-deletable modes

In ***roll mode***, a specified number of messages are removed (or “rolled off”) when a specified time interval elapses. Roll mode is particularly useful for monitoring heavy message traffic.

Roll-deletable mode is the same as roll mode except that action messages are not removed; they accumulate at the top of your screen. The operator can then delete the action messages one at a time, either by using the CONTROL E command or placing the cursor or light pen on the “*” or “@” that precedes the message and pressing ENTER.

Use the following keywords to control the rate of rolling for a console screen in roll or roll-deletable mode:

- RNUM** Controls the number of lines per screen roll; the default is 5 lines per roll.
- RTME** Controls the rate of the screen roll; the default is 2 seconds between rolls.

To request that roll mode go into effect and that two messages be rolled every second, code the following parameters on the CONSOLE statement for the console:
DEL(R) RNUM(2) RTME(1)

For roll mode and roll-deletable modes, messages are not numbered on the screen. Instead, a two-digit number appears in the first new message line after each screen roll. This number indicates the number of messages waiting for display, and includes any messages hidden by the status display information. If the number of lines waiting for display is more than 99, AA appears in the first new message line.

Note: For 3290 consoles, do not specify 1/4 second or 1/2 second for RTME. Specifying either this value can affect the performance of the console.

Adjusting RNUM and RTME values

Because system workload can vary, you might want to change RNUM and RTME values to meet the needs of your installation. To illustrate how you would code the values in CONSOLxx, assume for a console named TAPEMSG, that you want to define roll-deletable mode with five messages being deleted every 10 seconds. Also, you want each message to display the system name from where the messages are issued. Code the statement in CONSOLxx as follows:

```
CONSOLE DEVNUM(0C6) NAME(TAPEMSG) DEL(RD) RTME(10) RNUM(5) MFORM(S)
```

For a description of MFORM, see “Controlling the Format of Messages and Status Information on Console Screens” on page 82.

Wrap mode

In ***wrap mode*** operators can view messages without having messages move off the screen. When the screen is full, new messages overlay older messages. As the messages begin to fill up the screen in wrap mode, they appear from top to bottom on the console screen with the old messages on the top and the newer messages on the bottom.

The console screen still preserves the instruction line, entry area, and warning line. (See Figure 10 on page 75.) However, when the screen is filled, the messages themselves do not roll off the screen. Instead a highlighted separator line that separates the last displayed message from the newest displayed message moves to indicate the new boundary between old and new messages. (A two-digit number at the beginning of the separator line indicates the number of messages waiting for display.)

When a new message cannot fit on the screen, the separator line overlays the oldest message at the top of the screen and the new message appears at the bottom.

As new messages are added, the separator line continues to move and overlay the next oldest message on the screen with the newest message always appearing above the line:

```

IEE600I REPLY TO 01 IS:NONE
ICH501I -- RACF IS NOT ACTIVE --
IEF677I WARNING MESSAGE(S) FOR JOB JES2      ISSUED
*02 $HASP426 SPECIFY OPTIONS -- JES2 SP 3.1.1
ISG011I SYSTEM SYSTEM2 - JOINING GRS COMPLEX
ISG004I GRS COMPLEX JOINED BY SYSTEM2
CSV210I LIBRARY LOOKASIDE INITIALIZED
*IEE352A SMF ENTER DUMP FOR SYS1.MANA ON PAGE98
04 -----
IEA180I USING IBM DEFAULT VALUE PFK DEFINITIONS.
      NO PFK TABLES REQUESTED
IKJ712I DEFAULT VALUES WERE USED FOR TEST
IKJ712I DEFAULT VALUES WERE USED FOR PLATCMD
IRA600I SRM CHANNEL DATA NOW AVAILABLE FOR ALL SRM FUNCTIONS
ICH508I ACTIVE RACF EXITS: ICHDEX01
ICH509I SYSRACF DD STATEMENT NOT SPECIFIED INMSTRJCL OR
      ALLOCATION FAILURE FOR RACF DATA SET
*01 ICH502A SPECIFY NAME FOR PRIMARY RACF DATASET SEQUENCE 091 OR
      'NONE'
      R 1,none

IEE612I CN=MASTER   DEVNUM=0FE  SYS=SYSTEM1  CMDSYS=SYSTEM1  USERID=JIM
IEE163I MODE=W

```

Figure 9. Example of a Full Wrap Mode Screen

Figure 10 shows the same screen when a new message (IEE366I) appears:

```

IEE600I REPLY TO 01 IS:NONE
ICH501I -- RACF IS NOT ACTIVE --
IEF677I WARNING MESSAGE(S) FOR JOB JES2      ISSUED
*02 $HASP426 SPECIFY OPTIONS -- JES2 SP 3.1.1
ISG011I SYSTEM SYSTEM2 - JOINING GRS COMPLEX
ISG004I GRS COMPLEX JOINED BY SYSTEM2
CSV210I LIBRARY LOOKASIDE INITIALIZED
*IEE352A SMF ENTER DUMP FOR SYS1.MANA ON PAGE98
IEE366I NO SMF DATASETS AVAILABLE--DATA BEING BUFFERED TIME*13:42:42
03 -----
IKJ712I DEFAULT VALUES WERE USED FOR TEST
IKJ712I DEFAULT VALUES WERE USED FOR PLATCMD
IRA600I SRM CHANNEL DATA NOW AVAILABLE FOR ALL SRM FUNCTIONS
ICH508I ACTIVE RACF EXITS: ICHDEX01
ICH509I SYSRACF DD STATEMENT NOT SPECIFIED INMSTRJCL OR
      ALLOCATION FAILURE FOR RACF DATA SET
*01 ICH502A SPECIFY NAME FOR PRIMARY RACF DATASET SEQUENCE 091 OR
      'NONE'
      R 1,none

IEE612I CN=MASTER   DEVNUM=0FE  SYS=SYSTEM1  CMDSYS=SYSTEM1  USERID=JIM
IEE163I MODE=W

```

Figure 10. Example of the Wrap Mode Screen after the Next Wrap

Specifying RTME for WRAP mode

You can specify RTME for wrap mode to update the screen. As long as the device is not a 3290, an RTME value of 1/4 or 1/2 second is good for a console in wrap mode. To specify wrap mode for a console that displays a message every 1/2 second, code the following on the CONSOLE statement for the console:

```
DEL(W) RTME(1/2)
```

Restrictions using Wrap Mode

In wrap mode, new messages overlay WTORs and action messages; unlike these messages in roll deletable mode, WTORs and action messages are not retained on the screen. Note also that for a console screen in wrap mode, you cannot use the following commands:

- CONTROL A to define or change out-of-line display areas
- CONTROL D,N,HOLD to number and hold messages
- CONTROL E,nn(,nn) to remove specified lines from the screen
- CONTROL E,F to remove flagged messages from the screen
- CONTROL E,N to remove message numbers from the screen
- CONTROL T to refresh the TRACK command
- TRACK to enter the TRACK command

Note that using CONTROL E,SEG to eliminate groups of messages from a console screen in wrap mode clears the entire screen of messages. Consoles in wrap mode do not use out-of-line areas.

Temporarily Suspending the Screen Roll

Operators might need to suspend a rolling screen of messages to copy information from the screen or consult a messages reference book. To suspend a rolling screen of messages, you can use HOLDMODE to control how operators temporarily suspend or hold screens when in roll, roll-deletable, or wrap mode.

Use the following keyword on the DEFAULT statement of CONSOLxx:

HOLDMODE Specifies that you want hold mode for MCS consoles in the system; if YES, your operators can temporarily hold the message screen by entering nulls on the command line or by pressing the enter key. If NO, operators cannot use this method to hold messages on the screen.

When hold mode is in effect, an operator can press enter to hold the screen and read messages. The following shows the bottom of the console screen when HOLDMODE is in effect:

```
IEE163I  MODE = HELD
```

The following shows the bottom of the console screen in HOLDMODE when messages are waiting to be displayed:

```
IEE163I  MODE = HELD          IEE159I MESSAGE WAITING
```

The following shows the bottom of the console screen in HOLDMODE when messages are overlaid by a status display:

```
IEE163I  MODE = HELD          IEE160I UNVIEWABLE MESSAGE
```

To release the screen and return to roll, roll/deletable, or wrap mode, the operator presses enter again. HOLDMODE has system scope; if you define HOLDMODE in CONSOLxx for a system in a sysplex, it applies only to the MCS or SMCS consoles on that system. If messages are backed up on a system when a console is in hold mode, hold mode for the console is released.

Operators can also suspend the console screen using PFKs if the IBM defaults for console PFKs are in effect:

1. Press PFK 5 to stop messages from rolling. (At IPL, PFK 5 is assigned the command CONTROL S,DEL=N.)
2. Press PFK 6 to place the screen in roll-deletable mode and prevent message backup. (At IPL, PFK 6 is defined as CONTROL S,DEL=RD.)

Comparison of Roll, Roll-Deletable, Wrap Modes, and HOLDMODE

Table 11 shows a comparison of roll mode, roll-deletable mode, and wrap mode, and options you can specify including HOLDMODE.

Table 11. Comparison of Roll, Roll-deletable, and Wrap Mode

Mode	HOLDMODE allowed as option	RTME allowed as option	RNUM allowed as option	How action messages are handled
Roll	Yes	Yes	Yes	Roll off the screen after RTME interval
Roll-deletable	Yes	Yes	Yes	Accumulate at top of screen. Operator removes them.
Wrap	Yes	Yes	No	Overlaid by new messages

Manual Deletion of Messages

Operators can **manually** delete messages from the screen using the CONTROL E command, the cursor, or the selector pen. If your operators need to obtain screen space quickly, they can manually delete non-action messages as follows:

- Use the cursor or selector pen
- Use the CONTROL E command to select groups of messages to delete

Message deletion, like command entry, can be either **conversational** or **nonconversational**. In conversational mode, the operator must verify the deletion request using the cursor, selector pen, or CONTROL E command before the system can remove the messages from the screen. When the operator performs one of these functions, the screen displays the messages to be deleted and asks for verification. The operator can then make corrections or changes, if necessary, and then press the enter key.

In nonconversational mode, the operator can use the cursor, selector pen, or CONTROL E command to manually delete messages; however, the deletion requests do not need to be verified and messages are immediately deleted when the operator performs the function. This procedure minimizes operator intervention.

Use the following keyword on the CONSOLE statement to control conversational mode for the console:

CON Specifies whether you want conversational mode

In conversational mode where the operator must verify a deletion request, the procedure to manually delete non-action messages is as follows:

Manual deletion - operator must verify	Using a selector pen or cursor	Using the CONTROL command
If CON(Y)	<ol style="list-style-type: none"> 1. Place the pen or cursor on any part of a non-action message 2. Press ENTER key 3. Vertical lines appear in position 3 of the non-action message and each non-action message above it. In the instruction line, the following message appears: IEE157E DELETION REQUESTED 4. Message line numbers appear on screen. CONTROL E command appears on command line indicating the request. 5. Verify the request, make changes, if necessary, and press the ENTER key To cancel the request, enter CANCEL. 	<ol style="list-style-type: none"> 1. If DEL(N), enter CONTROL D,N to display message line numbers 2. Enter CONTROL E,<i>line number</i> of non-action message to delete (on CONTROL E, you can also specify a range of lines to delete, a SEG value, or F to remove all flagged messages. See <i>z/OS MVS System Commands</i>.) 3. CONTROL E command appears on command line as entered 4. Verify the request, make changes, if necessary, and press the ENTER key

In non-conversational mode, the procedure to manually delete messages is as follows:

Manual deletion - no verification	Using a selector pen or cursor	Using the CONTROL command
If CON(N)	<ol style="list-style-type: none"> 1. Place the pen or cursor on any part of a non-action message 2. Press the ENTER key The non-action message and all non-action messages above it are deleted from the screen. 	<ol style="list-style-type: none"> 1. Enter CONTROL D,N to display message line numbers (if DEL(N)) 2. Enter CONTROL E,<i>line number</i> of non-action message to delete (on CONTROL E, you can also specify a range of lines to delete, a SEG value, or F to remove all flagged messages. See <i>z/OS MVS System Commands</i>.) The messages are deleted from the screen.

How Operators Specify Message Numbering

If the console is not in automatic deletion mode, operators can control whether they want the message line numbers on the console screen. With message line

numbers, they can more easily determine the range of messages to delete using CONTROL E or CONTROL E,SEG. Consecutive numbers in positions one and two appear for each message line, including continuation lines, for all message area messages except status displays. A numbered message appears as follows:

```
12 IEE041I THE SYSTEM LOG IS NOW ACTIVE
```

To request message numbering, operators use the CONTROL D,N and CONTROL E,N commands to display and erase message numbers:

1. Enter CONTROL D,N to display consecutive numbers in character positions one and two of each message area line
2. Enter CONTROL E,N to remove the message numbers from the screen when CONTROL D,N HOLD is in effect

When the operator issues CONTROL D,N and then deletes a message or cancels an action, the numbers are removed from the screen. To ensure that the remaining messages are renumbered, the operator can add the HOLD operand to the command.

Notes:

1. Automatic message deletion (automatic mode, roll mode, or roll-deletable mode) stops message numbering requested by the CONTROL D,N,HOLD command.
2. Because a display console screen can be “burned” by the number images, it is recommended that you do not have the messages numbered all of the time. When you are in conversational mode and delete messages by the CONTROL command, all messages are temporarily numbered so that you can verify that you have entered the correct delete command.
3. For very large screen sizes, only the first 99 rows can be numbered. All rows after 99 contain AA in positions 1 and 2. Message lines with AA in the number field cannot be deleted.

Using SEG to Delete Groups of Messages from the Screen

Operators can delete groups or “segments” of non-action messages on the screen using the CONTROL E,SEG command. SEG specifies the number of message lines to be deleted; you can define this value as a keyword on the CONSOLE statement.

Use the following keyword on the CONSOLE statement to specify the number of lines the system deletes when an operator enters CONTROL E,SEG:

SEG Specifies the number of lines to be deleted when the operator enters a CONTROL E,SEG command.

The IBM default depends on the type of console. *z/OS MVS Initialization and Tuning Reference* provides default information for different console devices.

Status Displays and MCS and SMCS Consoles

A status display is a formatted, multi-line display of information about some part of the system. It is written to MCS consoles in full-capability or status display mode and to SMCS consoles, which must be in full capability mode, in response to certain subsystem commands or the following MVS commands:

- DISPLAY
- CONFIG
- DEVSERV
- TRACK

On consoles in status display or full capability mode, status displays are usually presented in display areas (called out-of-line display areas) set aside for their use. If you do not define one or more display areas, status displays appear in the general message traffic. The information in the status display could, therefore, roll off the screen before your operators can find it. “Setting Up Out-of-Line Display Areas on a Console” describes how you set up status displays for your consoles.

When you have defined your status display consoles and console areas, your operators can obtain information, such as the status of system devices and the identification of the jobs active in the system, that can help you decide how best to use system resources.

A status display is either static or dynamic. A display is static if it remains the same until it is removed from the screen; a dynamic display is created by the TRACK command and is updated by the system each time a preset time interval elapses. (You can set the time interval for updating dynamic status displays for a console using the UTME attribute. “Defining the Time Interval for Updating a Dynamic Status Display” on page 82 describes how you can control the time interval.) Displays of the TRACK command continue for the duration of the IPL or until the operator issues another request.

You can route the output of the DISPLAY, CONFIG, DEVSERV, or TRACK commands to any status display console or console area in your system or sysplex:

- DISPLAY provides information about job activity, TSO/E users, console configuration, device status, and more.
- Output from CONFIG contains information about changes in the configuration of processors, storage, channel paths, and other system resources.
- DEVSERV displays the status of DASD and tape devices.
- TRACK initiates a dynamic status display and provides job activity and TSO/E user information. On the CONSOLE statement, you can tailor the kind of status information that TRACK displays on a console.

For complete information on these commands, see *z/OS MVS System Commands*.

“Where to Route Status Displays” on page 81 describes how you route information from these commands to status display consoles and areas.

Setting Up Out-of-Line Display Areas on a Console

You can control the number of out-of-line display areas on a status display or full-capability console screen and the size of each area. You can specify up to 10 different out-of-line display areas, the location of the areas, and the number of screen lines in each area. In a sysplex, you might direct status information for several systems to different console areas on one screen of a full-capability console.

You define out-of-line display areas for an MCS console or SMCS console. The SMCS console must be in full-capability mode. You define the areas from the bottom of the message area to the top of the area. Each area consists of four or more screen lines designated to receive the status displays.

For each out-of-line display area, the system assigns the alphabetic display area identifiers. The bottom-most area is assigned identifier A and additional areas are assigned identifiers in alphabetic order, working toward the top of the screen. The identifier Z always refers to the portion of the message area that is not assigned.

Figure 11 shows the screen format for a display console in full-capability mode when two typical out-of-line display areas are defined for the screen. The first (bottom-most) area has four lines, and the second has six lines. You can route status displays for the console using the MSGRT keyword on the CONSOLE statement. After IPL, operators can route status displays using the location operand of the DISPLAY, TRACK, CONFIG, and MSGRT commands to area A or B, or to the general message area.)

Use the following keyword on the CONSOLE statement, to define the out-of-line

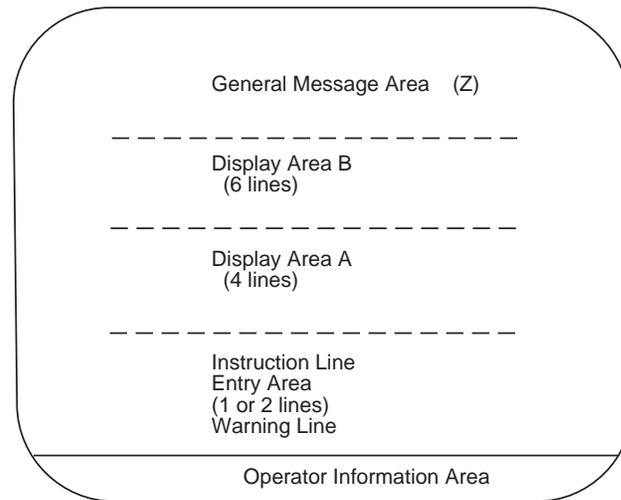


Figure 11. Sample Screen Showing Two Out-of-Line Display Areas on a Full-Capability Console

display areas for a console:

AREA Defines the console out-of-line display area. The total number of lines you specify for all out-of-line display areas must not exceed the size of the screen.

If you do not code the AREA parameter, the system defines two display areas for status display consoles and one display area for full-capability consoles. The number of lines in each area depends on the type of device.

Operators can use CONTROL A to change out-of-line display areas. For the maximum display area sizes for all devices that MVS supports as consoles, see *z/OS MVS Initialization and Tuning Reference*.

Where to Route Status Displays

You can decide where you want to route the status display information from the DISPLAY, MONITOR, CONFIG, or TRACK commands entered on the console. To route the status display information from these commands issued on the console to other consoles (or console areas) in the system or sysplex, use the following keyword on the CONSOLE statement:

MSGRT Routes output from the DISPLAY, MONITOR, and CONFIG commands issued on this console to other consoles.

One MSGRT parameter can define routing instructions for all or any combination of the commands for which MSGRT defines routing instructions. You can define different routing instructions for each console in the system or sysplex.

When you specify a console name and an out-of-line display area on the location operand (L=) of any MVS command, you specify the name of the console, followed by a dash and the alphabetic character for the out-of-line display area you want. For example, to direct the displays of the DISPLAY and CONFIG commands to two different out-of-line display areas of a console named MSG, the operator can enter:

```
MSGRT (D=A,L=MSG-A),(CF,L=MSG-B)
```

An operator can change these routing instructions through commands and route many of the displays that normally come to the console to another console's general message area, to a display area on another console, or to a specific display area on the operator's console. An operator can use the following to route status displays of the DISPLAY, TRACK, and CONFIG commands:

- L= on DISPLAY, TRACK, and CONFIG
- MSGRT command

Operators can use the location operand (L=) of the DISPLAY, TRACK, and CONFIG commands to route status displays to specific display areas on the requesting console or to route displays to other output-only consoles. However, operators must have the proper authority to route information to another console using the L= operand.

If the operator does not specify L=, the system presents the display according to the MSGRT instructions in effect for console. If there are no MSGRT instructions in effect, the system presents the display as follows:

1. In the issuing console's lowest unoccupied out-of-line display area.
2. If all areas are full, in the out-of-line display area containing the oldest display of the issuing console.
3. If no out-of-line display area can be found (because the screen has no display areas or because all areas are being used by dynamic displays), in the general message area Z of the issuing console.

See *z/OS MVS System Commands* for how to use the DISPLAY, TRACK, CONFIG, and MSGRT commands.

Defining the Time Interval for Updating a Dynamic Status Display

Each console where a dynamic status display is in progress has a time interval value that determines how often the system updates the TRACK command displays. Use the following keyword on the CONSOLE statement to define the time interval:

UTME Specifies, in seconds, the interval of time between updates of dynamic status displays for a console. The range is from 10 to 999.

If you do not define a time interval value, the system updates the display every 30 seconds.

Controlling the Format of Messages and Status Information on Console Screens

On a display console, a message can appear by itself or with information about the message, such as job and system identification and the time the message was issued. In a status display, information about when jobs start or stop, when a data set is freed, or information about TSO/E user sessions can appear. Also, mount messages in status displays can contain specific information about mounting volumes.

You can control the information for messages or status displays that operators view on the console screen. Controlling message formats can help free up screen space or make it easier for operators to read messages. Controlling status information can help operators monitor workload or handle job allocation that requires mounting requests.

Use the following keyword and its options on the `CONSOLE` statement to control information about messages for display:

MFORM Controls the message format on a console screen.

Options you can specify for `MFORM` are as follows:

- M** Specifies that the system display only the text of the message without time stamp, job id, or job name
- J** Specifies that the system display the job name or id along with the message text
- S** Specifies that the system display the system name that originated the message
- T** Specifies that the system display a time stamp with the message
- X** Specifies that the system suppress the job name and system name for JES3 messages issued from the global processor

How messages are displayed on the screen can affect your operations. Consider eliminating information from displayed messages when:

- Messages wrap to a second line making it difficult for operators to read the screen.
 - One way to prevent line wrapping, or to allow system name, job name, and time stamp to be displayed with all the message text on one line, is to use an emulator and set a large screen width to allow all the data to appear on one line.
- The system id is not important (for example, in a single system)

To request that the system add a time stamp, the name of the system that issued the message, and the job name or id of its issuer, code the following on the `CONSOLE` statement:

```
MFORM(J,S,T)
```

Operators can also use the `CONTROL S` command to make these same changes. The format of a message that includes information in the previous example is:

```
Time stamp System name Jobname/id Message text
```

MCS or SMCS console display

Defining the `X` option for an `MCS` or `SMCS` console allows you to suppress the system name and jobname for JES3 messages that are issued from the global processor when those messages appear on the `MCS` or `SMCS` console screen.

For example, to suppress both jobname and system name for JES3 messages issued on the global processor, code the following `MFORM` values on the `CONSOLE` statement for the `MCS` console:

```
CONSOLE DEVNUM(devnum) NAME(conname) MFORM(T,J,S,X)
```

For an `SMCS` console, devnum must be `SMCS`.

Displaying system names in a sysplex

In a sysplex, the number of characters displayed on the console screen for system name depends on the longest name of the system that joins the sysplex. If SYSB is the longest name, all system names will be four characters. If SYB is the longest name, all system names will be three characters.

For example, if three systems in a sysplex are named SYS1, SY2, and S3, the displayed messages from any system will have a four character system name:

```
SYS1 message  
SY2  message  
S3   message
```

If a system with longer name joins the sysplex, the length of the system name in the messages is adjusted to accommodate the new name. For consistency, you might want to use system names of the same character length.

DISPLAY R, CONTROL S, and MFORM

Operators can issue the DISPLAY R command with MFORM options to retrieve information about messages awaiting action. In a sysplex, if the operator issues DISPLAY R without MFORM, the format of the messages depends on how MFORM has been specified for CONSOLxx or on the CONTROL S command:

- If CONTROL S has NOT been issued, the format of the messages depends on MFORM values specified for CONSOLxx on the system where the command is issued.
- If CONTROL S with MFORM options has been issued before the DISPLAY R command has been issued, the format of the messages depends on MFORM values specified for CONTROL S.

For JES3 multisystem environments, when DISPLAY R is issued without MFORM, the system uses the S option as a default.

Displaying Jobname, Data Set Status, and TSO/E Information

You can request that the system notify operators in status displays when the following events occur:

- Whenever a job starts and ends
- Whenever a data set is freed
- Whenever a TSO/E user starts and ends a session

Use the following keyword on the CONSOLE statement to define job, data set, or TSO/E information:

MONITOR Specifies that you want to display certain status information

Options you can specify for MONITOR are as follows:

JOBNAMES Specifies that the name of the job is displayed in status display areas whenever the job starts and stops

STATUS Specifies that data set names and volume serial numbers are displayed in status display areas whenever data sets are freed

SESS Specifies that the TSO/E user identifier is displayed whenever the TSO/E session begins and ends

With JOBNAMES or SESS, you can add a time stamp (-T).

Adding Information to Mount Messages

You can request that the system add certain information to all mount messages on consoles. The MONITOR keyword on the INIT statement in the CONSOLxx member controls whether the system adds information to mount messages for all console status displays.

Use the following keyword and its options on the INIT statement to specify information about mount messages for status displays:

MONITOR Specifies that you want to display status information for mount messages

Options you can specify for MONITOR are as follows:

SPACE Specifies that the available space on the direct access volume appears in the message

DSNAME Specifies that the name of the first non-temporary data set allocated on the volume appears in the mount message that refers to it

Defining PFKs and Other Command Controls for Consoles

You can control the program function keys (PFKs) for MCS or SMCS consoles and also how operators can enter multiple commands using a command delimiter.

Setting up PFKs for Consoles

CONSOLxx and PFKTABxx let you define the PFKs for all your MCS or SMCS consoles on a system. For each console, you activate a PFK table — a table that your installation has defined — by specifying the PFK table name on the CONSOLE statement. The PFK table resides, optionally with PFK tables for other consoles, in a PFKTABxx Parmlib member.

Using entries in the PFK table, you can:

- Assign one or more commands to a PFK for the console
You can associate the text of one or more commands with a PFK. Later, when an operator presses this PFK on the console, the commands are entered into the system.
- Assign one or more other PFKs to a PFK for the console
You can associate the commands assigned to other PFKs with a PFK.

To create PFK table entries, use the following keywords in PFKTABxx of SYS1.PARMLIB:

TABLE Defines the table to contain PFKs for the console. You associate this table with the console by specifying the table name on the PFKTAB keyword of the CONSOLE statement.

PFK Defines the program function key.

CMD Defines the command or commands to be assigned to the PFK.

KEY Associates the PFK you define with another key or list of keys.

CON Defines whether the PFK you define operates in conversational or nonconversational mode.

Conversational or nonconversational mode applies to commands defined to a PFK. In nonconversational mode, the commands associated with a key are entered

immediately when the operator presses the key on the console. In conversational mode, pressing a PFK causes the command to appear in the entry area, but no enter action takes place. Operators can change, enter, or cancel the command according to their requirements.

In conversational mode, the cursor normally appears under the third non-blank character when the command is in the entry area. If you want the cursor to appear in a different location, when you define the command, type an underscore immediately to the right of the character under which the cursor is to appear. The system deletes the space occupied by the underscore in the actual command. For example, if you add the following entry to a PFK table:

```
PFK(5) CMD('D U,L=_XXX') CON(Y)
```

pressing PFK 5 causes the following to appear in the entry area:

```
D U,L=XXX
```

If you want an underscore to appear in the command, use two underscores. They are treated as one underscore, and are not used for cursor placement. For example, if the PFKTAB table contains:

```
PFK(17) CMD("E_XXXXXXXX,SRVCLASS=BATT__HI"),CON(Y)
```

when you press PFkey 17, the entry area will contain

```
E XXXXXXXX,SRVCLASS=BAT_HI
```

Selector pens also use the definitions in PFK tables.

When you have created your PFK tables in PFKTABxx, you can associate them with the consoles in your configuration. Specify the following keyword on the CONSOLE statement to associate a PFK table with the console:

PFKTAB Defines the name of the PFK table defined in PFKTABxx that contains PFKs for this console. The name must be the same as the name for TABLE in PFKTABxx.

When you have defined the PFK tables for all your consoles, you can activate the PFKTABxx member that contains the table definitions at IPL. Use the following keyword on the INIT statement of CONSOLxx to activate PFKTABxx:

PFK Defines the name of the PFKTABxx member that contains the PFK definition tables for your consoles. For PFK you specify a value that corresponds to xx in PFKTABxx. If you specify NONE for PFK, the system uses IBM defaults for console PFKs. See Table 12.

If you do not specify PFKs for your consoles or if the system does not find the PFK parameter, it issues the message:

```
IEAI180 USING IBM DEFAULT DEFINITIONS. NO PFK TABLES REQUESTED.
```

IBM supplies defaults for PFKs 1 through 9 as follows:

Table 12. IBM Defaults for PFKs

PFK	Command	Comment
1	CONTROL E,1	Erase one line from screen
2	CONTROL E	Erase one segment from screen
3	CONTROL E,D	Erase status display from screen

Table 12. IBM Defaults for PFKs (continued)

PFK	Command	Comment
4	CONTROL D,F	Frame display forward in area
5	CONTROL S,DEL=N	Hold in-line output
6	CONTROL S,DEL=RD	Resume in-line output
7	DISPLAY A,L	List active jobs and TSO/E users
8	DISPLAY R,L	List all outstanding operator action requests
9	CONTROL D,U	Update dynamic display
10 and up		No definition provided

In a sysplex, PFK settings have system scope; they apply only to the consoles on the system where they are defined.

An Example of Defining a PFK Table

The following example shows you how to define and activate a PFK table for a console configuration defined in CONSOL01. In this example, the installation has been using IBM defaults for PFKs 1 through 9. PFK table MVSCMDS to be created will reside in the PFKTAB01 Parmlib member.

Procedure	Coding of Parmlib Member
Create the PFK table named MVSCMDS	Assign commands to PFK(nn) definitions in PFKTAB01, where nn is the PFK number.
Associate MVSCMDS with a console	Specify PFKTAB(MVSCMDS) on the CONSOLE statement in the CONSOL01 Parmlib member.
Activate the PFKTAB01 Parmlib member that contains the PFK table named MVSCMDS	Specify PFK(01) on the INIT statement in the CONSOL01 Parmlib member.

When you IPL the system, the system uses MVSCMDS to define the PFKs on your console.

Use the same PFKTAB01 member to hold the PFK tables for your JES2 and tape library operators. Figure 12 shows the PFKTAB01 Parmlib member. It contains three tables: MVSCMDS, JES2CMDS, and TLCMDS.

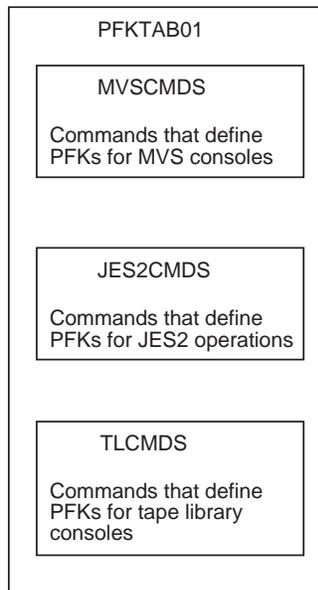


Figure 12. PFKTAB01 Parmlib Member.

For information about using the CONTROL command to modify PFKs for a console, see *z/OS MVS System Commands*.

Defining the Command Delimiter for Full-Capability Consoles

You can define full-capability consoles so that operators can enter multiple commands from the command line. You define a character that the operator can use to separate MVS commands. Operators can divide a series of commands on the command line using the character as the command delimiter. (You can also specify multiple commands using the command delimiter when defining PFKs for consoles.)

To define a command delimiter for MCS consoles, use the following keyword on the INIT statement of CONSOLxx:

CMDDDELIM

If you do not define a command delimiter, your operators cannot enter multiple commands from a full-capability console.

You can also use a command delimiter to separate subsystem commands; however, some delimiters might conflict with characters used in certain subsystem commands like JES commands.

For command delimiter characters that you can use and the restrictions that apply, see *z/OS MVS Initialization and Tuning Reference*.

Hardcopy Processing

Hardcopy processing allows your installation to have a permanent record of system activity and helps you audit the use of operator commands. You can record system messages and, optionally, commands, by using either the system log (SYSLOG), the operations log (OPERLOG), or an MCS printer. The group of messages and commands that are recorded is called the hardcopy message set. The system log, operations log, or MCS printer that receives messages is called the hardcopy

medium. You can specify a group of console devices that can serve as backup devices for the hardcopy medium. You can also allow an extended MCS console to receive hardcopy messages from one or more systems in a sysplex.

Hardcopy processing is required in a sysplex.

Note: The term “hardcopy log” can refer to:

- The system log (SYSLOG)
- The operations log (OPERLOG)
- The device used to print hardcopy messages
- The data set containing hardcopy messages
- The actual printed copy of the hardcopy messages

The Hardcopy Message Set

The hardcopy message set represents messages that can be either recorded in hardcopy on the system log or the operations log, or sent to an MCS printer. The hardcopy message set is usually sent to the current active log, either the system log or the operations log, or both, but may be sent to a printer console, if the installation chooses. The hardcopy message set is defined at system initialization and may subsequently be changed by the VARY command.

Characteristics of the Hardcopy Message Set

The hardcopy message set includes messages with one or more of the following characteristics. Messages in the hardcopy message set:

- Have the “hardcopy only” message delivery attribute
- Are WTOR messages
- Have descriptor codes of 1, 2, 3, 11, or 12
- Have no routing codes
- Have an installation-specified routing code
- Are command responses of the installation’s specified command level
- Have a message type specified.

Messages for which “no hardcopy” is requested are not included in the hardcopy message set, regardless of their other characteristics.

Defining the Hardcopy Message Set

Messages included in the hardcopy message set are either commands and command responses or unsolicited system messages. Installations can control the selection criteria for commands and command responses. Installations can control some of the criteria for unsolicited system messages; some of the criteria are fixed.

You define criteria for messages in the hardcopy message set at system initialization with the HARDCOPY statement in the CONSOLxx member of Parmlib:

- For commands and command responses, the CMDLEVEL option of the HARDCOPY statement controls the types of commands included in the hardcopy message set.
- For unsolicited system messages, the ROUTCODE option of the HARDCOPY statement controls the routing codes the system uses to select messages for the hardcopy message set. If an option is not specified, the default value is used for the hardcopy message set definition.

Once MVS has been initialized, you can modify the criteria of the hardcopy message set using the VARY HARDCOPY command.

Unsolicited system messages for which the criteria are fixed are those messages that match one or more of the following characteristics:

- Have descriptor codes of 1, 2, 3, 11, or 12
- Are WTORs
- Have no routing codes
- Have a message type specified
- Are hardcopy only messages.

In a JES2 complex, you define the hardcopy message set in the CONSOLxx member of Parmlib. If you are using the JES3 hardcopy log (JES3 DLOG), it is maintained on the JES3 global processor for all messages issued in the complex. For information, see *z/OS JES3 Initialization and Tuning Guide*.

Printing the Hardcopy Message Set

The hardcopy message set can be printed at once by the hardcopy output device or directed to either the system log, the operations log, or both; the system log is printed periodically. To obtain a permanent log about operating conditions and maintenance for all systems in a sysplex, you should use a coupling facility OPERLOG log stream. To obtain a permanent log about operating conditions and maintenance for a system operating independently, you can use either a DASD-only OPERLOG log stream or SYSLOG.

The Hardcopy Medium

You can specify whether the hardcopy medium is an MCS printer, the system log (SYSLOG), or the operations log (OPERLOG) at system initialization using the DEVNUM, UD, and HCPYGRP keywords on the HARDCOPY statement in the CONSOLxx member of Parmlib. Once the system has been initialized, operators can use the VARY HARDCPY command to redefine the hardcopy medium. Operators can, however, enter the VARY HARDCPY command to change a hardcopy device only from MCS, SMCS or extended MCS consoles with master authority.

Reference

For complete information about the HARDCOPY statement of CONSOLxx, see *z/OS MVS Initialization and Tuning Reference*.

An extended MCS console can also receive the hardcopy message set. You request that an extended MCS console receive the hardcopy message set by using the MCSOPER macro with the HARDCOPY attribute on the OPERPARM parameter. You can also use this macro to collect all the hardcopy messages from one or more systems in a sysplex. See *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU* for information about the MCSOPER macro.

Hardcopy Processing

In a sysplex, the values for the HARDCOPY statement have system scope; they apply only to the system where HARDCOPY is defined. If you use the MCSOPER macro to have an extended MCS console receive all the messages in the hardcopy message set from one or more systems in a sysplex, it will receive messages from the hardcopy message set *as it is defined on each system*.

Format of Hardcopy Records

Your hardcopy records can continue to have a 2-digit year format or can have a 4-digit year format. To specify use of a 4-digit year, use the HCFORMAT keyword

on the HARDCOPY statement in CONSOLxx. For any programs that read and analyze hardcopy records, the IHAHCLOG mapping macro is provided. The HCLFRMT or HCRFRMT fields will indicate which format is being used in the log records. The HCL mapping is used for a 2-digit year format and the HCR mapping is used for a 4-digit year format.

Using OPERLOG

The operations log (OPERLOG) is a log stream that uses the system logger to record and merge communications about programs and system functions from each system in a sysplex. Only the systems in a sysplex that have specified and activated the operations log will have their records sent to OPERLOG. For example, if a sysplex has three systems, SYS A, SYS B, and SYS C, but only SYS A and SYS B activate the operations log, then only SYS A and SYS B will have their information recorded in the operations log.

IBM recommends that JES3 customers with a multisystem sysplex use an OPERLOG coupling facility log stream and turn off JES3 DLOG and SYSLOG.

You can also use OPERLOG as a DASD-only log stream. This method is only suitable for a single system sysplex, because a DASD-only log stream is single-sysplex in scope and you can only have one OPERLOG log stream per sysplex. This means that if you make OPERLOG a DASD-only log stream, only one system can access it. See the system logger chapter of *z/OS MVS Setting Up a Sysplex* for information on DASD-only log streams.

The messages are logged using message data blocks (MDB), which provide more data than is recorded in the SYSLOG. You can use member IEAMDBLG, in SYS1.SAMPLIB, to convert OPERLOG records into SYSLOG format.

The operations log is operationally independent of the system log. An installation can choose to run with either or both of the logs. If you choose to use the operations log as a replacement for SYSLOG, you can prevent the future use of SYSLOG; once the operations log is started with the SYSLOG not active, enter the WRITELOG CLOSE command.

Although the operations log is sysplex in scope, the commands that control its status and the initialization parameter that activates it have a system scope, meaning that a failure in operations log processing on one system will not have any direct effect on the other systems in the sysplex. You can set up the operations log to receive records from an entire sysplex or from only a subset of the systems, depending on the needs of the installation.

Initializing the Operations Log

Before you can begin using the operations log, you must define a log stream using the system logger services. Specify the name of the log stream as SYSPLEX.OPERLOG in either the data administrative utility or in the IXGINVNT macro. See *z/OS MVS Setting Up a Sysplex* for more information about preparing to use a log stream and on sizing the coupling facility structure for OPERLOG.

You must also verify that the operations log will contain the messages you need. Messages in the operations log will include the hardcopy message set, which you control. See “The Hardcopy Message Set” on page 89 for more information.

To activate the operations log manually, enter a VARY command.

Processing Operations Log Records

You might have your own programs for analyzing SYSLOG records in batch jobs. These programs will not work with the operations log because the records are in MDB format. You can convert the SYSLOG analysis programs. The IEAMDBLG sample program, available in SYS1.SAMPLIB, is an example of a program that reads selected operations log records and converts them from MDB to SYSLOG format for analysis.

Using SYSLOG

The system log (SYSLOG) is a data set residing in the primary job entry subsystem's spool space. It can be used by application and system programmers to record communications about problem programs and system functions. The operator can use the LOG command to add an entry to the system log.

Note: You can change the SYSLOG data set characteristics dynamically through the dynamic allocation installation exit. See *z/OS MVS Installation Exits*.

SYSLOG is queued for printing when the number of messages recorded reaches a threshold specified at system initialization. The operator can force the system log data set to be queued for printing before the threshold is reached by issuing the WRITELOG command.

Specifying an Alternate Console Group for Hardcopy Recovery

You can plan the recovery of your hardcopy output device by specifying the name of a group of console devices that MVS can switch to if the device fails. MVS searches for an eligible device based on the order of the console members specified in the alternate console group, and selects the first available device as the hardcopy medium.

You define alternate console groups that contain eligible hardcopy devices in CNGRPxx. Eligible console devices to use for hardcopy switching must be printer devices; you can specify *SYSLOG*, in which case MVS can select the system log as the hardcopy medium.

You specify the alternate console group name on the HCPYGRP keyword of the HARDCOPY statement. The CNGRPxx member that contains the console group definition must be active in order for MVS to select an alternate device from the group during a hardcopy switch.

If you do not specify an alternate console group on HCPYGRP, MVS selects any active non-display console by checking all consoles in the order in which they were specified in the CONSOLxx member of SYS1.PARMLIB. MVS assigns the hardcopy function to the first active non-display console it finds. If the entire search is unsuccessful, MVS suspends hardcopy processing and notifies the master console.

Log Switching and JES2 Restart

If OPERLOG is the hardcopy medium and the OPERLOG fails on one system, the system attempts to switch hardcopy to the SYSLOG. The system on which the operations log fails will write messages about the failure to SYSLOG. Any systems in the sysplex that are not affected by the failure will continue to write to the operations log.

If the SYSLOG is also inactive, or if the use of SYSLOG has been prevented by the WRITELOG CLOSE command, the system attempts to switch hardcopy to an appropriate printer console. If a printer console is not available, the system searches the group name indicated in the HCPYGRP parameter of the CONSOLxx

parmlib member for a suitable backup. If the HCPYGRP parameter was not specified or the search for a suitable backup fails, the system attempts to reactivate the SYSLOG. If all of these attempts fail to provide a backup to the failed operations log, the system issues a message indicating that there is no hardcopy medium available.

When OPERLOG is restarted, the system automatically begins recording to the operations log again. If hardcopy recording had been switched to a printer console or to SYSLOG, you must manually vary these offline again.

If SYSLOG is the hardcopy medium and the SYSLOG device fails, the system attempts to switch hardcopy to an appropriate printer console. In a JES2 system, the hardcopy message set can be directed to a printer console if the installation chooses. If a system running JES2 fails, the system automatically switches from the printer console to SYSLOG when JES2 is restarted after the failure. If a suitable console is not active at the time of failure, the system suspends hardcopy processing and notifies the master console.

Table 13 summarizes log switching during an automatic restart:

Table 13. Automatic restart of log with JES2

Log device	JES2 terminates	JES2 restarts
OPERLOG	Does not affect OPERLOG.	Does not affect OPERLOG.
SYSLOG	System switches to an available console	System switches back to SYSLOG
hardcopy console	System ignores switch	System restores hardcopy console

Temporarily Disabling the Hardcopy Medium

If you are using SYSLOG as the hardcopy medium and it is not operating properly, MVS saves the messages in log buffers until their number reaches the value of LOGLIM. When this limit is reached, messages remain on the WQE queue until the WQEs reach the MLIM limit. At this point, no new messages can be displayed; the only way to re-start the log is to de-activate it. This requires a re-IPL.

In Version 1 Release 2 of z/OS a parameter is available to turn off hardcopy. The parameter, UNCOND, is coded on the VARY command. For example:

```
VARY devnum,HARDCPY,OFF,UNCOND
```

After this command is executed, you can remove SYSLOG as the hardcopy medium, and WRITELOG CLOSE is accepted.

Note: This should be a temporary measure because, if SYSLOG has been removed the system can lose messages from hardcopy.

If SYSLOG is removed and it was the only hardcopy medium, it is considered temporarily off. Log buffers can be saved only until they reach LOGLIM. After this point WQEs are not saved and some messages can be lost, but the outage will be prevented.

Chapter 3. Managing Messages and Commands

Whether you are defining a console configuration for a system or for several systems in a sysplex, you must take into account your operators, the amount of message traffic they must handle, and command processing.

Messages and commands form the basis of operator communication in an MVS system or sysplex. Message routing, sending the appropriate messages to the right consoles, helps your operators manage work efficiently. Message routing using CONSOLxx can simplify the work operators need to do.

If you want to increase system automation to simplify operator tasks, you should examine message flow to determine which messages you can select for your automation tasks and which you can suppress. Suppressing messages is important in any MVS environment because your operators deal with fewer messages on their console screens. Message suppression also serves as a basis for your NetView automation planning.

In a sysplex, operators can also route commands from a console on one system to be processed on one or more other systems in the sysplex. You might want to encourage the use of *system symbols* in routed commands so you can identify the systems that process those commands. For more information, see the section on sharing commands in *z/OS MVS System Commands Summary*.

MVS provides message processing facilities to help you and your operators cope with message flow on consoles. For example, MPF or the installation message processing exit IEAVMXIT can help you select messages to suppress or to perform further processing like message highlighting for more readable console displays. AMRF lets your operators retrieve important action messages no longer visible on the console screen. The MVS command installation exit lets you process and tailor system commands. You use one or more MPFLSTxx members in SYS1.PARMLIB to control much of this message and command processing for an MVS system.

This chapter describes how to manage messages and commands in an MVS system or sysplex. It describes message and command routing and the message processing that MVS provides to suppress messages, retain messages for console viewing by operators, and select messages for automation or for further processing by installation exits, and a brief description of automation in a sysplex. It also provides information on controlling WTO and WTOR message buffers, specifying installation exits to process commands, and using the MVS message service to handle the translation of messages into other languages. For additional information, see the section on issuing a command response message in *z/OS MVS Programming: Authorized Assembler Services Guide*.

General Characteristics of Messages and Commands

Operators can issue commands to correct problems or to query the system to determine if it is operating properly. They often do this in response to system messages. Some messages require a reply from the operator. These messages are called WTORs (write-to-operator-with-reply). The operator responds to these messages by entering the REPLY command. Automation programs like NetView use messages and command lists to simplify operator tasks and actions.

Messages and commands can be routed throughout a system or sysplex; the routing of messages and commands is an important part of operations planning.

You want to ensure that operators are receiving the necessary messages at their consoles to perform their tasks. You want to be able to select the proper messages for suppression, automation, or other kinds of message processing.

Commands in a sysplex can run on other systems and affect system processing. In a sysplex, operators can also route commands from one system to another for processing. You might want to limit command processing to a specific system in the sysplex, or handle commands through command installation exits.

MVS messages have routing codes and message levels that, in large part, determine how messages are routed in a system or sysplex. Routing codes are decimal numbers from 1 to 128 that can be assigned to a console. For example, routing codes can:

- Routing code 1 (master console action) indicates messages that require a specific operator action.

- Other routing codes indicate messages that convey information about a specific system function or operator area. For example, messages with routing code 6 convey about the disk library.

- Other routing codes indicate an error condition. For example, messages with routing code 10 convey information about a system error, an uncorrectable I/O error, or information about system maintenance.

- Other routing codes have specific meanings. For complete information, see *z/OS MVS Routing and Descriptor Codes*.

For MCS, SMCS and extended MCS consoles, you can specify which routing codes the console is to receive.

Message levels allow MVS to select messages according to the severity of the condition or situation described in the message. Message levels can range from WTOR messages that require an operator response, to informational messages that indicate system status. You assign these levels to specific MCS, SMCS or extended MCS consoles so the system can direct messages at those levels to the console. For example, you can assign message level (R) for WTOR messages to a master console or a full-capability console that handles critical system messages. Assigning message levels to the appropriate consoles in your configuration is a good way to control message traffic for MCS, SMCS and extended MCS consoles.

You can ensure that a console receives action messages, WTOR messages, and important informational messages that the system could not deliver to the expected console by assigning the UD attribute to the console. Any MCS, SMCS or extended MCS console with the UD attribute can receive and display these messages.

The system sometimes issues synchronous messages that bypass normal console queuing. These messages might require immediate operator action or can indicate system problems. You can define a group of consoles from which MVS can select a candidate to display these synchronous messages. For more information, see Chapter 2, "Defining Your Console Configuration" on page 13.

Message and Command Routing

Understanding message and command flow in an MVS system or sysplex can help you handle message and command processing.

Message Flow in a System

When MVS issues either a write-to-operator (WTO) message or write-to-operator-with-reply (WTOR) message, message processing exits receive control to allow the installation to process the message. Each time a WTO or WTOR message is issued, it flows through message exit IEAVMXIT, if it exists. You can specify other message processing exits to override IEAVMXIT in the MPFLSTxx Parmlib member. MPFLSTxx also allows you to control other kinds of message processing like message highlighting, message suppression, and message automation.

After a message passes through the message processing exits, subsystems like JES2, JES3, or NetView can receive the message for processing. For example, NetView can process any message that MPFLSTxx defines as eligible for automation. Subsystem allocatable consoles can receive the message for display.

After subsystem processing occurs, the message passes to the hardcopy log. Depending on CONSOLxx values that control hardcopy logging, the hardcopy log can record the message.

After the system records the message in the hardcopy log, the message passes to MCS, SMCS and extended MCS consoles where it can be displayed.

The following summarizes this generalized message flow for an MVS system:

1. The system issues the message.
2. Processing specified in MPFLSTxx for the message occurs. IEAVMXIT or the installation exits specified through MPFLSTxx receive control.
3. The subsystems receive the message.
4. Depending on CONSOLxx values, hardcopy log processing records the message.
5. The MCS, SMCS console or extended MCS console can display the message.

Command Flow in a System

When the operator issues a command in a single MVS system, the system records the command in the hardcopy log if the command is eligible for recording, as specified in CONSOLxx. The command then flows through one or more command installation exits specified in MPFLSTxx. If exit processing changes the original command, the system issues message IEE295I and then, if the modified command is eligible for recording, records the command in the hardcopy log. Finally, the command processor for the command gets control to process the command on the system.

The following summarizes this generalized command flow for an MVS system:

1. An operator or program issues the command.
2. Depending on CONSOLxx values, hardcopy log processing records the command.
3. Processing specified in MPFLSTxx for the command occurs. The installation exits specified through MPFLSTxx receive control.
4. If the exit processing modified the command, the system issues message IEE295I and depending on CONSOLxx values, hardcopy log processing records the command.
5. If any installation exit processes the command, no further command processing occurs.

6. The subsystems receive the command.
7. If any subsystem processes the command, no further command processing occurs.
8. The MVS command processor receives control to process the command.

Command Flooding

Most MVS commands are executed by attaching a task in either the *MASTER* or CONSOLE address space. If too many such tasks are attached at one time (usually because a program has issued too many MGCRE macros), command flooding occurs.

Attached commands that run in the *MASTER* or CONSOLE address space are divided into four “command classes”. In each class, only 50 commands can execute at one time. Any additional commands in that class must wait for execution. This prevents an out-of-storage condition. To manage the number of commands that are awaiting execution, the system operator can issue the CMDS command to display the status of commands, and remove selected commands that are awaiting execution. The IEECMDS macro provides similar function.

- For information about the CMDS command: *z/OS MVS System Commands*
- For information about the IEECMDS macro: *z/OS MVS Programming: Authorized Assembler Services Reference ENF-IXG*

Message and Command Flow in a Sysplex

In a sysplex, MVS messages and commands can flow from system to system. Because you can direct the message or command to one or more systems, you need to understand how exits, subsystems, and hardcopy log processing occurs in a sysplex.

Consider the following for message and command flow in a sysplex:

- Console operations. MCS, SMCS and extended MCS consoles can receive messages from different systems or enter commands to affect the processing of other systems.
- Installation exits. Installation exits can perform much of your message and command processing. For processing like message suppression, making messages eligible for automation, or for processing that alters commands, message and command flow in a sysplex become important.
- Subsystem processing. Subsystems can process MVS messages and commands. To help control or coordinate subsystem functions in a sysplex, subsystems need to consider from which systems messages and commands originate.
- Automation for systems in a sysplex. Automation programs like NetView use messages and commands to control automation in a system. How different systems control automation in a sysplex depends on how messages and commands can flow from system to system.
- Logging of messages and commands in a sysplex. You can use the operations log (OPERLOG) to record messages and commands from all the systems in a sysplex. The operations log centralizes log data in a sysplex.

Messages in a Sysplex

In a sysplex, you can direct a message to one or more systems for processing. You can control message routing to consoles in the sysplex through the MSCOPE keyword on the CONSOLE statement for MCS or SMCS consoles. For extended

MCS consoles, you can use RACF or MCSOPER to specify MSCOPE values. (See Table 1 on page 7.) Operators can use the VARY command to control MSCOPE. Specifying MSCOPE allows the console to receive messages from one or more systems in the sysplex.

Regardless of the console MSCOPE value, the sysplex can direct messages in the form of command responses to the system where the console that issued the command is attached. For example, a console on SYA that issues a DISPLAY command for other consoles on SYB can expect to receive the message display in response to the command. For a subsystem-allocatable console, the sysplex can deliver a message to the subsystem where the console is allocated. Thus, a subsystem console on SYA can receive messages intended for the console even if the messages originated on SYB.

On the system that issues the message, the message flow occurs as it does for a single system. If the message flows to other systems in the sysplex, sysplex services directs the message to the subsystems for processing, but the message bypasses the message processing exits and the hardcopy log on the target systems.

The following summarizes message flow through a sysplex:

1. The system issues a WTO or WTOR message.
2. The message processing exits on the system that issues the message get control.
3. The subsystems can receive the message on the system that issues the message.
4. Hardcopy log processing on the system that issues the message can record the message.
5. Sysplex services can route the message to consoles on other systems.
6. The subsystems on each receiving system can receive the message.
7. The MCS, SMCS and extended MCS consoles on the appropriate system can receive the message for display.

As a result of this message flow in a sysplex, message processing that occurs through exits is possible only on the system that issues the message. You need to keep this in mind when you plan your installation exits for messages. Similarly, the hardcopy recording of the message occurs only on the issuing system.

On the other hand, subsystems like NetView can receive the message on both the issuing system and any receiving system where NetView is installed. You can route messages to NetView on any system in order to control message automation for the system, but the NetView subsystems must coordinate automation for the sysplex based on the scope of the message flow to systems in a sysplex. (For planning automation using NetView, see *NetView Automation: Planning*.)

Message Recovery Following System Failures

Individual systems in a sysplex sometimes fail while the remaining systems continue to function normally. When planning recovery for a sysplex, consider the systems to which consoles are attached. To ensure that your operators receive needed messages during system failures, IBM recommends that you configure your consoles for message recovery following system failures. This will ensure that the message traffic that was being routed to a console on the inactive system is routed to active consoles on other systems.

Such coverage is especially important for controlling applications that run on multiple systems, or that manage a sysplex-wide resource (such as a JES2 checkpoint data set). These applications are more likely to direct important messages to consoles on other systems in the sysplex.

Here are two possible ways to configure your consoles for message recovery in a sysplex:

- For each console you want to cover, choose an alternate console group that includes consoles attached to other systems in the sysplex.
- Have consoles on one or more systems in the sysplex serve as “catch all” consoles, which display undelivered action messages and WTORs from all systems.

The following sections explain these techniques in greater detail.

Defining Alternate Console Groups for Message Recovery

Have each console in the sysplex belong to an alternate console group which contains at least one console on another system in the sysplex. This approach is recommended if your consoles do not normally receive messages from other systems (that is, the console is defined with **MSCOPE(*)**).

The more systems you use for your alternates, the better coverage you have for failures that span multiple systems. To be covered for all possible combinations of system failures, define an alternate console on every system that has consoles.

Specify the name of a group of consoles that can serve as back-ups on the **ALTGRP** parameter on the **CONSOLxx** parmlib member, where the group names are defined in the **CNGRPxx** member. In the **CNGRPxx** parmlib member, define **NAME** as the group of alternate consoles, and define the console names (in order of priority) on the **MEMBERS** parameter. For example, if the alternate group you want your messages routed to is named **JES1**, and the names of the consoles in group **JES1** were **CON1**, **CON2**, and **CON3**, and that is the ranked order of the consoles that you want your recovered messages sent to, you would code the **CNGRPxx** parmlib member:

```
NAME(JES1) MEMBERS(CON1,CON2,CON3)
```

In the **CONSOLxx** member, on the **CONSOLE** statement for each console, specify its alternate group:

```
ALTGRP(JES1)
```

The advantage of using alternate console groups is that your alternate message recovery consoles do not have to receive all of the other system’s messages on an ongoing basis. The selected alternate console will receive the target console’s messages only if the target console becomes inactive.

Defining “Catch All” Consoles for Message Recovery

In **CONSOLxx**, code **MSCOPE>(*ALL)** and **UD=YES** for each console that is to receive important undelivered (UD) action messages and WTORs from all systems in the sysplex. To have full coverage for any combination of system failures, define at least one console with **MSCOPE(*ALL)** and **UD=YES** for each system in the sysplex.

The advantage in this technique is that one “catch-all” console will display important “action” messages and WTORs that were directed to a console on the inactive system.

The disadvantages to this technique are:

- the console will receive all messages from all systems that are directed by route code.
- the console is not defined as the alternate console, however, so any informational messages sent to the console on the inactive system will still be lost.

Commands in a Sysplex

For commands in a sysplex, you need to consider:

- Command scope in the sysplex. You can route commands to one or more systems in the sysplex for processing.
- Type of command routing.

You can route commands in the following ways:

- Using the CMDSYS keyword in CONSOLxx. CMDSYS allows an operator to enter the command from the console and have the command automatically routed to another system for processing. (See “Using CMDSYS on the CONSOLE Statement” on page 109.) Thus, you can define your MCS or SMCS consoles with CMDSYS in CONSOLxx to control command routing in the sysplex. For extended MCS consoles, you can use RACF or MCSOPER to define CMDSYS values. (See Table 1 on page 7.) Operators can use CONTROL V to control CMDSYS.
- Using the command prefix facility (CPF). CPF allows you to identify a unique command prefix for each system or subsystem in the sysplex. (See “Using the Command Prefix Facility” on page 110.) CPF allows you to define prefixes for commands so that the operator can enter the command from any console in the sysplex and expect the command to run on the appropriate subsystem.
- Using the MVS ROUTE command. ROUTE specifies another command to be routed to one or more target systems for processing (see “Using the ROUTE command” on page 109 for details about ROUTE).
- Using the L= operand on certain MVS commands like CONTROL or DISPLAY. L= allows the operator to specify a target console name for a console on any system in the sysplex. (See “Using the L=Operand on Certain Commands” on page 111.) For example, the operator can enter the CONTROL command with L= on one console to change the console characteristics of another console on a different system.

Command Flow, CMDSYS and CPF in a Sysplex

If you specify CMDSYS or use CPF to route commands, consider the following:

- When the operator enters the command from the console, sysplex services can route the command to the system specified by CMDSYS or CPF.

Note: Processing for the MVS ROUTE command is different; see “Command Flow and the ROUTE Command in a Sysplex” on page 102.

- The system that issues the command and the system that receives the command can process the command as follows:
 1. Hardcopy log processing on both systems can record the command.
 2. The command processing exit or exits of both systems get control.

3. Hardcopy log processing on both systems can record the command if exit processing modified the command.
4. If any installation exit processed the command, no further command processing occurs.
5. The subsystems on both systems receive the command.
6. The command processor for the command on each subsystem can process the command. If the subsystem processes the command, no further command processing occurs.
7. The MVS command processor for the command on both systems processes the command.
8. If the command contains system symbols, the *system that receives the command* (not the system on which the command was entered) substitutes text for the system symbols.

For example, suppose your installation defines the command prefix S02 to system SYS2 and you enter the following command on system SYS1:

```
S02 START CICS,JOBNAME=CICS&SYSNAME.,...
```

First, system SYS1 sends the command to system SYS2. Then SYS2 substitutes the text that it has defined to the &SYSNAME system symbol:

```
START CICS,JOBNAME=CICSSYS2,...
```

9. If a command that is specified in the COMMNDxx parmlib member contains system symbols, the system does not substitute text for the system symbols during parmlib processing. The system that receives the command substitutes text for the system symbols when it processes the command.

Unlike message installation exits, command installation exits receive control on both the system that issues the command and the system that is the target of the command.

As with messages, NetView on any system can receive the routed command. To coordinate command activity for automated operations, you must consider the scope of the command flow in a sysplex.

Command Flow and the ROUTE Command in a Sysplex

If you use the ROUTE command, consider the following:

- The MVS ROUTE command is made up of two parts: the ROUTE command along with the target system(s) in the sysplex and a second command to be routed to the specified system(s).
- The system that issues the command processes the ROUTE part of the command as follows:
 1. Hardcopy log processing for the system can record the ROUTE command.
 2. The command processing exits of the system get control.
 3. Hardcopy log processing for the system can record the ROUTE command if exit processing modified the command.
 4. If any installation exit processed the ROUTE command, no further command processing occurs and sysplex services routes the second part of the command to the appropriate system.
 5. The subsystems on the system receive the ROUTE command.
 6. If any subsystem processed the ROUTE command, no further command processing occurs and sysplex services routes the second part of the command to the appropriate system.
 7. The command processor for ROUTE on the system processes the command.

8. Sysplex services can route the second part of the command to the appropriate system for processing.
 9. If CMDSYS is active for your console, a ROUTE command overrides but does not change the CMDSYS system.
- Each system that receives the routed command processes it as follows:
 1. Hardcopy log processing on the system can record the routed command.
 2. The command processing exits of the system get control of the command (for example, a DISPLAY command specified on ROUTE).
 3. Hardcopy log processing on the system can record the routed command if exit processing modified the command.
 4. If any installation exit processed the routed command, no further command processing occurs.
 5. The subsystems on the system receive the routed command.
 6. If any subsystem processed the routed command, no further command processing occurs.
 7. The MVS command processor for the command on the system processes the routed command.

For example, if a ROUTE command specifies a DISPLAY command and the operator enters the command from SYA, hardcopy log processing for SYA can log the ROUTE command. If the routed part of the command (DISPLAY) is intended for SYB, hardcopy log processing for SYB can log the DISPLAY command. The installation exits of SYA can process ROUTE, and the installation exits of SYB can process DISPLAY. Subsystems on SYA receive the ROUTE command, while sysplex services directs the routed DISPLAY command to SYB where the subsystems on SYB receive DISPLAY. The command processors for ROUTE on SYA and for DISPLAY on SYB can process the command.

Note: In ROUTE commands that specify system symbols, the system on which the command is entered processes the system symbols in the ROUTE portion of the command. The system to which the command is routed processes the remaining portion. See the description of the ROUTE command in *z/OS MVS System Commands* for details.

Command Flow and the L= Operand in a Sysplex

If you enter a command from a console on one system and specify L= to affect a console on another system, consider the following:

- The system that issues the command processes the command as follows:
 1. Hardcopy log processing for the system can record the command.
 2. The command processing exits of the system get control of the command.
 3. If exit processing modified the command, hardcopy log processing on the system can record the command.
 4. If any installation exit processed the command, no further command processing occurs.
 5. The subsystems on the system receive the command.
 6. If any subsystem processed the command, no further command processing occurs.
 7. The command processor for the command on the system processes the command.
 8. Sysplex services routes the command to the appropriate system where the target console is attached.

- Any system that receives the command processes it as follows:
 1. Hardcopy log processing for the system can record the command.
 2. The command processor on the system processes the command.

For example, if the operator issues a CONTROL command from CONS1 on SYA to change the display area of CONS2 on SYB, hardcopy log processing occurs for both SYA and SYB. The installation exits of SYA can get control. Subsystems on SYA receive the CONTROL command. The command processor for CONTROL processes the command.

Sysplex services directs the CONTROL command to SYB where SYB logs the command and changes the console display area for CONS2 on SYB. (Only the installation exits on SYA are able to process the command.)

Note: Do not use system symbols on the L= parameter on the ROUTE command.

Routing Messages

You can define routing codes and message levels to a specific console so that the console receives the appropriate messages indicated by the routing code or message. For MCS or SMCS consoles, you define routing codes and message levels in CONSOLxx. Your security or TSO/E administrator defines routing codes for users of extended MCS consoles.

Sometimes a message is issued without any assigned routing information. You can define default routing codes for these messages in CONSOLxx.

Whenever MVS cannot deliver action messages, important informational messages, or WTOR messages to an active console in a system or sysplex, MVS sends the messages to any console with the UD attribute for display. Specifying UD ensures that the messages are not lost and will appear not only on the master console but on any console with the UD attribute in the system or sysplex.

You can optionally define the hardcopy log device with the UD attribute. If no console can display undelivered messages, the hardcopy log can record them. "Defining the UD Attribute for Consoles" on page 107 describes how you define a console to receive and display undelivered action messages, important informational messages, or WTOR messages.

You can also define a group of consoles eligible to receive and display synchronous messages that bypass normal message queuing. "Display of Synchronous Messages" on page 51 describes how you can define console groups for synchronous messages.

Altering some console attributes might cause a message loss, UD loss, or SYNCHDEST loss. If a loss occurs, MVS issues a DISPLAY CONSOLE,HCONLY command and message IEE889I. You need to understand that this can happen and can affect automation.

The potential for this situation to occur comes from using these commands:

```
VARY CN  
VARY CONSOLE  
CONTROL V,LEVEL  
SWITCH CN
```

Defining Routing Codes

Most messages have one or more routing codes. The system uses these codes, decimal numbers from 1 to 128, to determine which console or consoles should receive a message. You can assign routing codes to consoles in a system or sysplex so that the appropriate messages are routed to the right console. In a sysplex, messages are routed from any system to consoles with the matching routing characteristics. To limit message routing in a sysplex, you can use the MSCOPE keyword on the CONSOLE statement. See “Directing Messages from Other Systems to a Console in a Sysplex” on page 108.

Use the following keyword on the CONSOLE statement to define routing codes for an MCS or SMCS console:

ROUTCODE Defines the routing codes in effect for the console.

The default is NONE; ROUTCODE(NONE) for a console without master authority means that the system assigns no routing codes to the console. ROUTCODE(NONE) for a console with master authority means that the system sends messages with routing codes 1 and 2 to the console. If you specify ALL, the system sends messages with routing codes 1 to 128 to the console. For a description of routing codes, see *z/OS MVS Routing and Descriptor Codes*.

For every routing code (except routing code 11), you should ensure that there is a receiving console. If there is no receiving console for a routing code, the system issues messages to inform you. (Operator consoles do not need to receive routing code 11, which indicates programmer information.)

Routing codes do not appear with a message at a console; routing codes 1 through 28 do, however, appear on the hardcopy log. To see the routing codes each console receives in a system or sysplex, operators can use the DISPLAY CONSOLES command.

To route all messages with routing codes 1, 2, 9, and 10 to MSTR2, code the following CONSOLE statement in the CONSOLxx member:

```
CONSOLE DEVNUM(81D) NAME(MSTR2) AUTH(MASTER) ROUTCODE(1,2,9,10)
```

Notice in the example that the console has master authority and that an operator can issue any MVS command from it. This console is not required to receive tape, DASD, or teleprocessing messages so the routing codes for those messages are omitted. In a sysplex, this console receives messages with defined routing codes 1, 2, 9, and 10 from all active systems unless MSCOPE limits the scope.

For users of extended MCS consoles on TSO/E, the security or TSO/E administrator can define routing codes 1 through 128. See “Controlling Extended MCS Consoles Using RACF” on page 152.

Operators can use the VARY CN command to change routing codes for both MCS and extended MCS consoles.

Handling Messages without Routing Codes

For queuing messages that have no defined routing codes, descriptor codes, or console destination, you can use DEFAULT ROUTCODE. Use the following keyword on the DEFAULT statement of CONSOLxx for messages that have no routing code information:

ROUTCODE Defines the routing codes for messages that do not have them.

You can assign any combination of routing codes from 1 through 128. If you specify ROUTCODE(ALL), the system assigns routing codes 1 through 128; if you specify NONE, the system does not assign any routing codes. If you do not code ROUTCODE on the DEFAULT statement, the default for messages without assigned routing codes is the range of routing codes 1 through 16.

Defining Message Levels for a Console

Assigning routing codes is one way to limit message traffic to a console. You can further reduce the number of messages that appear on a console by directing certain messages to consoles by message levels. Descriptor codes can also appear with messages and further describe the significance of the message levels.

The system differentiates among the following kinds of message levels:

- Write-to-operator with reply (WTOR) messages, which might demand an immediate reply.
- System failure and immediate action messages (descriptor codes 1 and 2), which indicate a serious error or that a task is awaiting a requested operator action.
- Critical eventual action messages (descriptor code 11), which indicate that an eventual action of critical importance is requested on the part of the operator.
- Eventual action messages (descriptor code 3), which request an eventual action that does not require immediate operator attention.
- Broadcast messages, which are messages normally sent to every active console regardless of the routing code you assigned to the console.
- Informational messages, which generally indicate system status. (Most messages are informational. MVS recognizes informational messages with descriptor code 12 for special routing.)

Descriptor codes and message levels

The system gives special consideration to messages with descriptor codes 1, 2, 3, 11, 12, and WTOR messages. Consoles with the UD attribute will receive these messages for display if they could not be queued to any active console which could have received the message. See “Defining the UD Attribute for Consoles” on page 107.

MVS also handles messages with descriptor code 13 in a special way. If a message has been specified for automation in MPF, you can assign descriptor code 13 to the message in a message processing exit (like IEAVMXIT) to indicate that the message has been previously automated. You can then reissue the message. Descriptor code 13 can be useful when a message has been automated on one system in a sysplex but needs to be reissued to other systems in the sysplex.

To limit message routing in a sysplex, you can use the MSCOPE keyword on the CONSOLE statement. See “Directing Messages from Other Systems to a Console in a Sysplex” on page 108.

To define message levels for a console, use the following keyword on the CONSOLE statement:

LEVEL Defines the message level in effect for the console.

Assignment by message level means that a console can accept combinations of action, broadcast, and informational messages that the system sends to a console. Options you can specify for LEVEL include the following:

- R** Messages that require an operator reply are to appear
- I** Immediate action messages (descriptor codes 1 and 2) are to appear
- CE** Critical eventual action messages (descriptor code 11) are to appear
- E** Eventual action messages (descriptor code 3) are to appear
- IN** Informational messages are to appear
- NB** Broadcast messages are **not** to appear
- ALL** All messages, including broadcast messages, are to appear

You can specify one or any combination of these options for LEVEL. If LEVEL in the CONSOLxx member is not coded, the system sends all messages, including broadcast messages, to the console.

To direct only WTOR messages and immediate action messages to a console named ACCT, code this statement in CONSOLxx:

```
CONSOLE DEVNUM(0C6) NAME(ACCT) LEVEL(R,I)
```

Operators can use the CONTROL V command to change LEVEL.

Specifying Message Levels and Routing Codes for a Console

The following example illustrates the relationship between the routing codes and the message levels assigned to a console named TDISK:

```
CONSOLE DEVNUM(81D) NAME(TDISK) ROUTCODE(5,6) LEVEL(R,IN)
```

In the example, TDISK receives informational messages directed to the tape libraries (routing code 5) and disk libraries (routing code 6). In a sysplex, console TDISK receives messages with these defined message levels from all active systems unless MSCOPE limits the system scope.

Defining the UD Attribute for Consoles

If action messages (those with descriptor codes 1, 2, 3, or 11), informational messages with descriptor code 12, or WTOR messages are not delivered to any console in the configuration, you can specify that the system display these messages on MCS, or SMCS consoles or on extended MCS consoles in a system or sysplex. Thus, operators can view these important messages on any console without having to rely only on the master console.

To specify that the system direct undelivered messages to a console, use the following keyword on the CONSOLE statement:

- UD** Specifies if you want to define the console to receive undelivered action messages, informational messages with descriptor code 12, and WTOR messages.

If you specify UD(Y), the console has the UD attribute and will display these messages. UD(N) is the default for any console that is not the master console and indicates that the console not display the messages. Note that the master console in the system or sysplex by default displays action messages, messages with descriptor code 12, and WTOR messages.

You can also specify UD(Y) on the HARDCOPY statement. Specifying UD on HARDCOPY prevents the system console from receiving undelivered messages.

When you specify UD(N) for HARDCOPY, if the system directs undelivered action messages, informational messages with descriptor code 12, or WTOR messages to the system console.

Operators can use the VARY CN command to change the UD attribute for a console or the VARY,HARDCOPY command to change the UD attribute for the hardcopy log.

Directing Messages from Other Systems to a Console in a Sysplex

In a sysplex, if you don't want your operators receiving certain messages from all systems, you can limit some of the messages they receive. These messages are any messages not explicitly routed to a console.

Use the following keyword on the CONSOLE statement to direct certain messages in a sysplex to a given console:

MSCOPE In a sysplex, defines the systems that can send messages to this console.

The default is MSCOPE(*ALL), which indicates that messages from the local system as well as all other systems in the sysplex appear on the console. If a system is specified on MSCOPE but is not active, the console does not receive any messages.

MSCOPE values override other routing attributes for the console; that is, the console receives messages only from the system you specify. However, if MSCOPE limits system scope, you can still send messages from other systems using the console name on commands and macros. Operators can use the VARY command to change MSCOPE.

Replying to Messages from Other Systems in a Sysplex

You can use the MSCOPE keyword to control which consoles can reply to messages issued from consoles that are defined to other systems in the sysplex. To use a console to reply to such messages, include the other system's name on the MSCOPE keyword of the CONSOLE statement in the CONSOLxx parmlib member for this system.

Directing Messages that Are Eligible for Automation to Extended MCS Consoles

You can specify a message as eligible for automation. In MPFLSTxx, you can specify AUTO(YES) for the message, or you can use message processing exits to indicate that the message is eligible for automation.

For any message that is eligible for automation, you can define an extended MCS console to receive the message for processing. To allow an extended MCS console to receive all messages that are eligible for automation, you define the console attribute through the MCSOPER macro or RACF using the OPERPARM keyword. In a sysplex, you can also define which systems are to direct the messages to the extended MCS console by specifying the OPERPARM console attribute for MSCOPE.

Using an extended MCS console in conjunction with an automation program like NetView can help you plan your automated operations for a system or sysplex. For information on extended MCS consoles, see "Extended MCS Consoles" on page 6. For information on MCSOPER, see *z/OS MVS Programming: Authorized Assembler*

Services Reference LLA-SDU. For information on NetView and automation planning, see *NetView Automation: Planning*.

Routing Commands

In a sysplex, you can route commands to other systems for processing in the following ways:

- “Using CMDSYS on the CONSOLE Statement”
- “Using the ROUTE command”
- “Using the Command Prefix Facility” on page 110
- “Using the L=Operand on Certain Commands” on page 111

Using CMDSYS on the CONSOLE Statement

Use the following keyword on the CONSOLE statement to define command association between a console and a system in a sysplex:

CMDSYS Defines the system in a sysplex where you want to send commands entered on this console for processing.

Defining your consoles through this kind of command association can help your operators view a particular system in the sysplex and limit activities to that system. The default is CMDSYS(*), which indicates that commands entered on the console are processed on the local system where the console is defined. If a system specified for CMDSYS is not active, the console receives an error message whenever the operator enters a command. Operators can use the CONTROL V command to change CMDSYS for MCS, SMCS, and extended MCS consoles.

To let SYA direct commands entered on an attached console called TAPE to SYB, code the following statement in CONSOLxx for SYA:

```
CONSOLE DEVNUM(243) NAME(TAPE) CMDSYS(SYB)
```

If a different system is specified for CMDSYS and an operator issues the MSGRT command from the console, the command is not processed on the system specified. MSGRT initiates status displays for TRACK, CONFIG, or DISPLAY commands for the system on which MSGRT is entered. Operators can use the ROUTE command to send MSGRT to another system for processing. Other commands not processed on a different target system specified on CMDSYS for the console include:

- All CONTROL commands except for CONTROL M
- LOGON/LOGOFF
- TRACK/STOPTR
- ROUTE

For example of how these commands operate in a sysplex, see “Commands in a Sysplex” on page 101.

Using the ROUTE command

Your operators can use the ROUTE command to send commands to other systems not specified on CMDSYS without changing the CONSOLE statement values in the CONSOLxx Parmlib member. In the following example, an operator wants to route the CANCEL command to SYB to cancel the job JOBPRINT:

```
ROUTE syb,CANCEL JOBPRINT
```

The ROUTE command directs a command to the system you specify, to all systems in a sysplex, or to a group of systems in a sysplex. ROUTE with the specified

command overrides but does not change the values you code for CMDSYS on the CONSOLE statement. In a sysplex, both systems invoke the command installation exits if they are installed. The exit on the issuing system handles the ROUTE part of the command; the command installation exit on the receiving system processes the command that ROUTE specifies.

For complete syntax information on the ROUTE command, and for the list of commands you should not route to multiple systems, see *z/OS MVS System Commands*.

The aggregated command response is logged on the system that processes the ROUTE *ALL or ROUTE *systemgroupname* command. This aggregated response is seen by the same system's MPF exits or user exits, and can be automated. The system that processes the ROUTE command is the system where the ROUTE command is issued, unless the ROUTE command was transported using CMDSYS (command association) or CPF (command prefix facility). The responses to the individual command that is imbedded inside of ROUTE *ALL or ROUTE *systemgroupname* are logged on the systems where the command is processed. The individual responses are seen by each target system's MPF exits or user exits, and can be automated. While the aggregated command response is logged with the issuing console's name, each individual response is logged with a system generated console name.

Setting up a System Group Name

You can define groups of systems to MVS by placing a list of systems in ECSA, and addressing the list using the name/token services. A ROUTE command that specifies the name of this group will cause a command to be routed to all active systems in the group.

For more information on setting up name/token pairs, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

The program that creates the list can be set up in PROCLIB, and can be run on each system in a sysplex at IPL using a START command in a COMMNDxx parmlib member. For ease of use, the COMMNDxx parmlib member can be shared by all systems.

To define a new or changed set of named system groups on all systems in the sysplex, use the command ROUTE *ALL,START *jobname*, where *jobname* is the name of the procedure that runs the program that creates or deletes the groups.

IBM provides a SYS1.SAMPLIB member to define named system groups.

For more information, see the comments in SYS1.SAMPLIB member IEEGSYS.

Using the Command Prefix Facility

The MVS command prefix facility (CPF) allows a subsystem (like JES2 or DB2) to create unique command prefixes for each copy of the subsystem in the sysplex and control which systems can accept the subsystem commands for processing. For example, using the JES2 CONDEF initialization statement, an installation can define a JES2 command prefix with sysplex scope. No matter which system an operator uses to enter the JES2 command, MVS can recognize the prefix and direct the command to the system where the prefix has been defined.

For information on the JES2 initialization statement that uses the command prefix facility, see *z/OS JES2 Initialization and Tuning Reference*. For information on the CPF macro that other subsystems or application programs can use to issue commands in a sysplex, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Defining a System Name as a Command Prefix

You can run IEECMDPF (an IBM-supplied sample program in SYS1.SAMPLIB) to define the system name as a command prefix that substitutes for the ROUTE command on each system.

For example, if you run IEECMDPF on system S01, then the following have the same effect on each system in the sysplex:

```
ROUTE S01,command
S01 command
S01command
```

Note: If the system name does not define a valid system, ROUTE name processing does not return an error message.

In a sysplex, if you put a START command into a common COMMNDxx Parmlib member, you could have a short-form ROUTE function for each system in the sysplex. Then from any system in the sysplex, any of the following would route a command to system S02:

```
ROUTE S02,command
S02 command
S02command
```

Using the L=Operand on Certain Commands

The L= operand on an MVS command (like CONTROL, DISPLAY, and MONITOR) allows an operator to specify a console name (or console id) for a console defined on a different system in the sysplex. Sysplex services can route the command to the system where the console is attached. For syntax of CONTROL, DISPLAY, or MONITOR, see *z/OS MVS System Commands*.

Sharing System Commands By Using System Symbols

MVS allows two or more systems in a multisystem environment to share commands while retaining unique values in those commands. When two or more systems share commands, you can view a multisystem environment as a *single system image* from which you can perform operations for several different systems.

This section explains how to plan for sharing system commands in a multisystem environment. It:

- Describes what system symbols are, and explains how they are used to represent the unique values in shared commands
- Describes what wildcards are, and explains how they are used to identify multiple resource names in commands
- Provides planning tasks for sharing system commands
- Provides tips for sharing commands in a multisystem environment.

For information about using system symbols in system commands, including lists of system symbols that the system provides, see *z/OS MVS System Commands*.

What Are System Symbols?

System symbols represent the values in shared commands that are unique on different systems. Each system defines its own values for system symbols; it replaces the system symbols with those values when it processes shared commands.

For detailed information about system symbols, including lists of system symbols that you can specify in system commands, see the section on system symbols in *z/OS MVS Initialization and Tuning Reference*.

What Are Wildcards?

Wildcards are characters that indicate a command applies to all resources whose names match a specified character string.

The asterisk (*) wildcard tells the system to match zero or more specified characters, up to the maximum length of the string. An asterisk can start the character string, end it, appear in the middle, or appear in multiple places in the string. A single * for the name indicates that all resource names for the particular field are to match.

For some values, the * must be a suffix and cannot appear alone. See *z/OS MVS System Commands* for examples of how to use wildcards in system commands.

Planning to Share System Commands

When planning to share system commands among different systems, ask yourself the following questions:

1. What resources are good candidates for sharing?

If your goal is to greatly simplify your operating environment, the answer is: As many as possible! If two or more systems require different names for a resource, chances are that you can use a single system symbol to represent the characters in the name that must be unique. If you have one “skeleton” that represents the unique names, you have one convenient place to maintain the resource definition. If you follow the same process with all commands that require unique values, you can view a multisystem environment as a *single system image* with one point of control.

Be aware that there are also reasons why you might *not* want to share certain commands. Perhaps the release level of MVS prevents you from using a resource on a particular system; or perhaps one or more systems do not require a particular resource. Whatever the case, your installation must examine the commands that are issued frequently and determine the extent to which they can be shared.

2. What commands support system symbols?

All MVS/ESA SP 5.2 commands support system symbols, with the exception of:

- The LOGON command
- The VARY CN(*),ACTIVATE form of the VARY command (all other forms of VARY support system symbols).

3. Do I want a job to have different names on each system where it runs?

If a job runs on two or more systems in a multisystem environment, IBM recommends that you use different jobnames for each instance of the job. Different jobnames allow you to easily identify the system on which a job runs.

The best way to explain how to use one command to start jobs with different names on different systems is through an example. Suppose your installation is to start Customer Information Control System (CICS) on each system in a sysplex and assign a different jobname to each instance of CICS. First your

installation establishes a consistent naming convention for the instances of CICS. For example, the jobname for each instance of CICS always begins with the characters **CICS** and ends with the last four characters of the system name. You can specify the &SYSNAME system symbol in the START command and route the command to all systems that require CICS:

```
ROUTE *ALL S CICS,JOBNAME=CICS&SYSNAME.,...
```

Each system substitutes the text it has defined to &SYSNAME into the command text. Assuming that you route the START CICS command to two systems named SYS1 and SYS2, the following commands result:

```
S CICS,JOBNAME=CICSSYS1,...  
S CICS,JOBNAME=CICSSYS2,...
```

Your installation can also specify system symbols in commands that are entered at system initialization using the COMMNDxx parmlib member. See the description of the COMMNDxx parmlib member in *z/OS MVS Initialization and Tuning Reference* for information about how the system processes system symbols in COMMNDxx.

Sharing Commands that Flow Through Multiple Systems: When you specify system symbols in commands that flow through several systems in a multisystem environment, the *target system* almost always substitutes text for the system symbols in the command text. This is true for the main ways to route commands to other systems:

- The CMDSYS keyword in CONSOLxx, which allows operators to enter commands from a console and have the commands automatically routed to another system for processing. The command is first transported to the system that has command association to the system on which the command is entered; then substitution takes place. See “Using CMDSYS on the CONSOLE Statement” on page 109 for more information.
- A CPF prefix, which allows operators to send commands to a system in a sysplex for which a unique prefix is defined. If a command has a CPF-defined prefix, the command is first transported to the system that has the prefix; then substitution takes place. See “Using CMDSYS on the CONSOLE Statement” on page 109 for more information.
- The ROUTE command, which allows operators to send commands to other systems for processing. The command is first routed to the other system; then substitution takes place. See “Using the ROUTE command” on page 109 for more information.

If a command is entered on one system, and the command affects an entity (such as a console) on another system, the *target system* almost always substitutes text for the system symbols in the command text. The DUMPDS, REPLY, and ROUTE commands have exceptions to these rules. See the descriptions of those commands in *z/OS MVS System Commands* for more information.

For example, suppose the following command changes the routing codes for a console on a different system from which the command is entered:

```
VARY CN(consname),ROUT=&SYSVAR1.
```

If the value of &SYSVAR1 is (1,2) on the system where the command was issued, and &SYSVAR1 is (3,4) on the system where the console *consname* is attached, the result of the system symbol substitution is:

```
VARY CN(consname),ROUT=(1,2)
```

For commands that accept the **L=cca** keyword, which specifies that the command output messages are to be directed to a different console, the system on which the command is entered substitutes text for system symbols in the command text (not the system where the **L=cca** console is attached).

MPF and MVS Operations Planning

The message processing facility (MPF) controls message processing for an MVS system. It controls the following:

- Message presentation (the color, intensity and highlighting, of messages) for an MCS or SMCS console
- The suppression of messages
- The retention of messages for the action message retention facility
- The selection of messages for automation programs like NetView
- Message processing exits other than IEAVMXIT that gain control when certain messages are issued
- Command installation exits that gain control when commands are issued

You can specify presentation options, message retention, message suppression, selection of messages by an automation program, and user exit information in the MPFLSTxx Parmlib member.

Specifying MPFLSTxx members

At IPL, the system uses the MPFLSTxx member or members indicated on the MPF keyword on the INIT statement in CONSOLxx. You can specify multiple MPFLSTxx members on the MPF keyword. In a sysplex, MPF processing has system scope; thus, you must plan MPF processing on each system in the sysplex.

Using multiple members allows your installation to define separate MPF members to handle specific message processing functions for messages. For example, you might specify two members of MPFLSTxx to handle different automation procedures. Or you might have one MPFLSTxx member handle messages for suppression and another to handle messages for automation for a system. (Note that the system default allows the system to consider all messages as eligible for automation.) Operators can use the SET MPF command to activate these members as needed (for example, during shift changes or for workload balancing).

If you do not have an active MPFLSTxx member:

- Default options for message presentation are in effect (all messages are eligible for automation).
- The action message retention facility, if it is active, retains all action messages (those with descriptor codes 1, 2, 3, 11, and WTOR messages).
- MPF does not suppress messages.
- No installation exit except IEAVMXIT can gain control to process messages.

MPF Options

Use the following keyword on the INIT statement of CONSOLxx to activate the MPFLSTxx member or members at your installation:

MPF Specifies whether you want to activate the message processing facility at your installation.

You can specify one or more 2-character suffixes for the MPFLSTxx members you want to activate at IPL, or NO, in which case, MPF is not active. MPF(NO) is the default. Operators can use the SET MPF command to change the status of MPF.

The following sections contain information about options you control in MPFLSTxx:

- For presentation options, see “Specifying Message Presentation”
- For message suppression options, see “Suppressing Messages”
- For message retention options, see “Retaining Messages” on page 116
- For message automation options, see “Selecting Messages for Automation” on page 119
- For message and command processing exits, see “Installation Exits for Messages and Commands” on page 121.

Specifying Message Presentation

Using MPFLSTxx and installation exits, you can control how you want messages to be presented on console screens. You can control color for messages, how you want to highlight messages, or specify the intensity of messages to make them stand out on the screen.

To specify color, highlighting, and intensity for messages, you can use the following statement in MPFLSTxx:

.MSGCOLR Controls message presentation

Options that you can use for **.MSGCOLR** are as follows:

- | | |
|-----------------|--|
| msgarea | Allows you to specify color, highlighting, and intensity for message displays |
| DEFAULT | Specifies that you want to use the IBM supplied defaults for color, highlighting, and intensity for message displays |
| NOCHANGE | Specifies that you want to use the values for color, highlighting, and intensity established in the previous MPFLST member in effect; NOCHANGE is the default. |

Various values for *msgarea* allow you to specify color, highlighting, and intensity for the entry area, for different message types or descriptor codes (action messages or WTOR messages, for example), for control lines or data lines, for status displays, and other screen controls.

z/OS MVS Initialization and Tuning Reference contains complete information about IBM defaults for color, highlighting, and intensity in MPFLSTxx.

You can further control color, highlighting, and intensity through installation exits like IEAVMXIT or other exits that your installation can define. You can change the message presentation information (color, highlighting, and intensity) for the console through a parameter list (CTXT) passed to the message processing exit. In the exit, you can modify specific fields in CTXT that control color, highlighting, and intensity.

z/OS MVS Installation Exits contains complete information about IEAVMXIT.

Suppressing Messages

For a multisystem environment like a JES3 complex or a sysplex, the large volume of messages produced by various systems makes message suppression an important part of your operations planning. But even for a single system, IBM recommends that you suppress informational messages that the operator does not need to see to manage the system.

Suppressed messages do not appear on any console; however, they do appear on the hardcopy log. If you use MPF to suppress messages, the hardcopy log must be active.

Message suppression is also important when you plan automation for an installation. The goal of automated operations is to streamline message flow and simplify operator actions at a console. Suppressing messages operators do not need to see is a good way to start your MVS automation planning. In a sysplex environment, NetView can make use of extended MCS consoles to help manage message automation for any system in the sysplex. For more information about automated operations, see *NetView Automation: Planning*.

Note that if you specify a message for automation and suppression using MPF, you can still deliver the message to an extended MCS console for display. When you activate the extended MCS console with the automation attribute, you allow the console to receive automated messages whether MPF indicates that the message is suppressed or not.

Through the MPFLSTxx Parmlib member, you can specify which messages the system is to suppress. Using the *msgid* parameter with the SUP option, you can select certain messages for suppression, or specify suppression for all messages. For further information about MPFLSTxx and examples of the kinds of messages your installation might decide to suppress, see *z/OS MVS Initialization and Tuning Reference*.

Using MPFLSTxx, you can select messages to suppress from display. To select messages for suppression using MPFLSTxx, you can use the following MPFLST parameter and its option:

msgid Specifies the id or list of ids for messages that you want to suppress

The option you can specify for the **msgid** is as follows:

SUP Specifies whether you want to suppress the message(s) identified by *msgid* for display; SUP(YES) is the default. SUP(NO) indicates that you do not want to suppress the message(s) for display. You can use SUP(YES) to suppress messages that are command responses.

z/OS MVS Initialization and Tuning Reference gives examples of the kinds of messages your installation might decide to suppress.

Retaining Messages

If your installation produces large volumes of messages for operators to monitor, it is a good idea to use the action message retention facility (AMRF). If you want operators to be able to retrieve action messages and WTOR messages that no longer appear on the console, use AMRF. AMRF keeps action messages so that the operator has a chance to see them at a later time. WTOR messages are always available for operator retrieval regardless of the state of AMRF.

Action Message Retention Facility

During initialization, the system starts AMRF if it is specified in CONSOLxx. Use the following keyword on the INIT statement of CONSOLxx to control the action message retention facility:

AMRF Specifies whether you want to activate the action message retention facility.

AMRF(Y) means you want to activate the action message retention facility and is the default. If you specify AMRF(N), AMRF is not active.

Unless you code otherwise in MPFLSTxx, AMRF retains in a buffer area all action messages, those messages with descriptor codes 1, 2, 3, and 11, and WTOR messages.

AMRF works as follows. When the operator has performed the action required by a message displayed on the screen, the system deletes the message, or the operator can use the CONTROL C command to delete the message. If AMRF is active, operators can remove action messages from the screen, then retrieve them in their entirety later by using the DISPLAY R command.

In a sysplex, it is recommended that you use AMRF. The AMRF keyword has sysplex scope.

Using MPF to Retain Messages

You can also control which action messages to retain through MPFLSTxx. Thus, you can specify on a message by message basis which messages you want the action message retention facility to retain or not. First, ensure that both MPFLSTxx and AMRF are active (either specified on the INIT statement of CONSOLxx or through the operator SET command). With MPF, you can only retain action messages (those with descriptor codes 1, 2, 3, and 11).

To specify which messages you want to retain or not in MPFLSTxx, you can use the following parameter and its option:

msgid Specifies the id or list of ids for messages that you want to suppress or retain

The option you can specify for the **msgid** is as follows:

RETAIN Specifies whether you want to retain the message(s) identified by msgid.

RETAIN(YES) is the default. RETAIN(NO) indicates that you do not want the system to retain the action message. Thus, with MPF you can indicate which action messages that AMRF retains you do not want to keep for retrieval.

Displaying Information About Messages Awaiting Action

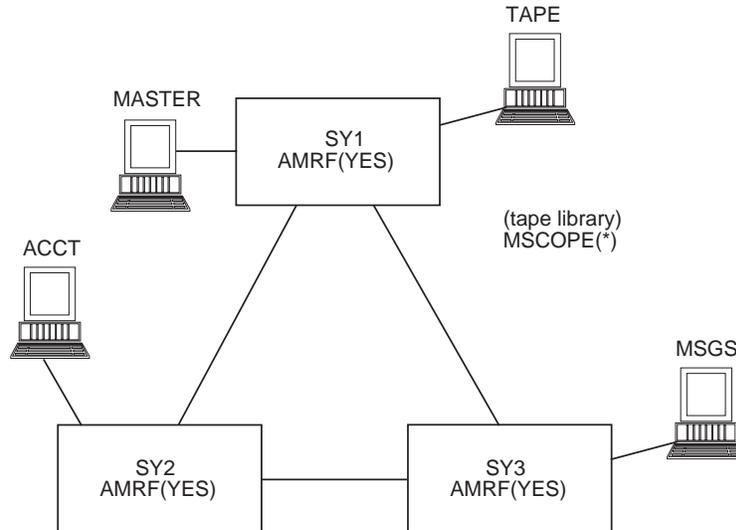
The DISPLAY R command allows an operator to display all outstanding action messages or a subset of these messages. For example, to display all the outstanding action messages at a console, an operator enters DISPLAY R,M; to display all the outstanding critical eventual action messages (descriptor code 11), an operator can enter DISPLAY R,CE.

In a sysplex, the best way to describe how to use the DISPLAY R command is through an example. Assume a sysplex has the following identifiers:

SY1	System 1 in the sysplex
SY2	System 2 in the sysplex
SY3	System 3 in the sysplex
MASTER	MCS master console attached to SY1
ACCT	MCS console attached to SY2
MSG3	MCS console attached to SY3
TAPE	MCS console attached to SY1. The console is controlling the tape

library and has an MSCOPE(*) specified. MSCOPE(*) limits the messages the console receives to SY1, the system to which it is locally attached.

The example assumes that the AMRF is active on all systems in the sysplex.



Operators can do the following:

- To see the texts and identification numbers of all outstanding action messages and WTORs destined for MASTER, enter the following command at MASTER:
`DISPLAY R,M`
- To learn the number of outstanding action messages with routing codes assigned to MASTER, enter the following command at MASTER:
`DISPLAY R,ROUT=ALL`

The message includes the total of outstanding action messages for all systems in the sysplex (SY1, SY2, and SY3) that are routed to MASTER.

- To see all outstanding action messages in the sysplex, enter the following command at MASTER, ACCT, or MSGS.
`DISPLAY R,M,CN=(ALL)`

The message includes the total of outstanding action messages for all systems in the sysplex (SY1, SY2, and SY3). AMRF has sysplex scope; if another system joins the sysplex, the action message retention facility is active no matter what is specified for AMRF on the INIT statement in CONSOLxx for that system.

- To see all outstanding action messages for the local application running on SY1, enter the following command on TAPE:
`DISPLAY R,M`

MSCOPE limits the message information in the sysplex that TAPE receives to SY1.

Grouping Messages by Function: To help you keep track of messages, your application programmer can also group and name messages by function. When AMRF is active, the WTO macro in MVS allows programs to associate a 1 to 8 alphanumeric character or “keyname” with certain messages. Operators on an MCS console can use the KEY operand on the DISPLAY R command to display all the

outstanding action messages by keynames. For example, if an application programmer assigned the characters "PAYROLL" to all payroll application messages, an operator can list all the outstanding messages for payroll messages by entering the following command from an MCS console in the system or sysplex:

```
DISPLAY R,M,KEY=PAYROLL
```

In a sysplex, you can control the scope of these messages using MSCOPE.

JES3 generally uses the dynamic support program (DSP) names as keynames to group messages by function. For information on available JES3 DSPs, see *z/OS JES3 Commands*.

Reference

For information on the MVS DISPLAY command, see *z/OS MVS System Commands*.

Selecting Messages for Automation

Using MPFLSTxx you can specify that an automation program like NetView use messages to automate certain system or operator actions on MVS.

Use the following parameter and its option in MPFLSTxx to specify an automation program like NetView:

msgid Specifies the id or list of ids for messages that you want to select for automation

The option you can specify for the **msgid** is as follows:

AUTO Specifies whether you want the automation program at your installation to handle the message(s) identified by msgid for automation

If you do not specify an MPFLSTxx member, all messages are eligible for automation. AUTO(YES) or AUTO(token) indicates that you want to use an automation program to process the message or messages. If you have defined an extended MCS console with the automation attribute, the console can receive any message that MPF has specified for automation from any system in the sysplex. See "Directing Messages that Are Eligible for Automation to Extended MCS Consoles" on page 108.

If a message has been specified for automation in MPFLSTxx, you can reissue the message with a descriptor code 13 from a message processing exit. Reissuing a message specified for automation might be useful in a sysplex where the message does not need to be automated on every system in the sysplex.

Reference

For information on using NetView to plan the automation of messages, see *NetView Automation: Planning*.

Automation in a Sysplex

Because the sysplex affects the way you use consoles to receive messages or send commands, you need to consider how sysplex functions can affect automation. Consider the following in a sysplex:

Console definitions

Console names for MCS, SMCS, and extended MCS consoles allow automation products like NetView to reference consoles throughout the sysplex regardless of their system attachment. The names must be unique for each console in the sysplex. (See “Using Console Names” on page 40.)

You can define extended MCS consoles to handle message and command processing as part of your automation in a sysplex. In a system or sysplex, defining extended MCS consoles allows you to exceed the 99-console limit for MCS and SMCS consoles. (See “Extended MCS Consoles” on page 6.)

Logging activity

Because the impact on hardcopy logging for systems in a sysplex is increased, analyzing the results of message and command logging for a sysplex becomes more complex than for a single system. For example, a message received on one system might have originated on another system where it has already been logged. How a system issues a command in a sysplex can affect how other systems log command responses. See “Message and Command Flow in a Sysplex” on page 98.

Understand that defining more consoles in a system or sysplex means that more hardcopy logging can occur. For extended MCS consoles in a sysplex, you can specify LOGCMDRESP=NO through RACF OPERPARM or on the MCSOPER macro to control logging for the console. As a result, command responses are not logged for the extended MCS console, and you can reduce the impact of hardcopy logging in the sysplex.

Note: LOGCMDRESP=NO will control logging only for messages issued by authorized programs. Messages issued by unauthorized programs are always logged.

Parmlib

Because CONSOLxx and MPFLSTxx are crucial to control message and command processing, you must define these Parmlib members so that they work together for all systems in a sysplex. For example, console attributes defined in CONSOLxx have either system or sysplex scope. As a result, these differences can affect console operations in the sysplex. MPFLSTxx has system scope so you must consider how differences in MPFLSTxx for each system might affect overall operations in the sysplex. (See “Using CONSOLxx” on page 14 and “MPF and MVS Operations Planning” on page 114.)

Message and command processing

In a sysplex, you need to consider the scope of your message and command processing. Messages and commands can flow from system to system. In order to coordinate automation functions for the entire sysplex, automation products on different MVS systems need to take this message and command flow into account. (See “Message and Command Flow in a Sysplex” on page 98.)

Installation exits for messages and commands must also take into account message and command routing in a sysplex. Although messages and commands can be routed to different systems in a sysplex, you must take into account where the message or command is issued, the systems that receive the message or command, how and when the exits get control, and when automation programs receive the message or command. These considerations can have an impact on how an automation program like NetView processes messages and commands that first pass through installation exits. (See “Installation Exits for Messages and Commands” and “Message and Command Flow in a Sysplex” on page 98.)

Installation Exits for Messages and Commands

MVS provides installation exits to allow further processing of messages and commands. Whenever these exits are active and the system issues a message, or an operator or program issues a command, the exits get control to process the message or command. For messages, MVS provides IEAVMXIT, which allows you to tailor your messages. You can also install your own message processing exits as needed. For commands, MVS provides the command installation exit that can accept, modify, or reject commands before the command processor for the command gets control.

Allocation exits can get control whenever the system issues WTOR messages to operators to cancel a waiting job, bring a device online, or allow a job to wait. These exits allow an installation to automate responses to the messages. For more information on allocation exits, see *z/OS MVS Installation Exits*.

IEAVMXIT and Message Processing

The message processing installation exit IEAVMXIT can gain control when any WTO or WTOR message is issued. In this exit, you can change routing codes, descriptor codes, and message texts and perform other message processing; you can also override the message processing facility (MPF).

If you do not code your own message processing exit, IEAVMXIT is available, and, if active, gets control when the system issues a WTO or WTOR message. If you specify your own message processing exit through MPFLSTxx, that exit gets control instead of IEAVMXIT. See “Message Processing Exits other than IEAVMXIT” on page 122.

To specify that you want to activate IEAVMXIT, use the following keyword on the INIT statement of CONSOLxx:

UEXIT Defines whether you want the installation exit IEAVMXIT to process messages

UEXIT(Y) is the default; if you do not code this parameter, IEAVMXIT will be activated if it’s installed. Operators can use CONTROL M to change the status of IEAVMXIT.

To have the user exit IEAVMXIT inactive at IPL, code the following parameter on the INIT statement:

UEXIT(N)

Reference

z/OS MVS Installation Exits describes IEAVMXIT in detail and provides a sample exit.

Message Processing Exits other than IEAVMXIT

Use the following parameter and its option in MPFLSTxx to specify an installation-defined message processing exit other than IEAVMXIT:

msgid Specifies the id or list of ids for messages that you want the exit to process

The option you can specify for the **msgid** is as follows:

USEREXIT Specifies the name of the installation supplied exit to handle messages identified by **msgid**

The exit gets control whenever the system issues the message or messages identified by **msgid**. If you do not supply an exit name, the system uses the IEAVMXIT, if it exists and is active.

Command Installation Exits

Using MPFLSTxx, you can specify MVS command installation exits to modify commands that an operator can issue at a console. You can authorize a console to use a specific command or commands, reject the command, direct the command to specific consoles for display, modify the command text, or execute the command in the exit.

Use the following parameter and its option in MPFLSTxx to specify command installation exits:

.CMD Specifies the statement that allows you to specify up to six command installation exits

The option you can specify for the **.CMD** is as follows:

USEREXIT Specifies from 1 to 6 names for command installation exits.

If you code USEREXIT but do not supply an exit name, the system issues a syntax error message.

See *z/OS MVS Installation Exits* for a detailed description of the command installation exit and a sample exit.

Considerations for a Sysplex: In a sysplex, when an operator uses the ROUTE command to direct a command to execute on a different system and the command installation exits are installed on both systems, both systems invoke the command installation exits. The exit on the issuing system handles the ROUTE part of the command; the command installation exit on the receiving system processes the command that ROUTE specifies. To understand the effect command routing in a sysplex has on the installation exits, see “Commands in a Sysplex” on page 101.

The exit changes the console authority of a console only to permit the console to enter the specified command or commands coded in the exit. The original AUTH attribute of the console is still in effect and determines the ability of the console to enter any other command. Note that RACF command profiles, if specified, override command authorization in the command authorization exits.

In a JES3 complex, use the JES3 exit IATUX18 to process JES3 commands and the MVS command installation exit to process MVS commands. For information on JES3 exits, see *z/OS JES3 Customization*.

Considerations for System Symbols: When a command contains system symbols, MVS provides the command text to command installation exits *after* it substitutes text for the system symbols. For example, if the following command is entered to display a console group on system SYS1:

```
DISPLAY CNGRP,G=(C1GP&SYSNAME.)
```

The command installation exit receives the following text:

```
DISPLAY CNGRP,G=(C1GPSYS1)
```

If a command installation exit requires the original command text (the one that existed *before* symbolic substitution), the exit can access the CMDXOLIB field in the command installation exit routine parameter list (CMDX). See *z/OS MVS Data Areas, Vol 1 (ABEP-DALT)* for a mapping.

Note: Do not use command installation exits to add or change system symbols in command text. The system cannot substitute text for system symbols that are added or changed through those exits.

See the section on sharing system commands in *z/OS MVS System Commands* for more information about using system symbols in commands.

Controlling WTO and WTOR Message Buffers

MVS places WTO and WTOR buffers in virtual storage. You can control the number of buffer areas for WTO and WTOR messages at your installation by using CONSOLxx.

To specify buffers for WTO and WTOR messages, use the following keywords on the INIT statement of CONSOLxx:

MLIM Defines the maximum number of buffers the system uses for writing WTO messages; the default is 1500.

RLIM Defines the maximum number of buffers the system uses for writing WTOR messages; the default is 10 for a single system. For a sysplex, see the following description for RMAX.

z/OS MVS Initialization and Tuning Reference provides the range of values for MLIM and RLIM. You should use an MLIM value for WTO messages that is significantly larger than the RLIM value for WTOR messages.

Controlling Reply IDs for WTOR Messages

Operators use an ID on the REPLY command to respond to WTOR messages. In CONSOLxx, you can also specify the maximum number (RMAX) for reply IDs to a WTOR message.

To specify the RMAX value, use the following keyword on the DEFAULT statement of CONSOLxx:

RMAX Defines the maximum number of reply ids. The default is 99.

z/OS MVS Initialization and Tuning Reference provides the range of values for RMAX.

Notes:

1. Set a value of 9999 for the RMAX parameter on the DEFAULT statement in the CONSOLxx parmlib member (if possible) for optimal performance.
2. When a sysplex is configured with a MAXSYSTEM value greater than 8, reply IDs are no longer assigned in strict sequential order. Instead, systems obtain groups of reply IDs for assignment to WTORs, and the ids might not be assigned in sequential order. This change requires no coding changes on the installation's part, but might surprise an operator. You should consider informing operators of this change.

RLIM and RMAX Values

The relationship between RLIM and RMAX values in your sysplex can help you plan for WTOR messages and operator replies. In a sysplex, the first system to join sets the RMAX value, which has sysplex scope. If you do not specify RLIM, the first system to join the sysplex sets RLIM to the value of RMAX.

In a sysplex running JES2, when XCFLOCAL is set, the sysplex runs without a couple data set and systems cannot join or use the services of the sysplex. In both these situations, the RLIM default of 10 is used, if no RLIM value is specified, regardless of what is set for RMAX.

Reply IDs and RMAX

The RMAX value determines the maximum number of reply IDs that an operator can use to respond to WTOR messages. Using the short form of the JES2 REPLY command, the operator can omit the comma, but the system might misinterpret the command depending on the RMAX value. For example, if RMAX is 99, and the operator enters the following:

```
103NONE
```

MVS interprets the command as follows:

```
R 10,3NONE
```

Using the JES3 form of the REPLY command, an operator must use a comma to separate the reply ID from the command text:

```
5,NONE
```

Controlling Automatic Ending of Multi-line WTO Messages

If a program issues a multi-line WTO message but does not end the message by issuing an endline, the target console might stop receiving message traffic. The system will detect this condition and end the message automatically.

To end a multi-line WTO message when it detects that no data line or endline has been issued for the message after an interval of 30 seconds, the system issues the following endline:

```
MESSAGE TIMED OUT - MESSAGE COMPLETION FORCED
```

The default interval is 30 seconds. You can control the length of the interval by using AMASPZAP or IGWSPZAP to set a value from X'0001' to X'FFFF' (1 second to 65,535 seconds). To update the time interval, run either AMASPZAP or IGWSPZAP with the following statements:

```
++USERMOD(MYMOD)          /* USERMOD name of your choice */  
++VER(Z038) FMID(HBB5510) /* FMID level of your system */  
++ZAP(IEECVUCM).
```

```
NAME IEANUC01 IEEUCMC
VER 00F8 001E          /* changing default of 001E (30) seconds */
REP 00F8 002D          /* to 002D (45) seconds */
```

Aggregating Messages Returned to the ROUTE Command

If an operator routes a command to more than one system, the command responses returned to the originating console can be very confusing if they are simply presented at the console in the order they are received. To help avoid the confusion, MVS collects the messages so they can be presented in a more readable format on the console. This is called an “aggregated response”. The messages that are aggregated are sorted in alphabetical order by system name.

If some messages arrive too late to be aggregated, MVS first displays the name(s) of the system(s) from which messages have not arrived in time, then displays the aggregated messages. Any messages that are not aggregated are displayed singly on the console, as they arrive.

By default, MVS waits as long as 30 seconds before displaying aggregated messages. However, MVS doesn’t always make the operator wait the maximum time. MVS displays the aggregated messages a short time after receiving at least one response from each system to which the command was routed.

By default, the maximum amount of time that MVS waits for messages before aggregating them is 30 seconds. You can change this maximum wait time as follows:

- Specify the ROUETIME parameter on the INIT statement in CONSOLxx. This affects the entire sysplex.
- Change the current ROUETIME value by entering the CONTROL M command. This affects the entire sysplex.
- Request a one-time routing time interval by entering the T= operand on the ROUTE command itself. This affects only the ROUTE command on which it is specified.

Command responses are aggregated if:

- The command responses are received within the timeout period.
- The command responses are identified with console IDs.

Note: If, when issuing a command response, a command processor does not use the console ID of the command issuer, MVS cannot return an aggregated command response to the ROUTE command issuer.

Command responses received after the timeout period are not aggregated. MVS attempts to send them back to the originator.

Note: If the current ROUETIME value is 0, or if T=0 is specified on the ROUTE command, no messages are aggregated; they are presented at the originating console as they are received.

Appearance of Aggregated Messages

The following examples illustrate how MVS aggregates command responses. In these examples, the command responses are returned to an out-of-line area on the console. The sysplex has three systems, named SYS1, SYS2, and SYS3.

Example 1: Comparison of Aggregated and non-Aggregated Messages

The following two panels use the D T command (DISPLAY TIME) to show how command responses are aggregated.

The following panel shows several uses of the D T command without aggregation of command responses:

```

- SYS1 d t
A SYS1 IEE136I LOCAL: TIME=09.59.49 DATE=1993.257 GMT: TIME=13.59.49
    DATE=1993.257
- SYS1 ro sys2,d t
B SYS2 IEE136I LOCAL: TIME=10.00.21 DATE=1993.257 GMT: TIME=14.00.21
    DATE=1993.257
- SYS1 ro t=0,*all,d t
C SYS1 IEE136I LOCAL: TIME=10.00.59 DATE=1993.257 GMT: TIME=14.00.59
    DATE=1993.257
    SYS3 IEE136I LOCAL: TIME=10.00.59 DATE=1993.257 GMT: TIME=14.00.59
    DATE=1993.257
    SYS2 IEE136I LOCAL: TIME=10.00.59 DATE=1993.257 GMT: TIME=14.00.59
    DATE=1993.257

IEE612I CN=MASTER1  DEVNUM=03E0 SYS=SYS1      CMDSYS=SYS1

IEE163I MODE= RD

```

- A** The D T command is issued and processed on SYS1.
- B** The D T command is issued on SYS1 and routed to SYS2 for processing.
- C** The D T command is issued on SYS1 and routed to all systems (SYS1, SYS2, and SYS3) for processing. To ensure that responses are not aggregated, T=0 is specified.

The following panel shows the difference between non-aggregated and aggregated command responses:

```

    SYS2 IEE136I LOCAL: TIME=10.00.21 DATE=1993.257 GMT: TIME=14.00.21
    DATE=1993.257
- SYS1 ro t=0,*all,d t
A SYS1 IEE136I LOCAL: TIME=10.00.59 DATE=1993.257 GMT: TIME=14.00.59
    DATE=1993.257
    SYS2 IEE136I LOCAL: TIME=10.00.59 DATE=1993.257 GMT: TIME=14.00.59
    DATE=1993.257
    SYS3 IEE136I LOCAL: TIME=10.00.59 DATE=1993.257 GMT: TIME=14.00.59
    DATE=1993.257
- SYS1 ro t=5,*all,d t
B
IEE421I RO *ALL,D T                FRAME LAST  F    E    SYS=SYS1
SYSNAME  RESPONSES -----
SYS1     IEE136I LOCAL: TIME=10.06.48 DATE=1993.257 GMT:
        TIME=14.06.48 DATE=1993.257
SYS2     IEE136I LOCAL: TIME=10.06.48 DATE=1993.257 GMT:
        TIME=14.06.48 DATE=1993.257
SYS3     IEE136I LOCAL: TIME=10.06.48 DATE=1993.257 GMT:
        TIME=14.06.48 DATE=1993.257

IEE612I CN=MASTER1  DEVNUM=03E0 SYS=SYS1      CMDSYS=SYS1

IEE163I MODE= RD

```

- A** The D T command is issued on SYS1 and routed to all systems (SYS1,

SYS2, and SYS3) for processing. With T=0 specified on the ROUTE command, responses to D T from the three systems are not aggregated.

- B** Again, the D T command is issued on SYS1 and routed to all systems (SYS1, SYS2, and SYS3) for processing. With T=5 specified on the ROUTE command, responses to D T from the three systems are aggregated. In this example, the aggregated messages are shown in **highlighted text**. Note how the responses in the T=5 response are formatted as compared to the T=0 response.

Example 2: Another Sample Aggregation of Command Responses

The following two panels use a very short timeout interval (T=1) to show how non-aggregated responses are handled.

```
A
- SYS1 ro t=1,*all,v 414,offline

B
IEE421I RO *ALL,V 414,OFFLINE          FRAME 1  F    E  SYS=SYS1
NO RESPONSE RECEIVED FROM THE FOLLOWING SYSTEM(S):
SYS2

IEE612I CN=MASTER1  DEVNUM=03E0 SYS=SYS1  CMDSYS=SYS1

IEE163I MODE= RD
```

- A** The ROUTE command is used to try to vary device 414 offline on all systems. A timeout interval of 1 second (T=1) is specified on the ROUTE command.
System SYS2 does not respond within one second. Therefore MVS cannot include the command response from SYS2 in the aggregated response.
- B** MVS lists the systems from which no response was received in time for aggregation. In this case, only SYS2 is listed, under NO RESPONSE RECEIVED FROM THE FOLLOWING SYSTEM(S):. This output is in FRAME 1 of message IEE421I.

```

- SYS1 ro t=1,*all,v 414,offline
C
- SYS2 IEF281I 0414 NOW OFFLINE

D
IEE421I RO *ALL,V 414,OFFLINE      FRAME LAST  F    E  SYS=SYS1
SYSNAME  RESPONSES -----
SYS1     IEF281I 0414 NOW OFFLINE
SYS3     IEE303I 0414    OFFLINE

IEE612I CN=MASTER1  DEVNUM=03E0 SYS=SYS1    CMDSYS=SYS1

IEE163I MODE= RD

```

- C** MVS displays the non-aggregated command response from SYS2. The time when the non-aggregated messages appear does not depend on when the operator scrolls to the second frame of message IEE421I.
- D** After the operator scrolls forward to the second (and last) frame of message IEE421I, MVS displays the aggregated messages.

Controlling Write-to-Log (WTL) Message Buffers

You can specify the number of buffers that the system uses to write messages to the hardcopy log. To specify the number of write-to-log (WTL) message buffers, use the following keyword on the INIT statement of CONSOLxx:

LOGLIM Defines the number of WTL buffers that the system uses.

Ensure that your installation has enough storage for the LOGLIM buffers. *z/OS MVS Initialization and Tuning Reference* provides the range of values for LOGLIM and provides examples.

Handling Translated Messages

The MVS message service (MMS) enables your installation to use message files for message translation. MMS substitutes a message translated into a different language for the U. S. English equivalent message. If MMS is active, authorized users of extended MCS consoles on TSO/E can select available languages for message translation and receive translated messages on their screens. Application programs can also use MMS to handle translation of messages. Depending on how the installation displays the messages, users can receive those translated messages wherever they are displayed or recorded.

TSO/E terminal users can also receive translated messages (including TSO/E messages) during a TSO/E session or from a batch job. In order to receive translated TSO/E messages, you must have TSO/E Version 2.2 installed on your system.

For MMS to handle translated messages, your installation must use the MVS message compiler to format install message files that contain English message skeletons and the translated language message skeletons.

IBM provides English and Japanese versions of MVS messages and English and Japanese versions of TSO/E messages. If you want languages other than Japanese, the installation must supply its own version of the translated message skeletons.

References

Applications can use macros for the MVS message translation services. For information on how applications handle message translation or how to create message skeletons for languages, see *z/OS MVS Programming: Assembler Services Guide*.

Steps for Providing Translated Messages

The following steps describe what your installation must do for users to receive translated messages.

1. Ensure that the appropriate system install message files have been installed on your system.
For MVS messages, IBM provides an install message file for U. S. English messages and an install message file for the Japanese translation. As a feature of TSO/E, IBM also provides an English and Japanese install message file for TSO/E messages. Each install message file for the language is a PDS. Your installation uses SMP/E to install each install message file on the system. You can install concatenated PDSs. For installation information, see the program directory for the product.
2. Allocate space for each run-time message file.
You use the MVS message compiler to format each install message file to a run-time message file. (The compiler formats one run-time message file for each language including English.) This run-time message file must be a VSAM linear data set. You must allocate a VSAM linear data set for each run-time message file. See “Allocating Storage for a Run-Time Message File” on page 130.
3. Use the MVS message compiler to format the install message file into a run-time message file.
The input to the compiler is the install message file PDS. The output from the compiler is the run-time message file (allocated in the previous step). See “Compiling Message Files” on page 130.
4. If needed, create installation exit routines.
IBM provides two exits that an installation can use for MMS processing. You specify the exit names in MMSLSTxx of SYS1.PARMLIB. See “Controlling MMS through Installation Exits” on page 132 and “Using Parmlib to Control Message Translation” on page 132.
5. Create or update the following Parmlib members to initialize values for MMS:
 - MMSLSTxx to define the available languages for message translation and other message translation processing
 - CNLcccxx to define the date and time formats for translated messages
 - CONSOLxx to specify the MMSLSTxx member in effect for the systemSee “Using Parmlib to Control Message Translation” on page 132.
6. Activate MMS.

You can activate, refresh, or stop MMS. You can use the INIT statement in CONSOLxx to activate MMS at initialization. The operator can activate or stop MMS by using the SET MMS command. See “Activating MMS” on page 133.

On TSO/E, the installation can indicate in the TSO/E LOGON exit a primary or secondary language for message translation. Otherwise, TSO/E users can specify the primary or secondary language on the TSO/E PROFILE command, and TSO/E can deliver the translated messages. See *z/OS TSO/E User's Guide* for information.

Allocating Storage for a Run-Time Message File

The install message file contains PDS members that include message skeletons for the language. (For the English PDS and Japanese PDS that IBM provides, each PDS member contains message skeletons for each MVS component.) The MVS message compiler converts the install message file into a run-time message file. The run-time message file for each language must be a VSAM linear data set.

To create the data set for the run-time message files, you need to specify the DEFINE CLUSTER function of access method services (IDCAMS) with the LINEAR parameter. When you code the SHAREOPTIONS parameter for DEFINE CLUSTER, use SHAREOPTIONS (1,3). For a complete explanation of SHAREOPTIONS, see *z/OS DFSMS: Using Data Sets*.

Figure 13 shows a sample job that invokes Access Method Services (IDCAMS) to create the linear data set named SYS1.ENURMF.DATA on the volume called MMSPK1. When IDCAMS creates the data set, it creates it as an empty data set. Note that there is no RECORDS parameter; linear data sets do not have records.

```
//DEFCLUS    JOB 'ALLOCATE LINEAR',MSGLEVEL=(2,0),
//          CLASS=R,MSGCLASS=D,USER=IBMUSER
//*
//*          ALLOCATE A VSAM LINEAR DATASET
//*
///*DCLUST   EXEC PGM=IDCAMS,REGION=4096K
//SYSPRINT  DD SYSOUT=*
//MMSPK1    DD UNIT=3380,VOL=SER=MMSPK1,DISP=OLD
//SYSIN     DD *
            DELETE (SYS1.ENURMF) CL PURGE
            DEFINE CLUSTER (NAME(SYS1.ENURMF) -
                           VOLUMES(MMSPK1) -
                           CYL(1 1) -
                           SHAREOPTIONS(1 3) -
                           LINEAR) -
            DATA (NAME(SYS1.ENURMF.DATA))
```

Figure 13. Sample JCL for Creating a Run-Time Message File

When you have allocated a VSAM linear data set for each run-time message file, you can run the message compiler to convert the install message file for messages in that language. (You must allocate one VSAM linear data set for each run-time message file.)

Compiling Message Files

The message compiler converts the message skeletons in an install message file into a run-time message file. The compiler expects a PDS or concatenated PDSs as input. The message compiler reads from the install message file and creates a run-time message file in the VSAM linear data set that you have allocated. If the compiler cannot process a message, it issues an error message. The message compiler also sets a return code.

You must run the message compiler:

- For each language install message file, including U. S. English
- Whenever you receive updates to the messages in the install message file

Invoking the Message Compiler: The message compiler is an executable program. You can use a batch job, a TSO/E CLIST, or a REXX EXEC to invoke the message compiler. The syntax to invoke the message compiler for each follows.

The lowercase variables used in the examples have the following meanings:

msg_pds

is the name of the install message file PDS containing all the message skeletons for a single language. *msg_pds* must be a partitioned data set.

msg_div_obj

specifies the name of the run-time message file that is to contain the compiled format of the message skeletons for the language. *msg_div_obj* must be a linear VSAM data set suitable for use as a data-in-virtual object.

lang,dbcs

specifies parameters. *lang* is the three character code of the messages contained in the install message file. *dbcs* indicates whether this language contains double-byte characters (y is yes, n is no).

Using JCL to Invoke the Message Compiler: To invoke the compiler as a batch job, code the following JCL:

```
/*  
/*      GENERATE DATA OBJECT FROM EXTRACTED MESSAGES  
/*  
//COMPILE EXEC PGM=CNLCCPLR,  
//          PARM=(lang,dbcs)  
//SYSUT1 DD DSN=msg_pds,DISP=SHR          /* THE INSTALL MESSAGE FILE */  
//SYSUT2 DD DSN=msg_div_obj,DISP=(OLD,KEEP,KEEP) /* THE VSAM DATA SET */  
//SYSPRINT DD SYSOUT=*
```

Using CLIST to Invoke the Message Compiler: To invoke the compiler as a CLIST, code the following statements:

```
PROC 0  
FREE DD(SYSUT1,SYSUT2,SYSPRINT)          /* FREE DD'S          */  
ALLOC DD(SYSUT1) DSN('msg_pds') SHR      /* ALLOC INPUT - INSTALL MESSAGE FILE */  
ALLOC DD(SYSUT2) DSN('msg_div_obj') OLD  /* ALLOC OUTPUT - VSAM DATA SET      */  
ALLOC DD(SYSPRINT) DSN(*)                /* ALLOC SYSPRINT          */  
CALL 'SYS1.LINKLIB(CNLCCPLR)' ('lang,dbcs') /* CALL MESSAGE COMPILER  */  
SET &RCODE = &LASTCC                    /* SET RETURN CODE        */  
FREE DD(SYSUT1,SYSUT2,SYSPRINT)          /* FREE FILES            */  
EXIT CODE(&RCODE)                        /* EXIT                    */
```

Using REXX to Invoke the Message Compiler: To invoke the compiler as a REXX exec, code the following statements:

```
/* MESSAGE COMPILER INVOCATION EXEC */
```

```
"FREE DD(SYSUT1,SYSUT2,SYSPRINT)"
```

```
"ALLOC DD(SYSUT1) DSN(msg_pds) SHR"
```

```
"ALLOC DD(SYSUT2) DSN(msg_div_obj) OLD"
```

```
"ALLOC DD(SYSPRINT) DSN(*)"
```

```
"CALL 'SYS1.LINKLIB(CNLCCPLR)' (lang,dbcs)"
```

```
compiler_rc=rc
```

```
"FREE DD(MSGIN,MSGOUT,SYSPRINT)"
return(compiler_rc)
```

Note: For the variables **msg_pds**, **msg_div_obj**, **lang**, and **dbcs**, REXX substitutes values that you have assigned. For information on using REXX, see *z/OS TSO/E REXX User's Guide*.

Example of Running the Message Compiler: Run a batch job to produce the run-time message file for the Japanese messages. In the example, the install message file is named **INSTALL.MSG.JAPAN**. The data set for the run-time message file is **SYS1.MSG.JAPAN** and has been previously defined. You can supply your own names.

```
/**
/**      Creating the run-time message file
/**
//COMPILE EXEC PGM=CNLCCPLR,PARM=('JPN,Y')
//SYSUT1  DD  DSN=INSTALL.MSG.JAPAN,DISP=SHR
//SYSUT2  DD  DSN=SYS1.MSG.JAPAN,DISP=OLD
//SYSPRINT DD  SYSOUT=*
/**
```

Message Compiler Return Codes: The message compiler generates a return code contained in register 15 and compiler error messages, both of which can be sent to SYSPRINT. The return codes are as follows:

Code	Meaning
0	Successful completion
4	Process complete. Run-time message file is complete but the compiler generated warnings.
8	Processing complete. The run-time message file is usable but incomplete.
12	Processing ended prematurely. The run-time message file is unusable.

Controlling MMS through Installation Exits

You can code two installation exits that the system invokes to tailor MMS processing. You specify the names of these exits in **MMSLSTxx**. See *z/OS MVS Installation Exits*.

Using Parmlib to Control Message Translation

To control information about the languages you have installed for translation, you must specify Parmlib members **MMSLSTxx** and **CNLcccxx**. To activate MMS, you use the **INIT** statement of **CONSOLxx**. (Operators can use the **SET MMS** command to affect the status of MMS.)

References

For the complete syntax of these **SYS1.PARMLIB** members, see *z/OS MVS Initialization and Tuning Reference*.

Using MMSLSTxx Statements: **MMSLSTxx** allows you to control information for each language on your system. It specifies the default language that the installation can use, the name of the installation exits, the name of the run-time message file, the name of the **SYS1.PARMLIB** member that controls the configuration of date and time formats, and an alternate name for the language, which is optional.

The following examples show how to use MMSLSTxx statements to specify two languages, U. S. English and Japanese. (Note that the number at the beginning of each statement is used for reference only; do not code it as part of the statement.)

Statement 1 specifies the language available for use by other MVS components and application programs. In the example, JPN is the language code for Japanese:

```
1  DEFAULTS  LANGCODE(JPN)
```

Statements 2 and 3 specify two installation exit routines to tailor MMS processing:

```
2  EXIT  NUMBER(1)  ROUTINE(NLSEXIT1)
3  EXIT  NUMBER(2)  ROUTINE(NLSEXIT2)
```

Statements 4 and 5 describe information for two languages installed on the system:

```
4  LANGUAGE  LANGCODE(JPN)  DSN(RUNTIME.VSAM.JAPAN)  CONFIG(CNLJPN01)
5  LANGUAGE  LANGCODE(ENU)  DSN(RUNTIME.VSAM.US)  CONFIG(CNLENU01)
      NAME(AMERICAN)  NAME(ENGLISH)
```

Statement 4 describes the language code for Japanese and names the run-time message file on the DSN parameter. It also specifies the CNLcccxx Parmlib member (CNLJPN01) that contains configuration data for the display of dates and times in Japanese messages.

Statement 5 describes the language code for U. S. English and names the run-time message file on the DSN parameter. It specifies the CNLcccxx Parmlib member (CNLENU01) for the display of dates and times in U. S. English. It also specifies two names for referencing the language. The first is the preferred name for the language (AMERICAN); the second is the alternate name (ENGLISH). TSO/E users can select the language using either name.

Using CNLcccxx: For each language that you define in MMSLSTxx, you must provide a CNLcccxx Parmlib member. CNLcccxx controls configuration data used to display dates and times for the translated messages of each language. In the member name, ccc is the three-character language code; xx uniquely identifies the member name. You specify the month, day, and date and time formats for the language. (If you want, you can specify defaults for date and time formats.)

Using the INIT Statement on CONSOLxx

Use the following keyword on the INIT statement of CONSOLxx to specify the MMSLSTxx member.

MMS Defines the MMSLSTxx member that contains information about languages available for translation.

Activating MMS

To activate MMSLST01, code the following on the INIT statement of CONSOLxx:

```
MMS(01)
```

If you specify MMS(NO), MMS is not active. After IPL, operators can issue the following command to activate MMS, where xx is the unique member name:

```
SET MMS=xx
```

To display information about MMS, operators can issue the following command:

```
DISPLAY MMS
```

The system displays information about MMS and the languages that are available for message translation.

Summary of MVS Message and Command Processing Services

Table 14 summarizes the message and command processing that MVS provides. It briefly describes the features of each service, indicates how the service is invoked, and gives the scope of the service in a sysplex environment:

Table 14. Summary of Message and Command Processing that MVS Provides

Service	Features	Where specified	Scope
CONSOLxx INIT	<ul style="list-style-type: none"> • Activate MPF • Activate AMRF • Specify WTO, WTOR, and WTL buffers • Activate MMS • Specify default timeout value for aggregating command responses from other systems in the sysplex 	Parmlib	Sysplex for activating AMRF and for aggregating command responses; system for other features.
MPF	<ul style="list-style-type: none"> • Suppress messages • Retain messages • Select messages for automation • Specify installation exits to process messages and commands 	MPFLSTxx	System
AMRF	<ul style="list-style-type: none"> • Retain action messages 	CONSOLxx INIT	Sysplex
IEAVMXIT	<ul style="list-style-type: none"> • Process messages • Override MPF processing • Control color, highlighting, and intensity of messages 	CONSOLxx INIT	System
Installation-defined message processing exits	<ul style="list-style-type: none"> • Process messages selected through MPFLSTxx • Control color, highlighting, and intensity of messages 	MPFLSTxx	System
Command Installation exits	<ul style="list-style-type: none"> • Process commands 	MPFLSTxx	System
MMS	<ul style="list-style-type: none"> • Process messages for translation 	CONSOLxx INIT	System

Chapter 4. Planning for Basic Operation Procedures

Once you have established your logical parmlib values to define your consoles and their use, you need to consider how your operators will interact with MVS at your installation.

The tasks of starting, running, and stopping releases involve controlling the MVS system software and most installation hardware, including processors, channel paths, I/O devices as well as the MCS consoles and extended MCS consoles that operators use to perform their tasks. In a multisystem environment, you need to decide how much control over the systems in a complex or sysplex you want your operators to have to meet your operations goals for the installation.

Reference

For migration concerns in a sysplex environment, see *z/OS MVS Migration*. It describes considerations about operating consoles in a sysplex for systems at different release levels of MVS.

While planning MVS operations, you or your operators need to understand how to develop procedures for daily operations and how to make those procedures work best for the installation. As operations planner, you and your operators must also be able to predict problems and set up procedures to handle them.

The tasks of operating a z/OS system that are described in this chapter include:

- Initializing the system
- Interacting with system functions
- Controlling shared DASD

Your installation can specify logical parmlib members that can affect how your operators handle these basic tasks. This chapter describes operator tasks from the point of view of MVS operations planning and what you can do to simplify how operators run MVS.

Other basic operator tasks include:

- Building, controlling, or rebuilding a global resource serialization ring or star complex. These tasks are described in *z/OS MVS Planning: Global Resource Serialization*.
- Responding to failing devices and reconfiguring system resources. These tasks are described in *z/OS MVS Recovery and Reconfiguration Guide*.
- Controlling the following system activities:
 - Controlling system status, device status, the availability of paths, or the system restart functions
 - Controlling time-sharing
 - Controlling jobs
 - Controlling system information recording for SMF, system trace, the generalized trace facility (GTF), or master trace.
 - Quiescing the system
 - Stopping the system

These tasks are described in *z/OS MVS System Commands*, which also describes the syntax for every MVS command and provides examples of commands.

Operators can activate dynamic I/O configuration for MVS using the Hardware Configuration Definition or the `ACTIVATE` command. For information, see *z/OS HCD Planning* and *z/OS MVS System Commands*.

Operators can use commands to control and display information about MVS and Advanced Program-to-Program Communication (APPC). APPC uses the Systems Network Architecture (SNA) LU 6.2 protocol and allows interconnected systems to communicate through applications that exchange data. The APPC/MVS environment is controlled through `SYS1.PARMLIB` members `APPCPMxx` and `ASCHPMxx`, and MVS commands `START`, `SET`, and `DISPLAY`. For information, see *z/OS MVS Planning: APPC/MVS Management*, *z/OS MVS System Commands*, and *z/OS MVS Initialization and Tuning Reference*.

Initializing the System

During initialization of an MVS system, the operator uses the system console or hardware management console, which is connected to the processor controller or support element. From the system console, the operator initializes the system control program during the nucleus initialization program (NIP) stage.

During the NIP stage, the system might prompt the operator to provide system parameters that control the operation of MVS. The system also issues informational messages that inform the operator about the stages of the initialization process.

The `LOADxx` parmlib member allows your installation to control the initialization process. For example, if you specify, in `LOADxx`, the `IEASYSxx` or `IEASYMxx` members that the system is to use, the system does not prompt the operator for system parameters that are specified in those members; it uses the values in those members instead.

For information about the placement of `LOADxx` at initialization, see *z/OS MVS Initialization and Tuning Reference*.

For specific information on initialization procedures and the system console, see the processor operator's guide.

The System Console and Message Processing

How you define the system console can determine the volume of messages that the system console receives during and after initialization.

During initialization you can control the volume of messages that the console receives. See "Specifying LOAD Information" on page 137. You can reply to all `WTOR` messages from the system console during initialization.

You can control how the system console receives messages after initialization by defining values in `CONSOLxx`. You define routing attributes for the system console in `CONSOLxx` that control message traffic when the operator places the console in problem determination mode. See "Messages that the System Console Receives in Problem Determination Mode" on page 140.

Using the System Console

Use the attached system console for initialization of MVS and for backup recovery purposes. For normal operation of the system, use MCS, SMCS consoles or extended MCS consoles, or subsystem consoles like NetView consoles. During abnormal situations when these consoles cannot operate, operators can use the system console to diagnose the console error and restore normal console operations. See “Problem Determination and the System Console” on page 139.

Specifying LOAD Information

From the system console or hardware management console, operators can specify the device number of the volume for the input/output definition file (IODF), select a LOADxx member, and control the display of messages and system prompts during initialization.

The operator can specify the following values to initialize the system control program:

- The device number of the volume where the IODF, a VSAM data set that manages system configuration data, resides. This is also the device on which the search for the LOADxx member of SYSn.IPLPARM or SYS1.PARMLIB begins. For information about IODF and SYSn.IPLPARM, see *z/OS MVS System Data Set Definition*.
- The LOADxx member of SYS1.PARMLIB or SYSn.IPLPARM (see *z/OS MVS Initialization and Tuning Reference* for a detailed description of LOADxx).
- The initialization message suppression indicator (IMSI) that controls the suppression of messages and system prompts during initialization.
- The alternate nucleus. (This specification overrides the value specified for the alternate nucleus in LOADxx.)

LOADxx allows you to specify I/O configuration data and information about the IODF data set, the nucleus, the master catalog, and the IEASYMxx and IEASYSxx parmlib members. For information about those parmlib members, see *z/OS MVS Initialization and Tuning Reference*. For information about the IODF data set, see *z/OS MVS System Data Set Definition*.

The IMSI character tells the system whether or not to do the following during system initialization:

- Display most informational messages.
- Prompt for system parameters.
- Prompt for the name of the master catalog.

See the section on loading the system software in *z/OS MVS System Commands* for a table that shows the possible values for the IMSI character. The values indicate all possible combinations of the actions listed above.

The NIP Console

The operator can complete the initialization process from the system console or hardware management console. The console acts as a NIP console. You may also define a device through HCD to act as a NIP console. During the NIP stage, an operator can continue to initialize the system from the NIP console

You can also define the same device that you use for the NIP console on a CONSOLE statement in CONSOLxx for the MCS master console. An SMCS console cannot be the NIP console. When MCS console services are active and

CONSOLxx values take effect, the NIP console can become the master console. See “Initializing the Master Console” on page 143.

If you define a NIP console for use during initialization, the system directs messages to the NIP console depending on the values that the operator specifies for IMSI.

Reference

For information about using HCD to define console devices, see *z/OS HCD User's Guide*.

The System Console and CONSOLxx

You can define the system console to MVS in CONSOLxx. You can define the system console on a CONSOLE statement by specifying SYSCONS for DEVNUM. You can specify CONSOLE keywords NAME, AUTH, ROUTCODE, LEVEL, UD, MONITOR, MSCOPE, and CMDSYS. The system ignores other CONSOLxx keywords. If you do not specify AUTH, the system console has master command authority.

RACF definitions for the system console may also be required. For more information about the system console and console security, see “Defining RACF Profiles” on page 55.

If you define message routing values for the system console in CONSOLxx, those values control message routing to the system console only when the operator activates problem determination mode. During normal operations, when problem determination mode is inactive, the system ignores these CONSOLxx routing values. For information about problem determination mode, see “Problem Determination and the System Console” on page 139.

During initialization, your operator can also specify CON=NONE in response to the system prompt for a CONSOLxx member. In that case, the system console assumes default CONSOLxx values and message routing depends on the IMSI values specified during initialization.

Naming the System Console

It is strongly recommended that you name the system console in CONSOLxx. You can specify a name for the system console using the NAME keyword. Select a unique name for the system console that cannot be confused with a valid device number. (For other console naming restrictions, see “Restrictions for Console Names” on page 42.)

If your operator specifies CON=NONE, or if you do not name the system console in CONSOLxx, MVS tries to use the name of the system to which the console is attached as the name of the system console. The system uses the system name defined on the IEASYSxx parameter SYSNAME as long as that name is unique and cannot be interpreted as a valid device number. If you do not name the system console in CONSOLxx, use a system name that cannot be confused with a device number that the system can use. For example, do not use a system console name like ABC, BAD, or C01.

If you specify a system name that the system can interpret as a valid device number, the system does not use SYSNAME as the name of the system console. If the system cannot use SYSNAME for the system console name, or if the system

console name is not unique, the name of the system console is SYSCNxxx, where xxx is a three-character suffix generated by the system.

The System Console During Normal Operations

During normal operations, when the system console is not in problem determination mode, it receives a minimal set of messages. Otherwise, the volume of messages received during normal operations might flood the system console and have an impact on operations.

When it is not in problem determination mode, the system console can receive the following kinds of messages:

- Synchronous messages that bypass regular MVS message queuing on MCS or other consoles. Synchronous messages can indicate system problems that require the operator to respond through the system console directly attached to the processor controller.
- Undelivered messages that MCS or extended MCS consoles could not receive. You can define an MCS or extended MCS console with the UD console attribute to receive all messages that the system cannot deliver. If no other console with the UD attribute is able to receive the message, the system directs the undelivered message to the system console.
- Messages that an operator directs to the system console by specifying the system console name.

During normal operations, an operator can reply to any WTOR message from the system console. However, the system console cannot receive messages defined by routing code or message level. Also, except for VARY CN,ACTIVATE, an operator cannot issue commands from the system console to change the system console characteristics.

Problem Determination and the System Console

For normal message traffic after initialization, operators use MCS consoles, extended MCS consoles, or subsystem consoles. During regular system operations, an operator does not generally use the system console to interact with MVS.

When hardware or software operation problems occur that might cause MCS, extended MCS, or subsystem consoles to fail, an operator can place the system console in problem determination mode. When the system console is in problem determination mode, the operator can:

- Enter commands and receive messages to help debug the system problem
- Control console attribute values for the system console.

The System Console in Problem Determination Mode

To respond to system problems when other consoles fail, an operator can enter VARY CN,ACTIVATE on the system console to activate problem determination mode. For diagnosis purposes, problem determination mode expands message and command processing for the system console. The operator restores the system console to normal operations by entering VARY CN,DEACTIVATE. For details on syntax and use of the command, see *z/OS MVS System Commands*.

Establishing Console Attributes for Problem Determination Mode in CONSOLxx

You can define system console attributes for problem determination mode in CONSOLxx. In CONSOLxx, you can define routing codes (ROUTCODE), message level (LEVEL), the undelivered message attribute (UD), and MONITOR attribute on the CONSOLE statement for the system console.

During regular operations (when the system console is not in problem determination mode), the system ignores CONSOLxx values for message routing to minimize message traffic. (See “The System Console During Normal Operations” on page 139.) When the operator activates problem determination mode for the first time after the IPL, the system uses the CONSOLxx values that you have defined to control problem determination mode for the system console.

If you do not define the system console in CONSOLxx, the system uses CONSOLxx default values to control problem determination mode for the system console. For information on console attribute default values, see *z/OS MVS Initialization and Tuning Reference*.

Changing Console Attributes through Commands

When the system console is in problem determination mode, the operator with the proper authorization can issue almost any MVS command. (Note that the operator cannot use the SWITCH command to switch system console attributes.) To alter the message routing values for the system console, the operator can issue VARY, CONTROL, or MONITOR commands. Making changes to system console attributes through commands allows the operator flexibility in controlling message processing for the system console during problem determination mode. For example, using the VARY command during problem determination mode, the operator can redefine routing codes for the system console without having to reIPL the system.

When the operator removes the system console from problem determination mode, the system stores the command changes to the console attributes. If the operator activates problem determination mode again from the system console during the same IPL, the system uses the console attribute changes it has stored instead of the values defined in CONSOLxx. See “Example of Controlling Problem Determination Mode for the System Console”.

Messages that the System Console Receives in Problem Determination Mode

When the system console is in problem determination mode, the system console receives all synchronous messages and can reply to all WTOR messages. In addition, the system console can receive the following messages:

- Messages identified by ROUTCODE or LEVEL either in CONSOLxx or by operator command. Note that if you use the default value for LEVEL, the system console in problem determination mode receives all messages except broadcast messages.
- Undelivered messages if the system console has the UD attribute. If no other console is able to display an undelivered message, the system console displays the message regardless of the UD attribute specification for the system console.
- Messages that an operator directs to the system console by specifying the system console name.

Example of Controlling Problem Determination Mode for the System Console

The following example shows how an operator can control problem determination mode for the system console. The example illustrates how the system handles console attribute definitions ROUTCODE and UD for the system console SYSCON1 defined in CONSOLxx as follows:

```
CONSOLE DEVNUM(SYSCONS) NAME(SYSCON1)
        ROUTCODE(1-5)
        UD(Y)
```

The operator initializes the system from the system console SYSCON1. After initialization, the CONSOLxx defaults are in effect for the system console. In this example, the following default values apply:

- ROUTCODE(NONE)
- UD(N)

Note: If no other console with UD(Y) is able to receive an undelivered message, the system console receives it. Otherwise, the system console does not receive an undelivered message during normal operations.

1. Normal operations

The operator receives a minimum set of messages on the system console and monitors normal system operations from an MCS console.

The MCS console fails on the system. The operator decides to enter VARY CN(syscon1),ACTIVATE on SYSCON1 to place the console in problem determination mode for the first time during this IPL.

2. Problem determination mode

The system console is now in problem determination mode. The system uses the values for ROUTCODE and UD defined in CONSOLxx:

- ROUTCODE(1-5)
- UD(Y)

Along with other messages it can receive, the system console receives messages defined by routing codes 1 through 5 and undelivered messages.

To receive more information about the problem, the operator decides to change the routing codes on the system console.

1. Problem determination mode

Without having to re-IPL, the operator issues the VARY command to change ROUTCODE to ALL. The system console can receive messages with all routing codes. The operator is able to restore the MCS console and continue normal operations. The operator enters VARY CN(SYSCON1),DEACTIVATE on the system console to deactivate problem determination mode.

2. Normal operations

The system console again receives a minimum set of messages. The CONSOLxx defaults for ROUTCODE and UD are in effect:

- ROUTCODE(NONE)
- UD(N)

The MCS console fails again on the system. The operator reissues VARY CN(SYSCON1),ACTIVATE on SYSCON1 for the second time during this ipl.

1. Problem determination mode

The system console is again in problem determination mode. In this example, the system uses the system console attribute for ROUTCODE based on when the operator last changed the routing code value. The UD value is based on CONSOLxx:

- ROUTCODE(ALL)
- UD(Y)

Along with other messages it can receive, the system console receives messages defined by all routing codes and undelivered messages.

The operator restores the MCS console and issues VARY CN(SYSCON1) DEACTIVATE on the system console to deactivate problem determination mode.

2. Normal operations

The operator continues normal operations from the MCS console.

Specifying the Time-of-Day Clock and the JES Subsystem

The system prompts the operator to set the date and time-of-day (TOD) clock and to start the job entry subsystem. You can

- Control if the operator needs to set the date and time by using CLOCKxx.
- Start JES automatically by using IEFSSNxx.

You can specify CLOCKxx and IEFSSNxx in IEASYSxx, and then specify the IEASYSxx member in LOADxx. Thus, depending on how you define values for your Parmlib members, the operator does not have to be prompted during initialization to set the clock or start JES. Using LOADxx is thus a good way to automate the initialization procedure for your system and simplify the process for your operators.

CLOCKxx and the Sysplex

CLOCKxx also allows you to specify that the system use an external time reference for sysplex operations. In a sysplex, each MVS system shares a clock that provides synchronized time stamps. This requirement allows the sysplex to monitor and sequence events across member systems. Systems that run on different processors in a sysplex require a Sysplex Timer to synchronize different TOD time stamps from the processors. Systems that run on a single processor in a sysplex (MVS systems running under VM as guest systems, or systems running in logical partitions in a PR/SM environment) can use the TOD clock in the processor to allow the sysplex to control timing events.

Plan the local time for CLOCKxx carefully. To maintain the integrity of time stamps within the sysplex, the standard time origin for the TOD clock must always be the same. Ensure that the TOD clock for each system in the sysplex is set to the same standard time origin. IBM strongly recommends the use of Greenwich Mean Time (GMT).

Consider those occasions when you want to adjust local time, such as the initiation of Daylight Savings Time in a system or sysplex. If you need to change the time zone, for example, you can change the time without resetting the TOD processor clocks. For a sysplex that uses the Sysplex Timer, you can adjust the time offset from the Sysplex Timer console, or use CLOCKxx and the SET CLOCK command to reflect the new time. For a sysplex that does not use the Sysplex Timer, you can use CLOCKxx and the SET CLOCK command. The changes that you make do not reset the TOD clock in the processor.

When you make adjustments to local time, IBM recommends that you do not reset the TOD clock on a processor in a sysplex. If you reset the TOD clock on a processor in a sysplex, the change affects sysplex timing.

References

For information about the LOAD parameter, see *z/OS MVS System Commands*. For information about LOADxx, IEASYSxx, CLOCKxx, and IEFSSNxx, see *z/OS MVS Initialization and Tuning Reference*.

For information on CLOCKxx, sysplex operations, and specifying local time changes, see *z/OS MVS Setting Up a Sysplex*.

Handling Wait States

When software errors occur during system initialization, the system enters a disabled wait state. To diagnose the problem, the operator must display the program status word (PSW) to determine the wait state code (the low-order 12 bits) and reason codes if any. *z/OS MVS System Codes* contains the operator responses to the wait state codes. The operator can follow the instructions for the specified wait state and reason codes. For how to display the PSW, see the operator's guide for the processor.

Initializing the Master Console

Once the system initializes the master console, it issues the message IEE612I to indicate that the master console is controlled by MCS.

The operator can enter the command DISPLAY C,K (or D C,K) to display a summary of the system CONTROL commands. Operators can use these commands to change the characteristics of the console.

Interacting with System Functions

To plan your installation's I/O operations so that operators can respond appropriately to mounting requests, device allocation, and I/O problems, you need to consider the following system functions:

- Device allocation
- Hot I/O detection
- Device boxing

Device Allocation

Device allocation is the assignment of input/output devices and volumes to job steps. Requests for device allocation come from data definition (DD) statements and dynamic device allocation requests.

The system accepts DD statements from:

- Job input to the JES reader
- Jobs submitted through the TSO SUBMIT command
- Started cataloged procedures
- The MOUNT command
- TSO/E LOGONs

Installation programs that run on the system can specify dynamic device allocation/unallocation requests.

To control the amount of work needed for device allocation, you might want to restrict device allocation requests. You can define default values for allocation

processing in ALLOCxx of the parmlib concatenation. ALLOCxx allows your installation to specify space, data set, and other allocation parameters for dynamic allocation requests. For more information about ALLOCxx, see *z/OS MVS Initialization and Tuning Reference*.

You can specify installation exits that get control whenever an allocation request occurs to perform further processing. In these exits, you can cancel the job that is making the request or satisfy the allocation request without having an operator perform actions like mounting volumes or varying devices on or offline. For more information about allocation exits, see *z/OS MVS Installation Exits*.

To control device allocation requests from DD statements, you might restrict each of the forms of input for these statements (for example, by holding the reader, or by setting a maximum LOGON count). However, because they originate within executing programs, you cannot control dynamic device allocation/unallocation requests.

While allocating devices, the system might ask operators to:

- Mount or dismount volumes
- Make decisions (for example, to bring a device online immediately or to wait)

Use VATLSTxx in the parmlib concatenation to control how to mount volumes for an installation. Based on the values you set in VATLSTxx, operators can issue MVS MOUNT and UNLOAD commands to mount or unload volumes efficiently. See “Specifying Shared DASD Mount Characteristics” on page 147 for a description of mount characteristics.

At IPL time or whenever a VARY command is issued, the system uses the VATLSTxx entries that you have specified. VATLSTxx helps reduce the amount of volume mounting so the system can process allocation requests for mounted devices quickly. Allocation processing is also faster when you define volumes as reserved rather than removable. For information on allocating devices in a multisystem that shares DASD, see “Controlling Shared DASD” on page 146. For more information using VATLSTxx, see *z/OS MVS Initialization and Tuning Reference*.

If a requested volume is not mounted, the system issues a mount message asking the operator to mount a specific volume or scratch volume. If the operator mounts the wrong volume, the system finds out as soon as it reads the volume label. The system unloads the volume and repeats the mount message.

If your system uses automatic volume recognition (AVR), operators can mount labeled volumes on unused drives not managed by JES3. The system recognizes these volumes and assigns the drives to later job steps as required.

Generally, to be allocated to job steps, devices must be online. Exceptions are (1) when the online test executive program (OLTEP) or a similar testing program is running and (2) when teleprocessing devices are allocated. Operators can bring offline devices online with the VARY command or in response to the allocation recovery message, IEF238D.

Operators can also specify that a pending offline device is eligible for allocation through their response to message IEF238D.

Considerations for Operators

Your operators should understand the need for enough work volumes to satisfy requests for temporary data sets at peak loads. A shortage of work volumes can cause the system to request additional scratch volumes so operators need to balance work volumes across channel paths to increase system efficiency.

Operators should not use the MOUNT command for devices managed by JES3. See *z/OS JES3 Commands*. They also should not mount a blank tape volume because the system scans the entire volume for a tape label and this scanning wastes time. If an unlabeled tape is needed, the operator can write a tapemark to avoid unnecessary scanning. After the operator mounts the tape volume and readies the drive, the system reads the volume label. If an incorrect volume is mounted, the system unloads the incorrect volume and repeats the mounting message.

Occasionally operators might receive two mount messages for the same volume, one starting with IEF and the other with IEC. They should treat the two messages as though they were one. The second is a reminder.

To refer to I/O devices in MVS commands, operators can use the unique device number assigned to each device (*devnum*).

In MVS commands, operators should not specify the symbolic names that programmers use in DD statements to group several devices for allocation to the job.

The IBM 3495 Tape Library Dataserver performs some operator actions such as mounts, demounts, and swaps. Operators might notice fewer messages associated with these actions. These messages are no longer sent to the console, but rather to the hardcopy log, where they are available for tracing and diagnosis.

Hot I/O Detection

Hot I/O refers to the repeated I/O interruptions that result from hardware malfunctions. Because hot I/O can cause the system to loop or to fill the system queue area with I/O control blocks, operators need to detect hot I/O quickly and correct the problem.

When the number of repeated interruptions exceeds an installation-defined threshold value, the system assumes there is a hot I/O condition. You can establish hot I/O recovery threshold values. If the threshold is reached, the system issues message IOS109I and attempts to recover from the hot I/O condition. The IECIOSxx parmlib member allows you to change threshold default values. See *z/OS MVS Initialization and Tuning Reference* for information on setting up hot I/O recovery defaults.

Considerations for Operators

Operators who must respond to hot I/O conditions should try to solve the problem at the lowest possible level; that is, they should try to correct the problem at the device first, and then the control unit. Operators can power the device off and on. If that does not help, they can reset the control unit if the affected device is not a direct access device. If these actions do not correct the problem, they might have to physically disconnect the device or control unit.

Whatever action operators take, they must respond to the prompting message or restartable wait state.

Device Boxing

In certain error recovery situations and in response to certain VARY and CONFIG commands, the MVS system can “box” an I/O device.

The system boxes a device:

- When it detects hot I/O on the device and the device cannot be recovered
- When, because of a channel path error, it takes the last path to the device offline
- When, because of a channel path error, it releases a reserve or assign on the device
- When it releases an unconditional reserve for the device
- When the operator issues a VARY OFFLINE command with the FORCE option for the device
- When the operator issues a CONFIG OFFLINE command with the FORCE operand for a channel path and the command releases a hardware reserve or removes the last path to the device

Once a device enters a boxed state, the system:

- Immediately terminates I/O in progress on the device
- Rejects future I/O requests (by a user or by the system) to the device as permanent I/O errors
- Rejects any attempts to allocate the device
- Puts the device in pending offline status

Note: For more information on device boxing, see *z/OS MVS Recovery and Reconfiguration Guide*.

Considerations for Operators

Because operators might release a reserve or assign on a device and cause a data integrity exposure, they should use the VARY OFFLINE and CONFIG OFFLINE commands with FORCE only in emergency situations.

When the boxing problem is fixed, operators can take the device out of the boxed state at any time by issuing VARY device ONLINE. Once the VARY command takes effect, the device is again available for I/O and allocations. Operators cannot take a boxed device out of the boxed state by replying with the device name to the allocation recovery message, IEF238D.

Controlling Shared DASD

The shared direct access storage device (DASD) option allows multiple systems to access common data on direct access storage devices. This sharing is accomplished through a hardware feature of the DASD control unit together with the reserve/release function of the operating system or through the global resource serialization function of the operating system. (For more information, see *z/OS MVS Planning: Global Resource Serialization*.)

During system installation, you can choose the shared DASD option. The advantages of using shared DASD include:

- Reducing the amount of time your operators have to spend moving volumes from one system to another.
- Minimizing the updating of data sets because operators have to update only one instead of two or more duplicates.

- Simplifying scheduling. Unless the job has other special requirements, you can run a job needing a specific data set on a shared device on any of the sharing systems.

Specifying Shared DASD Mount Characteristics

Shared DASD can affect the volume characteristics, device status, volume mounting, and unloading at your installation. You can define shared DASD in VATLSTxx as permanently resident on the system; volumes on the DASD can be shared but the DASD itself cannot be physically mounted on another system.

You can also define the DASD as removable; the DASD can be mounted on another system, but first any other system using the device must take the DASD offline. Finally, you can define DASD as reserved; operators can also reserve removable DASD by using the MOUNT command. This means that the DASD is reserved for use by the system and that the device is offline to other sharing systems.

You can control the mount characteristics for shared DASD in a system by using VATLSTxx. Use VATLSTxx to set initial values for the mount characteristics of shared DASD at your installation.

Your operators can use MOUNT, VARY, and CONFIG commands to reserve volumes for the system, take devices offline, and inform other sharing systems about the mounting of the volumes.

References

For information about VATLSTxx, see *z/OS MVS Initialization and Tuning Reference*. For information and examples on using MOUNT, VARY, and CONFIG see *z/OS MVS System Commands*.

Considerations for Operators

Before mounting a DASD volume to reserve it for the system, operators first must ensure that jobs requiring the volume are not selected by an initiator. Operators can hold up job selection by one of the following:

- Using the TYPRUN=HOLD parameter on the job statement.
- Using the appropriate subsystem command.
- Assigning the job to a job class and not activating that class for subsystem scheduling.

To reserve the volume, the operator then must:

1. Use the VARY command to put the device offline to each sharing system and wait for the offline message in each system. The device does not go offline until the message is issued. If no jobs are in progress, the offline message does not appear on the console. Operators can issue a START DEALLOC command to receive the message that the device is offline.
2. Use the MOUNT command to notify each sharing system of the unit where the new volume is being placed, and to put the volume in reserved status.
3. Use the MOUNT command to mount the volume.

After the volume is mounted, operators can use a JES command or activate the class for subsystem scheduling.

Notes:

1. To stop I/O to a shared device or group of devices, operators can use IOACTION QUIESCE. See *z/OS MVS System Commands* for syntax and examples.
2. If there is a hardware failure on a device other than the system residence device, the operators must vary the failing device offline on all sharing systems. Operators can then mount the shared volume on another shared device, if one is available, as long as parallel mount procedures occur on all sharing systems.
3. Operators can release a reserved device and remove a path to it by issuing CONFIG CHP,OFFLINE,FORCE. If operators try to remove a path to a reserved device with any other CONFIG command or with a VARY command, the system issues message IEE379I or IEE719I and does not execute the command.
4. When you want a shared non-JES3 device to be allocated by only one system, the operator of each system sharing the device should use the VARY command to place the device offline on the sharing systems.

IPLing a System that Shares DASD

Shared DASD can also affect how an operator IPLs a system that requires devices in use by other systems. An operator might have to re-IPL a system that is sharing DASD. If a device is being used by another system, the initializing system waits and then issues the following message to the operator:

```
* id IOS120A DEVICE ddd SHARED. REPLY 'CONT' or 'WAIT'
```

The operators should reply with “WAIT”.

“WAIT” causes the system to wait until the reserved device is released. If the system waits more than one minute, the operator should re-IPL.

Note: Operators must reply WAIT for 3344-emulated 3340 devices and 3350-emulated 3330 devices that are to be marked permanently resident by the volume attribute list (VATLST) facility.

If the device is still reserved, the system reissues message IOS120A. The operator should then reply with “CONT” and the path to the device is marked offline to the system. Thus, the device is also unavailable to jobs running on the system.

Chapter 5. Examples and MVS Planning Aids for Operations

This chapter provides some planning aids and reference information for MVS operations. It includes a summary of CONSOLxx statements and keywords, OPERPARM subkeywords for extended MCS consoles, and the MVS commands that operators can use to modify values. It also includes examples of using RACF to define and authorize a TSO/E user of an extended MCS console and how to control the console attributes associated with the user.

Finally, the chapter provides two examples for planning consoles in an MVS environment:

- Setting up an MCS console cluster for a single MVS system
- Setting up an MCS console configuration for a two-system sysplex

Summary of CONSOLxx and Commands to Change Values

The following tables summarize the CONSOLxx keywords and the operator commands that can change those keyword values. Table 15 describes the CONSOLE statement keywords, the OPERPARM equivalent, the MVS command to change the keyword value, the scope of the keyword, and meaning of the keyword.

Table 16 on page 151 describes the keywords INIT, HARDCOPY, and DEFAULT, the MVS command to change the keyword value, the scope of the keyword, and the meaning of the keyword.

“N/A” in a column indicates that no OPERPARM equivalent exists for the CONSOLE keyword. (There are no OPERPARM equivalents for keywords on INIT and DEFAULT statements.) “Must Re-IPL” in a column indicates that operators cannot change the keyword value through commands. *z/OS MVS System Commands* provides complete reference information and examples for using MVS commands.

Table 15. CONSOLE Statement Summary

CONSOLE statement keyword	OPERPARM equivalent for extended MCS consoles	Command to change keyword value	Scope	Meaning
CONSOLE DEVNUM	N/A	Must re-IPL	System	Identifies the 3-digit or 4-digit device number for the MCS console
CONSOLE UNIT	N/A	Must re-IPL	System	Defines the unit device for the MCS console
CONSOLE NAME	See Note 1	Must re-IPL	Sysplex	Defines the console name
CONSOLE ALTERNATE	N/A	VARY CN,ALTCONS	Sysplex	Defines the alternate for the MCS console
CONSOLE ALTGRP	OPERPARM ALTGRP	VARY CN,ALTGRP	Sysplex	Defines the alternate console group for the console
CONSOLE AUTH	OPERPARM AUTH	VARY CN,AUTH	Sysplex	Defines command groups or authority
CONSOLE USE	N/A	CONTROL V,USE	Sysplex	Defines the input/output capability of the console

Table 15. CONSOLE Statement Summary (continued)

CONSOLE statement keyword	OPERPARM equivalent for extended MCS consoles	Command to change keyword value	Scope	Meaning
CONSOLE DEL	N/A	CONTROL S,DEL	Sysplex	Specifies automatic message deletion
CONSOLE RNUM	N/A	CONTROL S,RNUM	Sysplex	Defines number of messages per screen rolls
CONSOLE RTME	N/A	CONTROL S,RTME	Sysplex	Defines interval of time between screen rolls
CONSOLE CON	N/A	CONTROL S,CON	Sysplex	Defines conversational or non-conversational message deletion
CONSOLE SEG	N/A	CONTROL S,SEG	Sysplex	Defines the number of lines to delete using CONTROL E,SEG
CONSOLE AREA	N/A	CONTROL A	Sysplex	Defines status display areas for a console
CONSOLE MSGRT	N/A	MSGRT	Sysplex	Routes output from the DISPLAY, MONITOR, and CONFIG commands
CONSOLE UTME	N/A	CONTROL T,UTME	Sysplex	Defines the time interval for updating dynamic status displays
CONSOLE MFORM	OPERPARM MFORM	CONTROL S,MFORM	Sysplex	Defines message formats for the console
CONSOLE MONITOR	OPERPARM MONITOR	MONITOR	Sysplex	Displays jobname, data set status, or TSO/E information
CONSOLE PFKTAB	N/A	CONTROL N,PFK	System	Defines the PFK table for the console
CONSOLE ROUTCODE	OPERPARM ROUTCODE	VARY CN,ROUT VARY CN,AROUT VARY CN,DROUT	Sysplex	Defines the routing codes for the console
CONSOLE LEVEL	OPERPARM LEVEL	CONTROL V,LEVEL	Sysplex	Defines message levels
CONSOLE UD	OPERPARM UD	VARY CN,UD	Sysplex	Specifies that the console receive undelivered messages
CONSOLE MSCOPE	OPERPARM MSCOPE	VARY CN,AMSCOPE VARY CN,DMSCOPE VARY CN,MSCOPE	Sysplex	Defines systems that direct messages to a console
CONSOLE CMDSYS	OPERPARM CMDSYS	CONTROL V,CMDSYS	Sysplex	Defines systems where commands on a console can be directed for processing
CONSOLE SYSTEM	N/A	VARY CN,ONLINE,SYSTEM	Sysplex	In a sysplex, specifies which system the installation expects the console to be initialized on.

Table 15. CONSOLE Statement Summary (continued)

CONSOLE statement keyword	OPERPARM equivalent for extended MCS consoles	Command to change keyword value	Scope	Meaning
CONSOLE LOGON	N/A	VARY CN,LOGON	Sysplex	Defines the LOGON attribute of this console.
CONSOLE LU	N/A	VARY CN,LU	Sysplex	Defines the predefined LU for an SMCS console only.

Note 1: For the name of the extended MCS console, the system uses the TSO/E userid defined by RACF and under which the OPERPARM segment is stored.

Table 16. Summary of INIT, HARDCOPY, and DEFAULT Statements

INIT, HARDCOPY, and DEFAULT statement keywords	Command to change keyword value	Scope	Meaning
INIT APPLID	CONTROL M,APPLID	System	Sets the APPLID used by SMCS on this system
INIT GENERIC	CONTROL M,GENERIC	Sysplex	Sets the GENERIC used by SMCS for the entire sysplex
INIT CNGRP	SET CNGRP	Sysplex	Activates the member of CNGRPxx that defines console groups for the system or sysplex
INIT NOCCGRP	See Note 2 on page 152.	System	Defines the alternate console group from which the system can select a master console when no consoles are available
INIT MONITOR	MONITOR	System	Displays mount message information
INIT PFK	SET PFK	System	Activates the PFKTABxx member for MCS consoles
INIT CMDDEL	Must re-IPL	System	Defines the command delimiter for entering multiple messages on MCS consoles
INIT MPF	SET MPF	System	Activates the message processing facility
INIT AMRF	CONTROL M,AMRF	Sysplex	Activates the action message retention facility
INIT UEXIT	CONTROL M,UEXIT	System	Activates message processing exit IEAVMXIT
INIT MLIM	CONTROL M,MLIM	System	Specifies buffers for WTO messages
INIT RLIM	CONTROL M,RLIM	Sysplex	Specifies buffers for WTOR messages
INIT LOGLIM	CONTROL M,LOGLIM	System	Specifies buffers for messages that the system writes to the hardcopy log
INIT MMS	SET MMS	System	Activates the MVS message translation service
INIT ROUTTIME	CONTROL M,ROUTTIME	Sysplex	In a sysplex, specifies the maximum amount of time MVS waits before aggregating responses to commands routed to other systems.

Table 16. Summary of INIT, HARDCOPY, and DEFAULT Statements (continued)

INIT, HARDCOPY, and DEFAULT statement keywords	Command to change keyword value	Scope	Meaning
DEFAULT SYNCHDEST	See Note 2.	System	Specifies the alternate console group from which the system can select a console to display synchronous messages
DEFAULT LOGON	Must re-IPL	System	Specifies operator LOGON to MCS Consoles
DEFAULT HOLDMODE	Must re-IPL	System	Specifies that the operator can freeze the display on MCS console screens
DEFAULT ROUTCODE	Must re-IPL	System	Assigns routing codes for messages without a target console
DEFAULT RMAX	K M,RMAX See Note 1.	Sysplex	Specifies maximum number of WTOR reply ids
HARDCOPY DEVNUM	VARY devnum,HARDCPY	System	Defines a device as the hardcopy log
HARDCOPY ROUTCODE	VARY ,HARDCPY,AROUT VARY ,HARDCPY,ROUT VARY ,HARDCPY,DROUT	System	Defines route codes for the hardcopy log
HARDCOPY CMDLEVEL	VARY ,HARDCPY,NOCMDS VARY ,HARDCPY,INCMDS VARY ,HARDCPY,STCMDS VARY ,HARDCPY,CMDS	System	Defines command recording options for the hardcopy log. See Note 3.
HARDCOPY UD	VARY ,HARDCPY,UD	System	Defines the hardcopy log device with the UD attribute to record undelivered messages
HARDCOPY HCPYGRP	See Note 2.	System	Specifies the alternate console group from which the system can select an alternate console device as hardcopy log
HARDCOPY HCFORMAT	HCFORMAT	System	Defines 4-digit year format for hardcopy records

Notes:

1. You can increase RMAX without a re-IPL in most cases.
2. You can activate another CNGRPxx member (SET CNGRP) that defines the same console group but with different console members.
3. HARDCOPY CMDLEVEL controls logging of responses to commands directed to MCS consoles. For extended MCS consoles, OPERPARM LOGCMDRESP controls the logging of command responses. LOGCMDRESP(SYSTEM) indicates that the value for HARDCOPY CMDLEVEL in effect for the system is in effect for the extended MCS console.

Controlling Extended MCS Consoles Using RACF

The following examples show how to use RACF commands to define user profiles for an extended MCS console user.

Defining the User Profile of an Extended MCS Console

The security administrator can define a RACF user profile to control the console attributes of the extended MCS console user.

The following example shows how to define a RACF profile for new TSO/E user TAPE1:

```
ADDUSER TAPE1 OPERPARM(ROUTCODE(46) AUTH(SYS) MFORM(S) ALTGRP(TAPEGR))
```

This example defines the userid TAPE1 as an extended MCS console with console attributes defined by the OPERPARM keyword. (Note that the example includes only the information about console attributes for TAPE1. For complete information on the RACF ADDUSER command, see *z/OS Security Server RACF Command Language Reference*.

When TAPE1 is active, TAPE1 receives messages with routing code 46, has a command authority of SYS, and receives messages prefixed with the name of the system that issues the messages. The console group TAPEGR defined in CNGRPxx contains consoles to use as an alternate for TAPE1.

For application programs, you can define console attributes for TAPE1 through the MCSOPER macro instead of through RACF. The console attributes specified on MCSOPER override the RACF values specified through RACF OPERPARM. See *z/OS MVS Programming: Authorized Assembler Services Guide*.

Granting the User Access to the RACF OPERCMDS class

Ensure that the user of the extended MCS console has READ access to a profile in the RACF OPERCMDS class named:

```
MVS.MCSOPER.console-name
```

For a TSO/E user, console-name is the TSO/E userid that issues the TSO/E CONSOLE command. For an application program, console-name is the name specified on the MCSOPER macro.

Before the RACF administrator can grant a RACF user (TSO/E user or MCSOPER name) access to the RACF OPERCMDS class, the administrator must ensure that the user has a RACF user profile. In the following example, assume that the TSO/E user or application program name has a RACF user profile already defined.

The RACF security administrator can take the following steps to give users access to the RACF OPERCMDS class:

1. Issue the SETROPTS command to activate the OPERCMDS class:

```
SETROPTS CLASSACT(OPERCMD)
```
2. Issue the SETROPTS command to activate generic profiles for the class:

```
SETROPTS GENERIC(OPERCMD)
```
3. Issue RDEFINE to establish a profile for MVS.MCSOPER.*:

```
RDEFINE OPERCMDS MVS.MCSOPER.* UACC(NONE)
```
4. Grant the TSO/E user or application program access to the APPL class (in this example user TAPE1):

```
PERMIT MVS.MCSOPER.* CLASS(OPERCMD) ID(TAPE1) ACCESS(READ)
```

TAPE1 must have a RACF user profile defined. See “Defining the User Profile of an Extended MCS Console”.

5. Issue SETROPTS RACLIST command to refresh the OPERCMDS reserve class:

```
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Allowing a TSO/E User to Issue the CONSOLE Command

The following steps allow TSO/E user TAPE1 to issue the TSO/E CONSOLE command to activate the extended MCS console. In the example, assume that TAPE1 has a RACF user profile already defined:

1. Issue SETROPTS to activate the TSOAUTH resource class:

```
SETROPTS CLASSACT(TSOAUTH)
```

2. Issue RDEFINE to define the command CONSOLE in the resource class TSOAUTH with a universal access authority (UACC) of NONE:

```
RDEFINE TSOAUTH CONSOLE UACC(NONE)
```

This command creates a profile in the RACF TSOAUTH class for the TSO/E CONSOLE command.

3. Issue RACF PERMIT to authorize TAPE1 to use the CONSOLE command:

```
PERMIT CONSOLE CLASS(TSOAUTH) ID(TAPE1) ACCESS(READ)
```

To limit from which TSO/E terminal TAPE1 can initiate an extended MCS console session, the security administrator can specify the following:

```
PERMIT CONSOLE CLASS(TSOAUTH) ID(TAPE1) ACCESS(READ)  
WHEN(TERMINAL(terminal-id))
```

In this example, user TAPE1 can enter the TSO/E CONSOLE command only from the terminal specified by terminal-id.

4. To refresh the TSOAUTH resource class using SETROPTS RACLIST, issue the following:

```
SETROPTS RACLIST(TSOAUTH) REFRESH
```

Changing Console Attributes Using RACF

To change the console attributes, the RACF security Administrator can use RACF ALTUSER:

```
ALTUSER TAPE1 OPERPARM(ROUTCODE(ALL))
```

This example changes the console routing code for TAPE1 to ROUTCODE(ALL). Other console attributes defined on the ADDUSER command remain the same. Note that the ADDUSER command does not affect console attributes specified on the MCSOPER macro.

Reference

For information about RACF, see *z/OS Security Server RACF Security Administrator's Guide*.

Using RACF to Control APF Lists

RACF allows you to control the use of the MVS commands SETPROG and SET PROG, and the use of the CSVAPF macro, for processing authorized program facility (APF) lists.

The SETPROG APF command allows a user to add and delete entries in the authorized program facility (APF) list, or to change the format of the APF list. SET PROG allows a user to activate the PROGxx member of SYS1.PARMLIB that contains definitions for controlling the format and contents of the list of APF-authorized libraries. CSVAPF is an authorized MVS macro that allows you to perform the same APF list processing from an application program.

Note: For information on using CSVAPF, including authorization required with RACF, see *z/OS MVS Programming: Authorized Assembler Services Guide*. For information on using PROGxx, see *z/OS MVS Initialization and Tuning Reference*.

Command Authorization

An operator can issue the SETPROG or SET PROG command from a console with AUTH(SYS) or higher. If RACF authorization checking is in effect, you can control the use of these commands through RACF profiles. RACF authorization checking overrides the CONSOLxx AUTH specification.

To use RACF authorization checking to control any MVS command, the security administrator must ensure that each userid that issues the command is defined to RACF. Operators with a userid and a RACF profile can log on to a console, or the installation can define a RACF userid for the console itself. (For information, see “Using RACF to Control Command Authority and Operator Logon” on page 54 and “Defining RACF Profiles” on page 55.)

Defining Command Profiles

To define the resource profile for SETPROG, the RACF administrator can take the following steps:

1. To create a profile for the SETPROG command, issue RDEFINE:
RDEFINE OPERCMDS MVS.SETPROG UACC(NONE)
2. To permit the userid for the user logging on to the console (in this example user OPER1) to use the command in the OPERCMDS class, issue the following:
PERMIT MVS.SETPROG CLASS(OPERCMDS) ID(OPER1) ACCESS(UPDATE)

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the OPERCMDS class is not already active, issue the SETROPTS command as follows:
SETROPTS CLASSACT(OPERCMDS)

(To ensure that the OPERCMDS class is active, you can issue the SETROPTS LIST command.)

4. To refresh the OPERCMDS resource class, issue SETROPTS RACLIST:
SETROPTS RACLIST(OPERCMDS) REFRESH

For the SET PROG command, you follow the same steps as outlined for SETPROG but use the following RACF profile name:

```
MVS.SET.PROG
```

When you have given access to users of SETPROG and SET PROG, you can further control the use of the command.

Controlling How to Add or Delete APF List Entries for a Library

To control who can add or delete APF list entries for a library name, the RACF security administrator can take the following steps:

1. To establish a profile for the library name for the FACILITY class, issue RDEFINE:

```
RDEFINE FACILITY CSVAPF.libname UACC(NONE)
```

where libname is the fully qualified data set name of the library (without quotation marks). For example,

```
CSVAPF.SYS1.SUPER.UTILS
```

The length of the RACF profile including qualifiers should not exceed 39 characters. Otherwise, if the length of the library name is greater than 32 characters, RACF truncates the profile to 39 characters.

You can use generic characters for the qualifiers in the library name. For example,

```
CSVAPF.*.SUPER.UTILS
```

If you have RACF 1.9 or higher installed, you can use the following generic to cover all APF library names:

```
CSVAPF.**
```

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

```
SETROPTS GENERIC(FACILITY)
```

For complete coverage of APF-authorized library names, check the names currently specified in the IEAAPFxx or PROGxx SYS1.PARMLIB members.

2. To permit the user (in this example user OPER1) to add or delete the library name, issue the following:

```
PERMIT CSVAPF.libname CLASS(FACILITY) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

If any library name is not covered by a RACF profile and a user has access to the SETPROG or SET PROG command, MVS accepts the command. To ensure that only authorized users can perform the operation, you might define a generic

profile for all library names (CSVAPF.***) with UACC(NONE), then define specific RACF profiles for each set of libraries that the user has authorization to control.

Controlling How to Change the APF List Format

To control who can make the APF list dynamic, the RACF security administrator can take the following steps:

1. To establish a profile for the following command name to the FACILITY class, issue RDEFINE:

```
RDEFINE FACILITY CSVAPF.MVS.SETPROG.FORMAT.DYNAMIC UACC(NONE)
```

2. To permit the user (in this example user OPER1) to use the command in the class, issue the following:

```
PERMIT CSVAPF.MVS.SETPROG.FORMAT.DYNAMIC CLASS(FACILITY) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

To control who can to make the APF list static, the RACF security administrator can take the following steps:

1. Issue RDEFINE to establish a profile for the following command name for the FACILITY class:

```
RDEFINE FACILITY CSVAPF.MVS.SETPROG.FORMAT.STATIC UACC(NONE)
```

2. To permit the user (in this example user OPER1) to use the command in the class, issue the following:

```
PERMIT CSVAPF.MVS.SETPROG.FORMAT.STATIC CLASS(FACILITY) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Using RACF to Control Dynamic Exits

RACF allows you to control the use of the MVS commands SETPROG and SET PROG, and the use of the CSVDYNEX macro, for processing dynamic exits.

The SETPROG command allows a user to add and delete routines associated with a dynamic exit, to change the state of an exit routine, to undefine an implicitly-defined exit, or to change the attributes of an exit. SET PROG allows a user to activate the PROGxx member of SYS1.PARMLIB that contains definitions for controlling dynamic exits. CSVDYNEX is an authorized MVS macro that allows you to perform the same dynamic exit processing from an application program, along with defining a dynamic exit, calling the exit routines associated with a dynamic exit, providing recovery for an exit call, and obtaining a list of the dynamic exits.

Note: For information on using CSVDYNEX, including authorization required with RACF, see *z/OS MVS Programming: Authorized Assembler Services Guide*. For information on using PROGxx, see *z/OS MVS Initialization and Tuning Reference*.

Command Authorization

An operator can issue the SETPROG or SET PROG command from a console with AUTH(SYS) or higher. If RACF authorization checking is in effect, you can control the use of these commands through RACF profiles. RACF authorization checking overrides the CONSOLxx AUTH specification.

To use RACF authorization checking to control any MVS command, the security administrator must ensure that each userid that issues the command is defined to RACF. Operators with a userid and a RACF profile can log on to a console, or the installation can define a RACF userid for the console itself. (For information, see “Using RACF to Control Command Authority and Operator Logon” on page 54 and “Defining RACF Profiles” on page 55.)

Defining Command Profiles

To define the resource profile for SETPROG, the RACF administrator can take the following steps:

1. To create a profile for the SETPROG command, issue RDEFINE:

```
RDEFINE OPERCMDS MVS.SETPROG UACC(NONE)
```
2. To permit the userid for the user logging on to the console (in this example user OPER1) to use the command in the OPERCMDS class, issue the following:

```
PERMIT MVS.SETPROG CLASS(OPERCMDS) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the OPERCMDS class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(OPERCMDS)
```

(To ensure that the OPERCMDS class is active, you can issue the SETROPTS LIST command.)

4. To refresh the OPERCMDS resource class, issue SETROPTS RACLIST:
SETROPTS RACLIST(OPERCMDS) REFRESH

For the SET PROG command, you follow the same steps as outlined for SETPROG but use the following RACF profile name:

```
MVS.SET.PROG
```

When you have given access to users of SETPROG and SET PROG, you can further control the use of the command.

Controlling Defining a Dynamic Exit

To control who can define a dynamic exit via the REQUEST=DEFINE option of the CSVDYNEX macro, the RACF security administrator can take the following steps:

1. To establish a profile for the exit name for the FACILITY class, issue RDEFINE:
RDEFINE FACILITY CSVDYNEX.exitname.DEFINE UACC(NONE)

where `exitname` is the name of the dynamic exit. For example,
MYEXIT

You can use generic characters for the qualifiers in the exit name. For example,
CSVDYNEX.MYEX*

If you have RACF 1.9 or higher installed, you can use the following generic to cover all dynamic exit names:

```
CSVDYNEX.**
```

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

```
SETROPTS GENERIC(FACILITY)
```

For coverage of exit names, check the names currently specified in the PROGxx parmlib members. Also use the DISPLAY PROG,EXIT system command.

2. To permit the user (in this example user USER1) to use the REQUEST=DEFINE option of the CSVDYNEX macro for exit e, issue the following:

```
PERMIT CSVDYNEX.e.DEFINE CLASS(FACILITY) ID(USER1) ACCESS(UPDATE)
```

USER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:
SETROPTS RACLIST(FACILITY) REFRESH

Controlling Adding, Modifying or Deleting Exit Routines

To control who can add an exit routine to a dynamic exit, or modify or delete an exit routine associated with a dynamic exit, the RACF security administrator can take the following steps:

1. To establish a profile for the exit name for the FACILITY class, issue RDEFINE:
RDEFINE FACILITY CSVDYNEX.exitname.modname UACC(NONE)

where exitname is the name of the dynamic exit. For example,
SYS1.IEFACTRT

modname is the name of the exit routine. For example,
MYACTRT

You can use generic characters for the qualifiers in the exit name or routine name. For example,
CSVDYNEX.SYS1.IEF*

If you have RACF 1.9 or higher installed, you can use the following generic to cover all dynamic exit names:

CSVDYNEX.**

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

SETROPTS GENERIC(FACILITY)

For coverage of exit names, check the names currently specified in the PROGxx parmlib members. Also use the DISPLAY PROG,EXIT system command.

2. To permit the user (in this example user OPER1) to add or delete the routine name r to exit e, issue the following:

```
PERMIT CSVDYNEX.e.r CLASS(FACILITY) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

If any exit or exit routine is not covered by a RACF profile and a user has access to the SETPROG or SET PROG command, MVS accepts the command. To ensure that only authorized users can perform the operation, you might define a generic profile for all exit names (CSVDYNEX.***) with UACC(NONE), then define specific RACF profiles for each exit or exit routine that the user has authorization to control.

Controlling How to Undefine a Dynamic Exit

To control who can undefine a dynamic exit, the RACF security administrator can take the following steps:

1. To establish a profile for the exit name for the FACILITY class, issue RDEFINE:
RDEFINE FACILITY CSVDYNEX.exitname.UNDEFINE UACC(NONE)

where exitname is the name of the dynamic exit. For example,
MYEXIT

You can use generic characters for the qualifiers in the exit name or routine name. For example,
CSVDYNEX.MYEX*

If you have RACF 1.9 or higher installed, you can use the following generic to cover all dynamic exit names:

```
CSVDYNEX.**
```

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

```
SETROPTS GENERIC(FACILITY)
```

For coverage of exit names, check the names currently specified in the PROGxx parmlib members. Also use the DISPLAY PROG,EXIT system command.

2. To permit the user (in this example user OPER1) to undefine exit e, issue the following:

```
PERMIT CSVDYNEX.e.UNDEFINE CLASS(FACILITY) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

If any exit or exit routine is not covered by a RACF profile and a user has access to the SETPROG or SET PROG command, MVS accepts the command. To ensure that only authorized users can perform the operation, you might define a generic profile for all exit names (CSVDYNEX.***) with UACC(NONE), then define specific RACF profiles for each exit or exit routine that the user has authorization to control.

Controlling How to Obtain a List of the Dynamic Exits

To control who can obtain a list of the dynamic exits via the REQUEST=LIST option of the CSVDYNEX macro, the RACF security administrator can take the following steps:

1. To establish a profile for the exit name for the FACILITY class, issue RDEFINE:
RDEFINE FACILITY CSVDYNEX.LIST UACC(NONE)

2. To permit the user (in this example user USER1) to use the REQUEST=LIST option of the CSVDYNEX macro for exit e, issue the following:

```
PERMIT CSVDYNEX.LIST CLASS(FACILITY) ID(USER1) ACCESS(READ)
```

USER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Controlling Calling of a Dynamic Exit’s Routines

To control who can call a dynamic exit’s routines via the REQUEST=CALL option of the CSVDYNEX macro, the RACF security administrator can take the following steps:

1. To establish a profile for the exit name for the FACILITY class, issue RDEFINE:
RDEFINE FACILITY CSVDYNEX.exitname.CALL UACC(NONE)

where `exitname` is the name of the dynamic exit. For example,
MYEXIT

You can use generic characters for the qualifiers in the exit name. For example,
CSVDYNEX.MYEX*

If you have RACF 1.9 or higher installed, you can use the following generic to cover all dynamic exit names:

```
CSVDYNEX.**
```

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

```
SETROPTS GENERIC(FACILITY)
```

For coverage of exit names, check the names currently specified in the PROGxx parmlib members. Also use the DISPLAY PROG,EXIT system command.

2. To permit the user (in this example user USER1) to use the REQUEST=CALL option of the CSVDYNEX macro for exit e, issue the following:

```
PERMIT CSVDYNEX.e.CALL CLASS(FACILITY) ID(USER1) ACCESS(UPDATE)
```

USER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Controlling Recovering of Dynamic Exit Processing

To control who can use the REQUEST=RECOVER option of the CSVDYNEX macro to have the system complete its recovery processing of a prior use of CSVDYNEX REQUEST=CALL, the RACF security administrator can take the following steps:

1. To establish a profile for the exit name for the FACILITY class, issue RDEFINE:

```
RDEFINE FACILITY CSVDYNEX.exitname.RECOVER UACC(NONE)
```

where `exitname` is the name of the dynamic exit. For example,
MYEXIT

You can use generic characters for the qualifiers in the exit name. For example,
CSVDYNEX.MYEX*

If you have RACF 1.9 or higher installed, you can use the following generic to cover all dynamic exit names:

```
CSVDYNEX.**
```

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

```
SETROPTS GENERIC(FACILITY)
```

For coverage of exit names, check the names currently specified in the PROGxx parmlib members. Also use the DISPLAY PROG,EXIT system command.

2. To permit the user (in this example user USER1) to use the REQUEST=RECOVER option of the CSVDYNEX macro for exit e, issue the following:

```
PERMIT CSVDYNEX.e.RECOVER CLASS(FACILITY) ID(USER1) ACCESS(UPDATE)
```

USER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Using RACF to Control LNKLST Concatenations

RACF allows you to control the use of the MVS commands SETPROG and SET PROG, and the use of the CSVDYNL macro, for processing LNKLSTs.

The SETPROG command allows a user to update LNKLST concatenations, by defining a LNKLST set, adding data sets to or deleting data sets from a LNKLST set, removing the definition of a LNKLST set from the system, testing for the location of a specific module in the LNKLST concatenation, activating a LNKLST set, and updating a job to use the current LNKLST set. SET PROG allows a user to activate the PROGxx member of SYS1.PARMLIB that contains definitions for controlling LNKLSTs. CSVDYNL is an authorized MVS macro that allows you to perform this LNKLST processing from an application program.

Note: For information on using CSVDYNL, including authorization required with RACF, see *z/OS MVS Programming: Authorized Assembler Services Guide*. For information on using PROGxx, see *z/OS MVS Initialization and Tuning Reference*.

Command Authorization

An operator can issue the SETPROG or SET PROG command from a console with AUTH(SYS) or higher. If RACF authorization checking is in effect, you can control the use of these commands through RACF profiles. RACF authorization checking overrides the CONSOLxx AUTH specification.

To use RACF authorization checking to control any MVS command, the security administrator must ensure that each userid that issues the command is defined to RACF. Operators with a userid and a RACF profile can log on to a console, or the installation can define a RACF userid for the console itself. (For information, see “Using RACF to Control Command Authority and Operator Logon” on page 54 and “Defining RACF Profiles” on page 55.)

Defining Command Profiles

To define the resource profile for SETPROG, the RACF administrator can take the following steps:

1. To create a profile for the SETPROG command, issue RDEFINE:

```
RDEFINE OPERCMDS MVS.SETPROG UACC(NONE)
```
2. To permit the userid for the user logging on to the console (in this example user OPER1) to use the command in the OPERCMDS class, issue the following:

```
PERMIT MVS.SETPROG CLASS(OPERCMDS) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the OPERCMDS class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(OPERCMDS)
```

(To ensure that the OPERCMDS class is active, you can issue the SETROPTS LIST command.)

4. To refresh the OPERCMDS resource class, issue SETROPTS RACLIST:

```
SETOPTS RACLIST(OPERCMS) REFRESH
```

For the SET PROG command, you follow the same steps as outlined for SETPROG but use the following RACF profile name:

```
MVS.SET.PROG
```

When you have given access to users of SETPROG and SET PROG, you can further control the use of the command.

Controlling Defining a LNKLST Set

To control who can define a LNKLST set, the RACF security administrator can take the following steps:

1. To establish a profile for the LNKLST set name for the FACILITY class, issue RDEFINE:

```
RDEFINE FACILITY CSVSYNL.lnk1stname.DEFINE UACC(NONE)
```

where lnk1stname is the name of the LNKLST set. For example,
MYLNKLST.SET

You can use generic characters for the qualifiers in the LNKLST set name. For example,

```
CSVSYNL.MYLNK*
```

If you have RACF 1.9 or higher installed, you can use the following generic to cover all LNKLST set names:

```
CSVSYNL.**
```

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

```
SETOPTS GENERIC(FACILITY)
```

For coverage of LNKLST sets, check the names currently specified in the PROGxx parmlib members. Also use the DISPLAY PROG,LNKLST system command.

2. To permit the user (in this example user OPER1) to use the REQUEST=DEFINE option of the CSVSYNL macro for LNKLST set I, issue the following:

```
PERMIT CSVSYNL.1.DEFINE CLASS(FACILITY) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See "Defining RACF Profiles" on page 55.

3. If the FACILITY class is not already active, issue the SETOPTS command as follows:

```
SETOPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETOPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETOPTS RACLIST:

```
SETOPTS RACLIST(FACILITY) REFRESH
```

Controlling Adding a Data Set to a LNKLST Set

To control who can add a data set to a LNKLST set, the RACF security administrator can take the following steps:

1. To establish a profile for the LNKLST set name for the FACILITY class, issue RDEFINE:

```
RDEFINE FACILITY CSVDYNL.lnk1stname.ADD UACC(NONE)
```

where `lnk1stname` is the name of the LNKLST set. For example,
MYLNKLST.SET

You can use generic characters for the qualifiers in the LNKLST set name. For example,

```
CSVDYNL.MYLNK*
```

If you have RACF 1.9 or higher installed, you can use the following generic to cover all LNKLST set names:

```
CSVDYNL.**
```

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

```
SETROPTS GENERIC(FACILITY)
```

For coverage of LNKLST sets, check the names currently specified in the PROGxx parmlib members. Also use the DISPLAY PROG,LNKLST system command.

2. To permit the user (in this example user OPER1) to add a data set to LNKLST set I, issue the following:

```
PERMIT CSVDYNL.1.ADD CLASS(FACILITY) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See "Defining RACF Profiles" on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Controlling Deleting a Data Set from a LNKLST Set

To control who can delete a data set from a LNKLST set, the RACF security administrator can take the following steps:

1. To establish a profile for the LNKLST set name for the FACILITY class, issue RDEFINE:

```
RDEFINE FACILITY CSVDYNL.lnk1stname.DELETE UACC(NONE)
```

where `lnk1stname` is the name of the LNKLST set. For example,
MYLNKLST.SET

You can use generic characters for the qualifiers in the LNKLST set name. For example,

```
CSVLYNL.MYLNK*
```

If you have RACF 1.9 or higher installed, you can use the following generic to cover all LNKLST set names:

```
CSVLYNL.**
```

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

```
SETROPTS GENERIC(FACILITY)
```

For coverage of LNKLST sets, check the names currently specified in the PROGxx parmlib members. Also use the DISPLAY PROG,LNKLST system command.

2. To permit the user (in this example user OPER1) to delete a data set from LNKLST set I, issue the following:

```
PERMIT CSVLYNL.1.DELETE CLASS(FACILITY) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Controlling Removing the Definition of a LNKLST Set

To control who can remove the definition of a LNKLST set, the RACF security administrator can take the following steps:

1. To establish a profile for the LNKLST set name for the FACILITY class, issue RDEFINE:

```
RDEFINE FACILITY CSVLYNL.lnk1stname.UNDEFINE UACC(NONE)
```

where lnk1stname is the name of the LNKLST set. For example,
MYLNKLST.SET

You can use generic characters for the qualifiers in the LNKLST set name. For example,

```
CSVLYNL.MYLNK*
```

If you have RACF 1.9 or higher installed, you can use the following generic to cover all LNKLST set names:

```
CSVLYNL.**
```

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

```
SETROPTS GENERIC(FACILITY)
```

For coverage of LNKLST sets, check the names currently specified in the PROGxx parmlib members. Also use the DISPLAY PROG,LNKLST system command.

2. To permit the user (in this example user OPER1) to remove the definition of LNKLST set I, issue the following:

```
PERMIT CSVDYNL.1.UNDEFINE CLASS(FACILITY) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Controlling Testing of a LNKLST Set

To control who can test a LNKLST set, the RACF security administrator can take the following steps:

1. To establish a profile for the LNKLST set name for the FACILITY class, issue RDEFINE:

```
RDEFINE FACILITY CSVDYNL.1nk1stname.TEST UACC(NONE)
```

where 1nk1stname is the name of the LNKLST set. For example,
MYLNKLST.SET

You can use generic characters for the qualifiers in the LNKLST set name. For example,

```
CSVDYNL.MYLNK*
```

If you have RACF 1.9 or higher installed, you can use the following generic to cover all LNKLST set names:

```
CSVDYNL.**
```

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

```
SETROPTS GENERIC(FACILITY)
```

For coverage of LNKLST sets, check the names currently specified in the PROGxx parmlib members. Also use the DISPLAY PROG,LNKLST system command.

2. To permit the user (in this example user OPER1) to test LNKLST set I, issue the following:

```
PERMIT CSVDYNL.1.TEST CLASS(FACILITY) ID(OPER1) ACCESS(READ)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Controlling Updating of a Job’s LNKLST Set

To control who can update a job to use the current LNKLST, the RACF security administrator can take the following steps:

1. To establish a profile for updating LNKLSTs for the FACILITY class, issue RDEFINE:

```
RDEFINE FACILITY CSVDYNL.UPDATE.LNKLST UACC(NONE)
```

2. To permit the user (in this example user OPER1) to update a job to use the current LNKLST, issue the following:

```
PERMIT CSVDYNL.1.UPDATE CLASS(FACILITY) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Controlling Activation of a LNKLST Set

To control who can activate a LNKLST set, the RACF security administrator can take the following steps:

1. To establish a profile for the LNKLST set name for the FACILITY class, issue RDEFINE:

```
RDEFINE FACILITY CSVDYNL.lnk1stname.ACTIVATE UACC(NONE)
```

where lnk1stname is the name of the LNKLST set. For example,

```
MYLNKLST.SET
```

You can use generic characters for the qualifiers in the LNKLST set name. For example,

```
CSVDYNL.MYLNK*
```

If you have RACF 1.9 or higher installed, you can use the following generic to cover all LNKLST set names:

CSVDYNL.**

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

```
SETROPTS GENERIC(FACILITY)
```

For coverage of LNKLST sets, check the names currently specified in the PROGxx parmlib members. Also use the DISPLAY PROG,LNKLST system command.

2. To permit the user (in this example user OPER1) to activate LNKLST set I, issue the following:

```
PERMIT CSVDYNL.1.ACTIVATE CLASS(FACILITY) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Using RACF to Control Dynamic LPA

RACF allows you to control the use of the MVS commands SETPROG and SET PROG, and the use of the CSVDYLPA macro, for processing Dynamic LPA.

The SETPROG command allows a user to add modules to the LPA following IPL, delete modules from the LPA following IPL, and set threshold values for minimum amounts of CSA storage that still must be available after an ADD operation. SET PROG allows a user to activate the PROGxx member of SYS1.PARMLIB that contains definitions for controlling Dynamic LPA. CSVDYLPA is an authorized MVS macro that allows you to perform the dynamic LPA processing from an application program.

Note: For information on using CSVDYLPA, including authorization required with RACF, see *z/OS MVS Programming: Authorized Assembler Services Guide*. For information on using PROGxx, see *z/OS MVS Initialization and Tuning Reference*.

Command Authorization

An operator can issue the SETPROG or SET PROG command from a console with AUTH(SYS) or higher. If RACF authorization checking is in effect, you can control the use of these commands through RACF profiles. RACF authorization checking overrides the CONSOLxx AUTH specification.

To use RACF authorization checking to control any MVS command, the security administrator must ensure that each userid that issues the command is defined to RACF. Operators with a userid and a RACF profile can log on to a console, or the

installation can define a RACF userid for the console itself. (For information, see “Using RACF to Control Command Authority and Operator Logon” on page 54 and “Defining RACF Profiles” on page 55.)

Defining Command Profiles

To define the resource profile for SETPROG, the RACF administrator can take the following steps:

1. To create a profile for the SETPROG command, issue RDEFINE:
RDEFINE OPERCMDS MVS.SETPROG UACC(NONE)
2. To permit the userid for the user logging on to the console (in this example user OPER1) to use the command in the OPERCMDS class, issue the following:
PERMIT MVS.SETPROG CLASS(OPERCMDS) ID(OPER1) ACCESS(UPDATE)

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the OPERCMDS class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(OPERCMDS)
```

(To ensure that the OPERCMDS class is active, you can issue the SETROPTS LIST command.)

4. To refresh the OPERCMDS resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(OPERCMDS) REFRESH
```

For the SET PROG command, you follow the same steps as outlined for SETPROG but use the following RACF profile name:

```
MVS.SET.PROG
```

When you have given access to users of SETPROG and SET PROG, you can further control the use of the command.

Controlling Adding A Module to LPA after IPL

To control who can add a particular module to the LPA after IPL, the RACF security administrator can take the following steps:

1. To establish a profile for the library name for the FACILITY class, issue RDEFINE:

```
RDEFINE FACILITY CSVDYLPA.ADD.modname UACC(NONE)
```

where modname is the name of the module to add to the LPA. For example, MYMODULE

You can use generic characters for the qualifiers in the module name. For example,

```
CSVDYLPA.ADD.M*
```

If you have RACF 1.9 or higher installed, you can use the following generic to cover all module names:

```
CSVDYLPA.ADD.**
```

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

```
SETROPTS GENERIC(FACILITY)
```

2. To permit the user (in this example user OPER1) to add module m to the LPA, issue the following:

```
PERMIT CSVLYLPA.ADD.m CLASS(FACILITY) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Controlling Deleting A Module from LPA after IPL

To control who can delete a particular module from dynamic LPA, the RACF security administrator can take the following steps:

1. To establish a profile for the library name for the FACILITY class, issue RDEFINE:

```
RDEFINE FACILITY CSVLYLPA.DELETE.modname UACC(NONE)
```

where modname is the name of the module to delete from the LPA. For example, MYMODULE

You can use generic characters for the qualifiers in the module name. For example,

```
CSVLYLPA.DELETE.M*
```

If you have RACF 1.9 or higher installed, you can use the following generic to cover all module names:

```
CSVLYLPA.DELETE.**
```

To ensure that generic profile checking is in effect for the class FACILITY, issue the following command:

```
SETROPTS GENERIC(FACILITY)
```

2. To permit the user (in this example user OPER1) to delete module m from the LPA, issue the following:

```
PERMIT CSVLYLPA.DELETE.m CLASS(FACILITY) ID(OPER1) ACCESS(UPDATE)
```

OPER1 must be the name of a RACF-defined user or group profile.

Note: Instead of specifying individual userids, you can specify the name of a RACF group profile and connect authorized users to the group. See “Defining RACF Profiles” on page 55.

3. If the FACILITY class is not already active, issue the SETROPTS command as follows:

```
SETROPTS CLASSACT(FACILITY)
```

(To ensure that the FACILITY class is active, you can issue the SETROPTS LIST command.)

4. To refresh the FACILITY resource class, issue SETROPTS RACLIST:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Managing Messages with a Console Cluster

A console cluster is a good way to divide functions and handle message traffic in an MVS console configuration. In a console cluster, you define a group of consoles located together, each console handling a different function. One console can receive system status displays, another unsolicited messages, and still another operate as a full-capability console to handle commands.

You can set up a console cluster for any console location. Because of the high volume of message traffic to the master console in some installations, however, the usual place for a console cluster is in the master console area.

You can design a master console cluster to suit the special needs of your installation. A typical master console cluster might consist of four consoles, placed next to each other with one console as the master console. You could set up the consoles to receive these messages:

- The master console in full-capability mode to receive the action messages or important informational messages (those with descriptor code 12) that you **must** see.
- A console in message stream mode to receive the information messages that you **must** see.
- A console in message stream mode to receive ordinary system message traffic (This console gives you basic information on how the system is running.)
- A console in status display mode to dynamically display the active jobs in the system and provide display areas for system status displays.

Include other consoles in the cluster if you want to divide the master console message traffic even more.

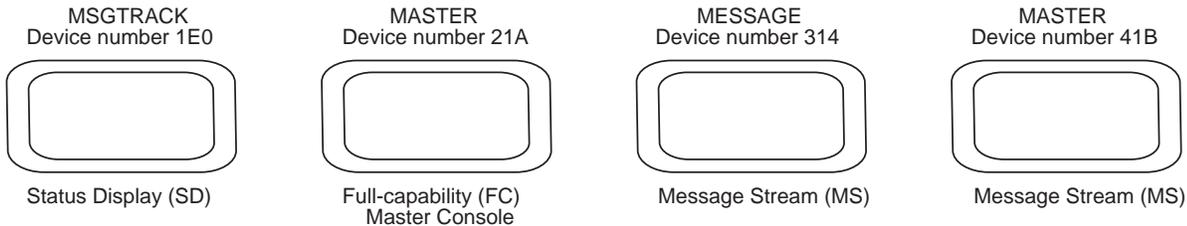
Setting Up and Using a Master Console Cluster

If you decide to set up a master console cluster, you might want to follow the procedures outlined in the following detailed example. You need not follow the example exactly as it is given. Depending on your needs and the characteristics of your consoles, choose your own values for area sizes and numbers, PFK definitions, commands, and so forth.

This example describes how to set up and use a master console cluster that consists of four consoles. The example assumes that:

- Each of the devices is a 3270-type device with a screen that holds 43 lines. Also, the device has 24 PFKs.
- Two consoles require keyboards: the master console and the console you specify as its alternate.

- All devices in the cluster come online during the IPL process. They come online with the characteristics that you define in CONSOLxx member. The PFKs on the consoles are defined at IPL with the definitions you establish in the PFK table you assign to the console.
- The console names and device numbers of the consoles used in this example are as follows (the mode each console is in when you finish setting up the cluster is also shown):



You should put the four devices in the cluster on different control units, if possible, to make recovery easier if a control unit fails.

Many of the statements you define in the CONSOLxx member serve to divide the message traffic among the consoles and set up the message roll rate for each screen. When you complete the procedure described on the following pages:

- MSGTRACK, the status display console, will receive the output from the DISPLAY and TRACK commands.
- MASTER, the full-capability console, will receive the messages that the master console operator **must** act on. The console will be in roll-deletable mode. (In roll-deletable mode, outstanding action messages are not automatically removed from the screen.)
- MESSAGE, a message stream console, will receive the messages that operators at other consoles must act on. The console will be in wrap mode.
- INFO, another message stream console, will receive all the information messages in the system. The console will be in roll mode. (In roll mode, a specified number of flagged messages roll off the screen after a specified time interval.)

The procedure for setting up a console cluster involves coding the statements in CONSOLxx and placing PFK definitions in a PFK table in the PFKTABxx Parmlib member. See “Summary of Contents of CONSOLxx for the Cluster” on page 180 for a summary of the coded CONSOLE statements used in this example. See “Defining PFKs for the Master Console” on page 181 for a summary of the PFK table definitions for the console MASTER.

Operators can use commands to change these values; however, in this example, only the SYS1.PARMLIB definitions are shown.

Setting up a master console cluster requires several steps. This example describes:

- How to set up console recovery for the consoles
- How to define routing codes for the consoles
- How to define the operating modes and message levels for the consoles
- How to set up display areas
- How to set up a TRACK display

- How to set message roll rates and message deletion specifications for the consoles
- How to direct command responses to specific consoles
- How to set up a periodic display of outstanding requests for JES2 or JES3
- How to define program function keys (PFKs)

Setting Up Console Recovery for the Consoles

If a console fails, MVS can search the chain of consoles that are members of the console group defined on ALTGRP for the console to find an active alternate. In this example, the console group MSTCLST defines the chain of consoles MVS can search to select an active alternate for MASTER. Console groups ALTCLST1 and ALTCLST2 establish console recovery for the other consoles in the example.

All console groups for this example are defined in CNGRP01 as follows:

```
GROUP  NAME(MSTCLST)
        MEMBERS (MESSAGE, INFO,MSGTRACK)

GROUP  NAME(ALTCLST1)
        MEMBERS (MSGTRACK,MASTER)

GROUP  NAME(ALTCLST2)
        MEMBERS (MASTER,MESSAGE)
```

Code the following statements in CONSOLxx with the device numbers appropriate to your installation. (In each example, the appropriate keywords are highlighted.)

```
CONSOLE DEVNUM(1E0) NAME(MSGTRACK) ALTGRP(ALTCLST2)
CONSOLE DEVNUM(21A) NAME(MASTER) ALTGRP(MSTCLST) AUTH(MASTER)
CONSOLE DEVNUM(314) NAME(MESSAGE) ALTGRP(ALTCLST1)
CONSOLE DEVNUM(41B) NAME(INFO) ALTGRP(ALTCLST2)
INIT CNGRP(01)
```

In this example, CNGRP01 must be active for the consoles to use console group recovery. If the master console fails, the system merges the console attributes of MASTER with those of MESSAGE, the first console member in the console group MSTCLST, makes MESSAGE the new master console, and changes its operating mode to full-capability. If MESSAGE fails, MVS can merge the console attributes of MASTER with those of INFO and make INFO the new master console. The attributes for the failing console MESSAGE are merged with those of MSGTRACK if active, the first console in MESSAGE's alternate console group ALTCLST1.

Console switching by group allows the console attributes of a failing console to remain in the functional group of consoles defined on the ALTGRP keyword. Thus, you can design console recovery in the configuration according to console function (for example, a console group to handle printer functions, one to handle tape functions, or another for master console functions).

Defining Routing Codes for the Consoles

Use routing codes to set up MASTER so that it receives only messages for which the master console operator is responsible. Direct other messages to MESSAGE and all routing codes to INFO. In the next section, you will see how the LEVEL parameter in CONSOLxx further limits messages to these consoles. Code the following statements to set up the routing codes for MASTER and the message stream consoles (MESSAGE and INFO), substituting device numbers appropriate to your installation:

```

CONSOLE DEVNUM(21A) NAME(MASTER ) ROUTCODE(1,2,9,10)
CONSOLE DEVNUM(314) NAME(MESSAGE) ROUTCODE(3-8,12-15,42)
CONSOLE DEVNUM(41B) NAME(INFO) ROUTCODE(ALL)

```

Note: You do not have to direct tape, DASD, unit record, and teleprocessing messages (routing codes 3, 4, 5, 6, 7, and 8) to MESSAGE if a console other than the master is receiving these routing codes and the master console operator does not have to take any action for such messages.

As a result of these statements, MASTER, MESSAGE, and INFO display the complete range of master console messages. You do not need to define routing codes for MSGTRACK because you are going to put MSGTRACK in status display mode. In status display mode, a console can display only system status display messages that you direct specifically to the console.

Defining the Operating Modes and the Message Levels for the Consoles

Code the following statement in CONSOLxx to define the operating mode of MSGTRACK to output-only for system status displays:

```

CONSOLE DEVNUM(1E0) NAME(MSGTRACK) USE(SD)

```

Use statements in CONSOLxx to define the operating modes and the message levels for MASTER, MESSAGE, and INFO. To further reduce the messages that appear at the master console, which is already in full-capability mode, eliminate non-action messages from MASTER. Code the following statement in CONSOLxx:

```

CONSOLE DEVNUM(21A) NAME(MASTER) LEVEL(R,I,CE,E,NB)

```

As a result of this statement MASTER receives all action messages with routing codes 1, 2, 9, and 10; it receives no informational or broadcast messages.

Define MESSAGE and INFO as message stream consoles. Set up message levels for the two consoles so that MESSAGE receives all the action messages that the master console does not receive, and INFO receives all the informational messages for the system. Code the following statement in CONSOLxx:

```

CONSOLE DEVNUM(314) NAME(MESSAGE) USE(MS) LEVEL(R,I,CE,E,NB)
CONSOLE DEVNUM(41B) NAME(INFO) USE(MS) LEVEL(IN)

```

As a result of these statements, MESSAGE receives all messages with routing codes 3, 4, 5, 6, 7, 8, 12, 13, 14, 15, and 42 that require operator response; INFO receives all informational messages that the system issues.

Setting Up Display Areas

The next step is to define two display areas on MSGTRACK. In this example, the screen size of your consoles is 43 lines. Specify 28 lines for the bottommost area (area A). When you put a TRACK command display in this area, you can monitor the activity of up to 50 jobs. Specify the remaining 15 lines for the second area (area B). You can direct system status displays to this area. To define these areas, code the following statement in CONSOLxx:

```

CONSOLE DEVNUM(1E0) NAME(MSGTRACK) AREA(28,15)

```

MASTER should also have a display area because display area B on MSGTRACK might not be able to receive all system status displays. An area of ten lines should be enough. To establish this area, code the following statement in CONSOLxx:

```

CONSOLE DEVNUM(21A) NAME(MASTER) AREA(10)

```

The display areas you have established on the consoles are:

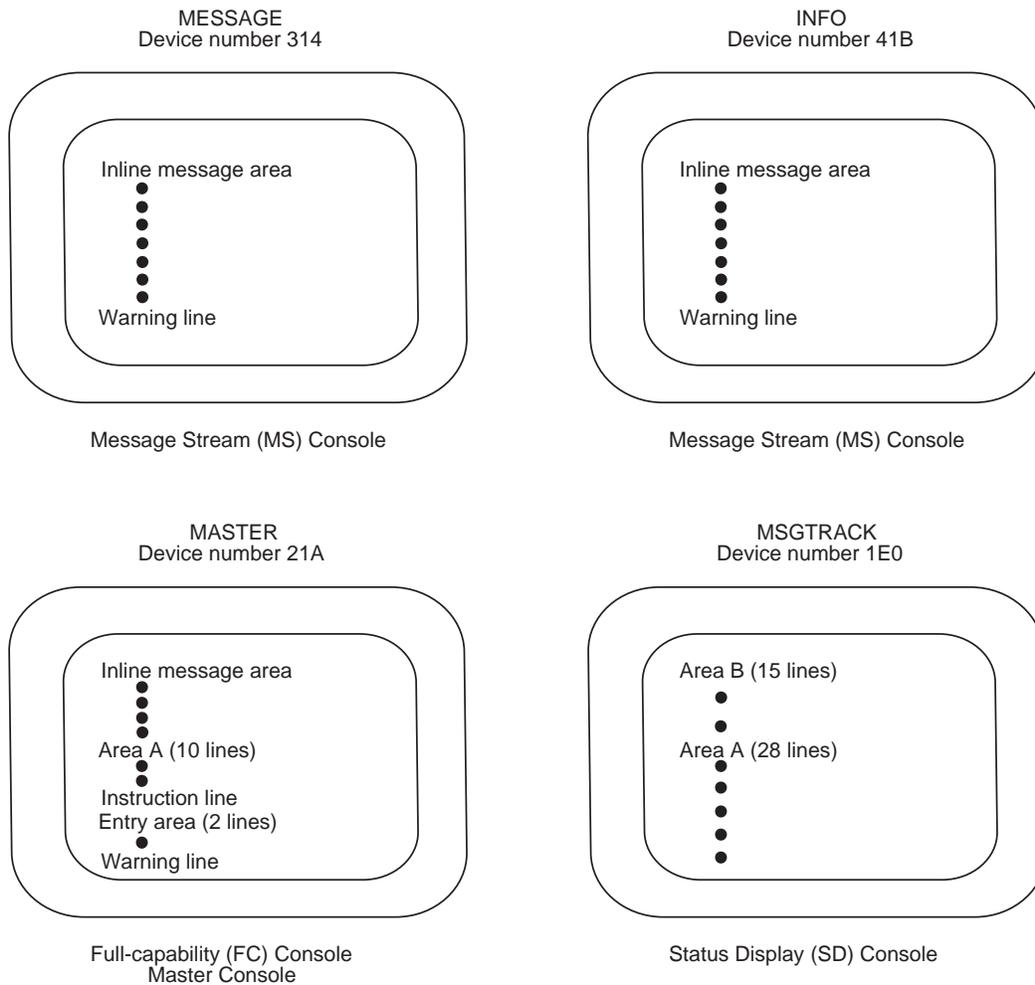


Figure 14. Display Areas on Consoles in the Console Cluster

Setting Up a TRACK Display

Output from the TRACK command is the same as output from the DISPLAY A command. The main difference is that the system periodically updates the display produced by the TRACK command.

Output from the TRACK command starts with general system statistics: the number of jobs, mounts and starts, time-sharing users, system address spaces, initiators, actual TSO/VTAM users, and allowed TSO/VTAM users. This information appears on the first three lines of the display area. Each of the other lines lists information for two specific jobs in the system. The information displayed for each job includes: the job name, the step name, the procedure step name, and the status of the job's address space.

If fewer than 50 jobs are executing and time-sharing is active, the status of time-sharing users follows the status of jobs on the first frame. If there are no time-sharing users, the bottom lines of the display area are blank. If more than 50 jobs are active and you want to see the status of the jobs or time-sharing users that are not displayed, you might want to use a PFK to frame through the display. The

statement in CONSOLxx that defines this PFK is described later in this example under "Defining PFKs for the Master Console" on page 181.

When you issue a TRACK command, the system automatically updates the TRACK display every 30 seconds. Because you might need more frequent status updates to keep track of a specific job's execution, you should cause the system to update the TRACK display every ten seconds.

To update the TRACK display every ten seconds, code the following statement in CONSOLxx:

```
CONSOLE DEVNUM(1E0) NAME(MSGTRACK) UTME(10)
```

If the TRACK display is updated this frequently, you might not want its output to appear in the hardcopy log. To keep the TRACK display out of the hardcopy log, code the following statement in CONSOLxx:

```
HARDCOPY CMDLEVEL(STCMDS)
```

The mode of the TRACK display can be either update mode or hold mode. In update mode, the system updates the display, for this example, every ten seconds. An operator should not try to frame forward through the TRACK display when it is in update mode.

The following shows you the information that appears in MSGTRACK when the TRACK display is in update mode:

```
IEE104I 14.15.04 90.120 ACTIVITY      FRAME LAST          PT  H  1A
JOBS      M/S      TS USERS      SYSAS  INITS      ACTIVE/MAX VTAM
00000      00002      00000      00012  00001      00000/00000
LLA      LLA      LLA      NSW S  JES2      JES2      IEFPROC  NSW  S
```

When the system updates the display, it returns the display to its first frame, requiring the operator to start framing all over again to find the needed information.

To freeze the TRACK display, the operator can use the CONTROL D,H command. Then the operator can frame through the display with the CONTROL D,F command.

The following shows you the information that appears in MSGTRACK when the TRACK display is in hold mode:

```
IEE104I 14.15.04 90.120 ACTIVITY      FRAME LAST          F  U  1A
JOBS      M/S      TS USERS      SYSAS  INITS      ACTIVE/MAX VTAM
00000      00002      00000      00012  00001      00000/00000
LLA      LLA      LLA      NSW S  JES2      JES2      IEFPROC  NSW  S
```

In hold mode, the system does not update a TRACK display until the operator issues the CONTROL D,U command.

Next, place the TRACK command in COMMNDxx:

```
TRACK A,L,L=MSGTRACK-A
```

COMMNDxx is the SYS1.PARMLIB member that allows you to specify commands the system automatically issues at initialization. When the system is IPLed, it automatically issues the TRACK command and sends the output to out-of-line area A on MSGTRACK. For information about COMMNDxx, see *z/OS MVS Initialization and Tuning Reference*.

When the operator brings up the console, the TRACK display appears in area A of MSGTRACK. In “Defining PFKs for the Master Console” on page 181, PFKs 22, 23, and 24 are defined to allow the operator to use the TRACK display efficiently.

Setting Message Roll Rates and Message Deletion Specifications for the Consoles

The message roll rate appropriate for a console depends on the message traffic to that console. To establish a starting message roll rate for consoles 2, 3, and 4, code the following statements in CONSOLxx:

```
CONSOLE DEVNUM(21A) NAME(MASTER) DEL(RD) SEG(39) RNUM(10) RTME(2)
CONSOLE DEVNUM(314) NAME(MESSAGE) DEL(W) RTME(1/2)
CONSOLE DEVNUM(41B) NAME(INFO) DEL(R) SEG(39) RNUM(10) RTME(2)
```

These statements put MASTER in roll-deletable mode, MESSAGE in wrap mode, and INFO in roll mode. (See “Specifying Automatic Message Deletion for MCS or SMCS Consoles” on page 72 for a description of roll, roll-deletable, and wrap modes.) Adjust the RNUM and RTME specifications until the roll rate is appropriate for the message traffic on MASTER and INFO. Adjust the RTME specification for MESSAGE in wrap mode. Also specify HOLDMODE for MASTER, MESSAGE, and INFO. Code the following DEFAULT statement:

```
DEFAULT HOLDMODE(YES)
```

HOLDMODE allows operators to freeze the screen for consoles in roll, roll-deletable, or wrap mode by pressing the enter key. To unfreeze the screen, operators can press the enter key again. See “Temporarily Suspending the Screen Roll” on page 76.

Once you determine the appropriate values for RNUM and RTME, code the values in the RNUM and RTME parameters in CONSOLxx.

Use the MSGRT statement in CONSOLxx to direct the responses to certain MVS commands to specific consoles without having to specify the location operand (L=) on each command. For this example, direct the responses to all DISPLAY commands you enter on the master console (MASTER) to the out-of-line display area B of MSGTRACK. You should also cause any TRACK commands you issue on MASTER to appear in the out-of-line display area A of MSGTRACK. Code the following statement in CONSOLxx:

```
CONSOLE DEVNUM(21A) NAME(MASTER) MSGRT('D=A,C,CONSOLES,D,DUMP,GRS,M,MPF
PFK,R,S,SMF,U,3850,OPDATA),L=MSGTRACK-B','TR=A,L=MSGTRACK-A')
```

As a result of this statement:

- Any of the following DISPLAY commands you enter on MASTER appear in display area B of MSGTRACK:

DISPLAY A	DISPLAY GRS	DISPLAY SLIP
DISPLAY C,K	DISPLAY M	DISPLAY SMF
DISPLAY CONSOLES	DISPLAY MPF	DISPLAY U
DISPLAY DMN	DISPLAY PFK	DISPLAY 3850
DISPLAY DUMP	DISPLAY R	DISPLAY OPDATA

- The responses to all TRACK commands you enter on MASTER appear in display area A on MSGTRACK.

If your system includes JES2, when you bring up the console, you can use the JES2 REDIRECT command to direct the responses to certain JES2 \$D commands to specific consoles. For this example, direct the responses to the JES2 commands

\$DA, \$DF, \$DI, \$DJ, \$DN, \$DQ, and \$DU to display area B of MSGTRACK. Issue the following command to make this change:

```
REDIRECT(MASTER),DA=MSGTRACK-B,DF=MSGTRACK-B
DF=MSGTRACK-B,DI=MSGTRACK-B,DJ=MSGTRACK-B,DN=MSGTRACK-B,
DQ=MSGTRACK-B,DU=MSGTRACK-B
```

Note: Put this command in the JES2 initialization data set so that it is issued automatically once JES2 is initialized. For more information, see *z/OS JES2 Initialization and Tuning Reference*.

If your system includes JES3, use JES3 commands to direct messages to specified consoles.

Setting Up a Periodic Display of Outstanding Requests

If your system includes JES2, you can have the system periodically display outstanding requests so that you always know how many there are.

You can set up such a periodic display through the JES2 automatic command facility, telling JES2 to issue a command at a defined interval. (The minimum time interval you can specify is 30 seconds.) You must use the \$TA JES2 command to define both the command you want issued and the number of seconds in the interval between commands. To cause JES2 to issue a DISPLAY R command every 60 seconds and to direct the command output to display area B of MSGTRACK, issue the following command:

```
$TA,I=60,'$VS,' 'D R,L,L=MSGTRACK-B''
```

Note: Put this command in the JES2 initialization data set so that it is issued automatically once JES2 is initialized.

You use the \$ZA JES2 command to temporarily stop JES2 from issuing the defined commands. You use the \$SA command to cause JES2 to resume issuing the defined commands. Use the \$CA command to cancel both the defined commands and time interval:

```
$CAxxxx
```

where xxxx is the ID of the periodic display.

Summary of Contents of CONSOLxx for the Cluster

The statements you place in CONSOLxx to initialize the cluster are:

```
CONSOLE DEVNUM(1E0) NAME(MSGTRACK) ALTGRP(ALTCLST2)
USE(SD)
AREA(28,15)
```

```
CONSOLE DEVNUM(21A) NAME(MASTER) ALTGRP(MSTCLST) AUTH(MASTER)
ROUTCODE(1,2,9,10)
LEVEL(R,I,CE,E,NB)
AREA(10)
DEL(RD) SEG(39) CON(N) RNUM(10) RTME(2)
MSGRT('D=(A,C,CONSOLES,D,DUMP,GRS,M,MPF,PFK,R,S,
SMF,U,3850,OPDATA),L=MSGTRACK-B','TR=A,L=MSGTRACK-A')
PFKTAB(MASTCMDS)
```

```
CONSOLE DEVNUM(314) NAME(MESSAGE) ALTGRP(ALTCLST1)
ROUTCODE(3-8,12-15,42)
USE(MS) LEVEL(R,I,CE,E,NB)
DEL(W) RTME(2)
```

```
CONSOLE DEVNUM(41B) NAME(INFO) ALTGRP(ALTCLST2)
```

```
ROUTCODE(ALL)
USE(MS) LEVEL(IN)
DEL(R) SEG(39) RNUM(10) RTME(2)
```

```
HARDCOPY CMDLEVEL(STCMDS)
```

```
DEFAULT HOLDMODE(YES)
```

```
INIT PFK(02) CNGRP(01)
```

Notes:

1. Substitute the device numbers and console names that are appropriate to your installation.
2. Adjust SEG, RNUM, RTME, and other values, as appropriate to the devices in your console cluster.
3. If you have JES2 at your installation, place the following command in the initialization data set:
 - REDIRECT(MASTER),DA=MSGTRACK-B,DF=MSGTRACK-B,
DF=MSGTRACK-B,DI=MSGTRACK-B,DJ=MSGTRACK-B,
DN=MSGTRACK-B,DQ=MSGTRACK-B,DU=MSGTRACK-B
 - \$TA,I=60,\$VS,"D R,L"

Defining PFKs for the Master Console

You have to redefine some of the PFKs 1 through 9 that the system assigns at IPL and define additional PFKs for MASTER because:

- The nine PFKs are not enough to set up and use the console cluster effectively.
- PFKs 1 through 9 do not put the commands you need to operate the console cluster in the most convenient places.

You need to define PFKs on MASTER for the common operator command functions and the commands to control the console cluster because the master console is the only full-capability console in the cluster.

Place all your definitions for PFKs in a PFK table that you create with the name MASTCMDS. All the definitions in this section follow the first statement in the table:

```
PFKTAB TABLE(MASTCMDS)
```

The commands you define in this table go into effect at IPL, providing you activate the table. The section "Activating the PFK Table" on page "Activating the PFK Table" on page 184 describes how you activate MASTCMDS by defining the PFKTABxx member in SYS1.PARMLIB member that contains it. (To change PFK tables, operators can use the SET PFK command. To dynamically redefine a PFK, operators can use the CONTROL N,PFK command. See *z/OS MVS System Commands* for how to use these commands.)

Define PFKs 13, 14, 17, and 18, to enter the functions defined for PFKs 1, 2, 5, and 6 at IPL. Add the following entries to MASTCMDS to control erasing and displaying of messages on MASTER:

```
PFK(13) CMD('K E,1')
PFK(14) CMD('K E')
PFK(17) CMD('K S,DEL=N')
PFK(18) CMD('K S,DEL=RD')
```

For controlling the cluster, define PFKs 15 and 16 to erase and frame system status displays on MASTER, code the following entry in MASTCMDS:

```
PFK(15) CMD('K E,D,L=MSGTRACK-B')
PFK(16) CMD('K D,F,L=MSGTRACK-B')
```

As a result of these definitions, PFK 15 erases a status display from display area B of MSGTRACK and PFK 16 frames a status display in display area B of MSGTRACK.

To establish the message routing instructions for JES2 messages, add the following entry to MASTCMDS:

```
PFK(3) CMD("$ADD REDIRECT(MASTER),DA=MSGTRACK-B,DF=MSGTRACK-B,
           DF=MSGTRACK-B,DI=MSGTRACK-B,DJ=MSGTRACK-B,DN=MSGTRACK-B,
           DQ=MSGTRACK-B,DU=MSGTRACK-B;$TA,I=60,'$VS,
           'D R,L,L=MSGTRACK-B''')
```

For more information about the JES2 REDIRECT command, see *z/OS JES2 Commands*.

As a result of this definition, pressing PFK 3

- Directs the output of any of the following JES2 \$D commands entered on MASTER to display area B of MSGTRACK:

```
$DA    $DJ    $DQ
$DF    $DN    $DU
$DI
```

- Makes the JES2 automatic command facility issue a DISPLAY R command every 60 seconds and direct the command response to display area B of MSGTRACK.

You should define PFKs to remove action messages quickly from the screen of the master console (MASTER) because the console will be in roll-deletable mode. In roll-deletable mode, outstanding action messages are not automatically removed from the screen. Therefore, if you do not remove the action messages, the screen eventually fills with these messages and messages that are waiting to appear start to use up the message buffer space.

Define PFK 12 by adding the following entry to MASTCMDS:

```
PFK(12) CMD('K S,NAME(MASTER) DEL=R')
```

As a result of this command, pressing PFK 12 causes the master console to roll all messages.

When PFK 12 makes MASTER roll all its messages, it reduces the number of backed-up messages, in effect, by displaying them all.

Notes:

1. If the action message retention facility is active, operators can issue a DISPLAY R command to display again any action messages that are retained (that is, the messages that roll off or are erased from a screen).
2. If action messages fill up a console screen frequently, operators should first make sure that they are responding to the messages. If they do not respond to them, the system cannot remove them automatically from the screen. If they are responding to the messages as they should, check the configuration of the console cluster. You might have to:
 - Add another console to the cluster so you can split up the message traffic even more
 - Include more large-screen devices (such as the 3290) in the cluster

- Keep the master console in the roll mode of message deletion (instead of the roll-deletable mode) so that all messages roll off the screen
3. Because console MESSAGE is in wrap mode, action messages are automatically overlaid as new messages appear on the screen. There is no need to define PFKs to remove action messages as for MASTER.

You should define a PFK to display all the outstanding requests at once so you can always keep track of or respond to them. Define PFK 21 by adding the following entry to MASTCMDS:

```
PFK(21) CMD('K V,USE=MS,L=MSGTRACK;K V,USE=SD,L=MSGTRACK;K A,15,18,10,
           L=MSGTRACK;$DU,L=MSGTRACK-A;D R,L,L=MSGTRACK-B')
```

As a result of this command, pressing PFK 21:

- Changes MSGTRACK to message stream mode
- Puts MSGTRACK back in status display mode
- Defines new out-of-line display areas A (15 lines), B (18 lines), and C (10 lines) for MSGTRACK
- Displays JES2 unit record device status in out-of-line display area A of MSGTRACK
- Displays outstanding requests in out-of-line display area B of MSGTRACK

When MSGTRACK enters message stream mode, the system stops the TRACK display in area A and erases any display in area B, thus clearing the screen for the display of outstanding requests.

You should define a PFK to restore the original TRACK command display when you no longer need the display of outstanding requests. Define PFK 22 by issuing:

```
PFK(22) CMD('K V,USE=MS,L=MSGTRACK;K V,USE=SD,L=MSGTRACK;K A,28,15,
           L=MSGTRACK;TR A,L;K T,UTME=10,L=MSGTRACK')
```

As a result of this command, pressing PFK 22:

- Erases any system status displays on MSGTRACK's screen by changing the mode of MSGTRACK from status display to message stream and back to status display.
- Defines display areas A and B on MSGTRACK.
- Starts a TRACK display on MSGTRACK.
- Sets the time interval for updating the TRACK display to 10 seconds.

You should define a PFK to let you frame through the TRACK display on MSGTRACK. Define PFK 23 by adding the following entry to MASTCMDS:

```
PFK(23) CMD('K D,H,L=MSGTRACK-A;K D,F,L=MSGTRACK-A')
```

As a result of this command, pressing PFK 23:

- Puts the TRACK command display area in hold mode. (For information on the modes of a TRACK display, see "Status Displays and MCS and SMCS Consoles" on page 79.)
- Causes the next frame of the TRACK display to appear.

If you want to frame further through the display, press PFK 23 again.

You should define another PFK to cause the system to continue updating the TRACK display after you are finished framing through it. Define PFK 24 by adding the following entry to MASTCMDS:

```
PFK(24) CMD('K D,U,L=MSGTRACK-A')
```

As a result of this command, pressing PFK 24 puts the TRACK display back in update mode.

Summary of the PFK Definitions for the Cluster

The PFK table named MASTCMDS now contains the definitions that have been defined in the previous section. If you issue DISPLAY PFK, TABLE=MASTCMDS, the definitions, including those that IBM supplies, display. In message IEE235I, the NO that appears in the column labelled CON, indicates that the commands are non-conversational. The display appears as follows:

```
PFK DEFINITIONS FOR MASTER TABLE=MASTCMDS IN PFKTAB02
```

```
KEY# CON -----DEFINITION-----
1 NO K E,1 ERASE TOP LINE FROM SCREEN
2 NO K E ERASE ONE SEGMENT FROM SCREEN
3 NO $ADD REDIRECT(MASTER),DA=MSGTRACK-B,DF=MSGTRACK-B,
DF=MSGTRACK-B,DI=MSGTRACK-B,DJ=MSGTRACK-B,DN=MSGTRACK-B,
DQ=MSGTRACK-B,DU=MSGTRACK-B;$TA,I=60,'$VS,'D R,L,L=MSGTRACK-B'
4 NO K D,F FRAME DISPLAY FORWARD IN AREA
5 NO K S,NAME(MASTER) DEL=N HOLD IN-LINE OUTPUT
6 NO K S,NAME(MASTER) DEL=RD RESUME IN-LINE OUTPUT
7 NO D A,L LIST ACTIVE JOBS AND TSO NAME(MASTER) USERS
8 NO D R,L LIST OPERATOR REQUESTS
9 NO K D,U UPDATE DYNAMIC DISPLAY
10 NOT DEFINED
11 NOT DEFINED
12 NO K S,NAME(MASTER) DEL=R
13 NO K E,1
14 NO K E
15 NO K E,D,L=MSGTRACK-B
16 NO K D,F,L=MSGTRACK-B
17 NO K S,DEL=N
18 NO K S,DEL=RD
19 NOT DEFINED
20 NOT DEFINED
21 NO PFK(21) CMD('K V,USE=MS,L=MSGTRACK;K V,USE=SD,L=MSGTRACK;
K A,15,18,10,L=MSGTRACK;$DU,L=MSGTRACK-A;
D R,L,L=MSGTRACK-B)
22 NO K V,USE=MS,L=MSGTRACK;K V,USE=SD,L=MSGTRACK;
K A,28,15,L=MSGTRACK;TR A,L;K T,UTME=10,L=MSGTRACK
23 NO K D,H,L=MSGTRACK-A;K D,F,L=MSGTRACK-A'
24 NO K D,U,L=MSGTRACK-A
```

Notes:

1. The PFKs that are noted NOT DEFINED are available for you to define according to your needs.
2. If you put the console into message stream or display status mode, you can no longer use the PFKs.

Activating the PFK Table

The PFK table named MASTCMDS must reside in a PFKTABxx Parmlib member. In this example, assume that the member is named PFKTAB02. The following statements in CONSOLxx activate MASTCMDS:

```
CONSOLE DEVNUM(21A) NAME(MASTER) PFKTAB(MASTCMDS)
INIT PFK(02)
```

The PFK commands you defined in MASTCMDS go in effect for MASTER at the next IPL.

Using the Master Console Cluster and Setting It Up Again

Now that you have set up all the consoles and defined all the PFKs that you need in the cluster, you can work normally with the console message traffic at MASTER. Press PFK 21 once in a while to display both the JES2 unit record device status and the outstanding requests on MSGTRACK. When you are done with this display, press PFK 22 to restore the TRACK display on MSGTRACK.

When you IPL again, the definitions you established in the CONSOLxx and PFKTABxx members are in effect. If you have JES2 on your system, as soon as JES2 becomes active, press PFK 3 to set up the JES2 message routing for the cluster and a periodic display of outstanding requests.

At this point, you have fully set up the cluster as before. You can now work normally again from the master console, using PFKs 21 and 22 as described earlier.

The 3290 as a Console Cluster

You can set up a single 3290 as an entire console cluster, as described in Figure 15. This example shows a four-part 3290 screen initialized like the four 3278 consoles in Figure 14. The consoles on the 3290 are arranged so that the master console is the lower left-hand quadrant with the status display console beside it; the two message stream consoles are above them. To set up the cluster, follow the steps described earlier in this section. (Because each of the four consoles screens on the 3290 is shorter than the 3278-4 used in the console cluster example, adjust the values for SEG, RNUM, RTME, and area sizes.)



Figure 15. Single 3290 Screen As a Console Cluster

Defining a Console Configuration for a Sysplex Environment

In a sysplex, your operators can receive messages from other systems and send commands to process on another system in the sysplex.

In this example, you want to define the console configuration for two MVS systems (SYA and SYB) that are part of a sysplex. Your console definitions reside in CONSOLxx, and you need to define your console configuration separately for each

system in the sysplex. In this example you will define two CONSOLxx members, one for SYA and one for SYB. (Both SYA and SYB are at the latest level of MVS/ESA SP Version 4.)

Planning Your Console Configuration for Each System

Before you start to define your consoles, it is a good idea to plan the console attachments to each system in the sysplex as well as the console groups that each system uses for recovery. You might ask yourself how you want your operators to be able to monitor the systems in the sysplex (you might want to limit message traffic, for example, using MSCOPE). For recovery purposes, you might want to define alternate consoles on other systems as part of console group definitions, or set up a console group on each system to handle synchronous messages.

Figure 16 illustrates one plan that you might use:

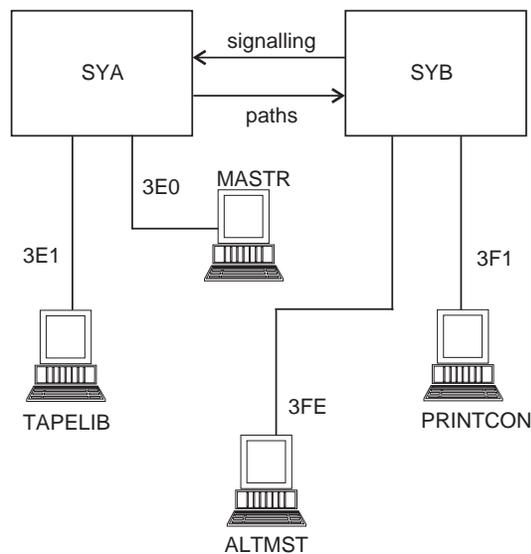


Figure 16. Console Configuration for a Two-System Sysplex

This configuration uses four consoles in the sysplex. Solid lines indicate physical attachments. MASTR and TAPELIB are both defined to SYA. PRINTCON and ALTMST are both defined to SYB. However, all consoles in this configuration have a logical connection to both systems. Full-capability consoles can receive messages from both SYA and SYB and enter commands to run on either SYA or SYB. PRINTCON is a console that monitors print operations for both systems. TAPELIB is a full-capability console that handles information for tape libraries. MASTR is the master console in the configuration, and ALTMST is one of the master console alternates defined with master authority.

Console Recovery for the Sysplex

The following console groups definitions are defined for the sysplex:

```
CNGRPOA  GROUP NAME (SYNCHSYA)
          MEMBERS (*MSTCON*, TAPELIB, *SYSCON*)

          GROUP NAME (MASTGRP)
          MEMBERS (TAPELIB, ALTMST)

          GROUP NAME (TAPEGRP)
          MEMBERS (MASTR, ALTMST, PRINTCON)

          GROUP NAME (NEWMSTG)
```

```

MEMBERS(ALT MST,TAPELIB,PRINTCON)

CNGRP0B GROUP NAME(SYNCHSYB)
MEMBERS(ALT MST,PRINTCON,*SYSCON*)

GROUP NAME(ALT MGRP)
MEMBERS(TAPELIB,PRINTCON)

GROUP NAME(PRINTGRP)
MEMBERS(ALT MST,MASTR,TAPELIB)

```

Both CNGRP0A and CNGRP0B can be specified on the INIT statement of the first system that joins the sysplex (in this example, SYA). Console group definitions are inherited by SYB when it joins the sysplex. When both SYA and SYB are active, console switching between systems is possible.

Group SYNCHSYA in CNGRP0A and SYNCHSYB in CNGRP0B define consoles that can receive synchronous messages. Because a console must be physically attached to the system that issues the synchronous message, consoles in SYNCHSYA are all attached to SYA, and consoles in SYNCHSYB are all attached to SYB. The system console (*SYSCON*) must be attached to ES/9000 control units 9021 or 9121 to be able to display synchronous WTO and WTOR messages. You define these console group names on DEFAULT SYNCHDEST for each system. (See "Planning Console Recovery" on page 43.)

The group NEWMSTG defines console members that the sysplex can use to select a master console during a no consoles condition.

Defining CONSOLxx for Each System

SYA and SYB use unique CONSOLxx members to define the console configuration for the sysplex in Figure 16 on page 186.

The CONSOLxx definitions for SYA are as follows:

```

CONSOLE DEVNUM(3E0) NAME(MASTR) UNIT(3270-X)
AUTH(MASTER) ROUTCODE(1,2,4,6,8,65-96,9,10)
ALTGRP(MASTGRP) USE(FC) DEL(RD) AREA(28,15)
RNUM(15) RTME(1)

CONSOLE DEVNUM(3E1) NAME(TAPELIB) UNIT(3270-X) AUTH(SYS,I0,CONS)
ROUTCODE(3,5,42) LEVEL(R,I,IN)
ALTGRP(TAPEGRP) USE(FC) DEL(R) RNUM(15) RTME(1)
AREA(28,15) MFORM(S)

INIT MPF(01,02,03) CNGRP(0A,0B) NOCCGRP(NEWMSTG)

DEFAULT HOLDMODE(YES) SYNCHDEST(SYNCHSYA)

```

You plan to IPL SYA into the sysplex first.

When SYA is IPLed into the sysplex, MASTR and TAPELIB are active. Both MASTR and TAPELIB are full-capability consoles. Console group members CNGRP0A and CNGRP0B are active and the console group definitions in both members are established for the sysplex. However, consoles on SYB are not yet active so switching from a console on SYA to an alternate console on SYB is not yet possible.

MASTR is in roll-deletable mode with 15 messages rolling off the screen every second. Action messages accumulate at the top of the message display area where the operator can delete them. MASTR also has two display areas defined of 28 lines and 15 lines.

TAPELIB is in roll mode also with 15 messages rolling every second. The CONSOLE statement for TAPELIB includes MFORM(S), which specifies that the name of the system that issues a message (SYA or SYB) will appear with the message text on the screen display for TAPELIB.

MASTR receives master console action and informational messages, messages about the disk library, processor information, security, and system error messages (indicated by routing codes) from both SYA and SYB. TAPELIB receives tape messages and general informational messages for JES2 (indicated by routing codes) from both systems. TAPELIB also receives certain messages indicated by message level (WTOR messages, immediate action messages, and informational messages). MASTR by default receives messages from all message levels.

SYA specifies MPFLST01, MPFLST02, and MPFLST03 on the INIT statement. The console group NEWMSTG specified on INIT NOCCGRP provides master console recovery if a no consoles condition occurs in the sysplex.

SYSLOG is the default for the hardcopy of messages that SYA issues, and HOLDMODE is in effect for the consoles on SYA. The console group SYNCHSYA is specified on DEFAULT SYNCHDEST. Consoles defined in SYNCHSYA can display synchronous messages.

The CONSOLxx definitions for SYB are as follows:

```
CONSOLE DEVNUM(3FE) NAME(ALTMST) UNIT(3270-X) AUTH(MASTER)
          ALTGRP(ALTGRP)
          ROUTCODE(1,2,8,12) LEVEL(R,I,CE)
          UD(Y) USE(FC) DEL(RD) RNUM(15) RTME(1)

CONSOLE DEVNUM(3E1) UNIT(3270-X) NAME(PRINTCON) ALTGRP(PRNTGRP)
          ROUTCODE(97-128) UD(Y) USE(FC) DEL(W) RTME(1/4)

INIT MPF(04) MMS(01) CNGRP(0C)

DEFAULT SYNCHDEST(SYNCHSYB)
```

When SYB is IPLed into the sysplex, ALTMST and PRINTCON are active on SYB. Both CNGRP0A and CNGRP0B are already active in the sysplex, so console switching to an alternate on a different system is possible.

Although SYB has specified CNGRP0C on the INIT statement, the sysplex ignores it. The first system that joins the sysplex with active CNGRPxx members establishes console group definitions for all systems in the sysplex. Operators must use the SET CNGRP command to change the members.

When both systems are active, all consoles receive messages from both systems. For example, ALTMST receives master console messages and messages for emulators and teleprocessing from SYA and SYB. ALTMST also receives selected messages by message level (WTOR, immediate action, and critical eventual action messages) from both systems. In addition, ALTMST has the UD attribute. The UD attribute allows ALTMST to receive undelivered action messages, WTOR messages, and informational messages with descriptor code 12. PRINTCON receives printer messages from SYA and SYB.

ALTMST is in roll-deletable mode with 15 messages rolling off the screen every second. PRINTCON is in wrap mode with messages appearing at the rate of 1/4 second.

SYB specifies MPFLST04 and MMSLST01 on the INIT statement. Because MMS, like MPF, has system scope, the MVS message service for translating messages is available only on SYB.

HOLDMODE(NO) and SYSLOG are default values for SYB. HOLDMODE is not in effect for consoles attached to SYB. The console group SYNCHSYB is specified on DEFAULT SYNCHDEST. Consoles defined in SYNCHSYB can display synchronous messages.

Appendix. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen-readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen-readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Volume I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Notices

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

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, NY 12601-5400
USA

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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

This information contains 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 contains 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 the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. 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.

Programming Interface Information

This book is intended to help a customer plan operations for MVS. This book primarily documents information that is NOT intended to be used as Programming Interfaces of z/OS.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- AnyNet
- BookManager
- C/370
- CICS
- CT

- DB2
- DFSMS
- DFSMS/MVS
- DFSMSdfp
- DFSMSdss
- DFSMShsm
- DFSMSrmm
- eNetwork
- ESCON
- ES/3090
- ES/9000
- GDDM
- Hiperbatch
- Hiperspace
- IBM
- IBMLink
- InfoWindow
- IMS/ESA
- Language Environment
- MVS/DFP
- MVS/ESA
- MVS/SP
- MVS/XA
- NetView
- OS/2
- OS/390
- Parallel Sysplex
- Processor Resource/Systems Manager
- PR/SM
- PSF
- Resource Link
- Resource Measurement Facility
- RETAIN
- RMF
- S/390
- SAA
- SecureWay
- SOM
- SOMobjects
- Sysplex Timer
- SystemView
- System/390
- Virtual Machine/Enterprise Systems Architecture
- VisualLift
- VM/ESA
- VTAM
- z/Architecture
- z/OS
- zSeries
- 3090

UNIX is a registered trademark of The Open Group in the United States and other countries.

NetView is a registered trademark of the International Business Machines Corporation or Tivoli Systems Inc. in the United States, other countries, or both.

Other company, product or service names may be the trademarks or service marks of others.

Glossary

This glossary defines technical terms and abbreviations used in z/OS MVS documentation. If you do not find the term you are looking for, refer to the index of the appropriate manual or view IBM Glossary of Computing Terms, located at:

<http://www.ibm.com/ibm/terminology>

A

action message retention facility (AMRF). A facility that, when active, retains all action messages except those specified by the installation in the MPFLSTxx member in effect.

action message sequence number. A decimal number assigned to action messages.

Advanced Program-to-Program Communications (APPC). A set of inter-program communication services that support cooperative transaction processing in a SNA network.

allocate. To assign a resource for use in performing a specific task.

AMRF. action message retention facility

APPC. Advanced Program-to-Program Communications

automated operations. Automated procedures to replace or simplify actions of operators in both systems and network operations.

AVR. Automatic volume recognition.

C

CART. Command and response token.

CNGRPxx. The Parmlib member that defines console groups for the system or sysplex.

command and response token (CART). A parameter on WTO, WTOR, MGCRE, and certain TSO/E commands and REXX execs that allows you to link commands and their associated message responses.

command prefix facility (CPF). An MVS facility that allows you to define and control subsystem and other command prefixes for use in a sysplex.

console. That part of a computer used for communication between the operator or user and the computer.

console group. In MVS, a group of consoles defined in CNGRPxx, each of whose members can serve as an alternate console in console or hardcopy recovery or as a console to display synchronous messages.

CONSOLxx. The Parmlib member used to define message handling, command processing, and MCS and SMCS consoles.

control unit. Synonymous with device control unit.

conversational. Pertaining to a program or a system that carries on a dialog with a terminal user, alternately accepting input and then responding to the input quickly enough for the user to maintain a train of thought.

CPF. Command prefix facility.

D

DASD. Direct access storage device.

data definition name. The name of a data definition (DD) statement, which corresponds to a data control block that contains the same name. Abbreviated as **ddname**.

data definition (DD) statement. A job control statement that describes a data set associated with a particular job step.

data set label. (1) A collection of information that describes the attributes of a data set and is normally stored on the same volume as the data set. (2) A general term for data set control blocks and tape data set labels.

deallocate. To release a resource that is assigned to a specific task.

device control unit. A hardware device that controls the reading, writing, or displaying of data at one or more input/output devices or terminals.

device number. The unique number assigned to an external device.

device type. The general name for a kind of device; for example, 3330.

direct access storage device (DASD). A device in which the access time is effectively independent of the location of the data.

display console. In MVS, an MCS or SMCS console whose input/output function you can control.

DOM. An MVS macro that removes outstanding WTORs or action messages that have been queued to a console

E

end-of-tape-marker. A marker on a magnetic tape used to indicate the end of the permissible recording area, for example, a photo-reflective strip, a transparent section of tape, or a particular bit pattern.

entry area. In MVS, the part of a console screen where operators can enter commands or command responses.

extended MCS console. In MVS, a console other than an MCS or SMCS console from which operators or programs can issue MVS commands and receive messages. An extended MCS console is defined through an OPERPARM segment.

F

full-capability console. An MCS or SMCS console that can receive messages and send commands. See *message-stream console* and *status-display console*.

H

hardcopy log. In systems with multiple console support or a graphic console, a permanent record of system activity.

hardware. Physical equipment, as opposed to the computer program or method of use; for example, mechanical, magnetic, electrical, or electronic devices. Contrast with *software*.

hardware configuration dialog. In MVS, a panel program that is part of the hardware configuration definition. The program allows an installation to define devices for MVS system configurations.

HCD. Hardware configuration definition.

I

initial program load (IPL). The initialization procedure that causes an operating system to begin operation.

instruction line. In MVS, the part of the console screen that contains messages about console control and input errors.

internal reader. A facility that transfers jobs to the job entry subsystem (JES2 or JES3).

IPL. Initial program load.

J

JES2 multi-access spool configuration. A multiple MVS system environment that consists of two or more JES2 processors sharing the same job queue and spool

K

keyword. A part of a command operand or Parmlib statement that consists of a specific character string (such as NAME= on the CONSOLE statement of CONSOLxx).

L

line number. A number associated with a line in a console screen display.

M

MAS. Multi-access spool.

master console. In an MVS system or sysplex, the main console used for communication between the operator and the system from which all MVS commands can be entered. The first active console with AUTH(MASTER) defined becomes the master console in a system or sysplex.

master console authority. In a system or sysplex, a console defined with AUTH(MASTER) other than the master console from which all MVS commands can be entered.

MCS. Multiple console support.

MCS console. A non-SNA device defined to MVS that is locally attached to an MVS system and is used to enter commands and receive messages.

message processing facility (MPF). A facility used to control message retention, suppression, and presentation.

message queue. A queue of messages that are waiting to be processed or waiting to be sent to a terminal.

message-stream console. An MCS console which receives messages but from which an operator cannot enter commands. See *full-capability console* and *status-display console*.

message text. The part of a message consisting of the actual information that is routed to a user at a terminal or to a program.

message window. The area of the console screen where messages appear.

MMS. In MVS, the MVS message service.

MPF. Message processing facility.

MPFLSTxx. The Parmlib member that controls the message processing facility for the system.

multiple console support (MCS). The operator interface in an MVS system.

multi-access spool (MAS). A complex of multiple processors running MVS/JES2 that share a common JES2 spool and JES2 checkpoint data set.

multisystem console support. Multiple console support for more than one system in a sysplex. Multisystem console support allows consoles on different systems in the sysplex to communicate with each other (send messages and receive commands)

MVS message service (MMS). An MVS component that allows an installation to display messages translated into other languages on a console or terminal.

N

NIP. Nucleus initialization program.

no-consoles condition. A condition in which the system is unable to access any full-capability console device.

nonstandard labels. Labels that do not conform to American National Standard or IBM System/370 standard label conventions.

nucleus initialization program (NIP). The stage of MVS that initializes the control program; it allows the operator to request last minute changes to certain options specified during initialization.

O

offline. Pertaining to equipment or devices not under control of the processor.

online. Pertaining to equipment or devices under control of the processor.

operations log. In MVS, the operations log is a central record of communications and system problems for each system in a sysplex.

OPERLOG. The operations log.

OPERPARM. In MVS, a segment that contains information about console attributes for extended MCS consoles running on TSO/E.

out-of-line display area. For status-display and full-capability MCS and SMCS consoles, areas of the

screen set aside for formatted, multi-line display of status information written in response to certain MVS and subsystem commands.

P

PFK. Program function key.

PFK capability. On a display console, indicates that program function keys are supported and were specified at system generation.

PFKTABxx. The Parmlib member that controls the PFK table settings for MCS consoles in a system.

printer. (1) A device that writes output data from a system on paper or other media.

program function key (PFK). A key on the keyboard of a display device that passes a signal to a program to call for a particular program operation.

program status word (PSW). A doubleword in main storage used to control the order in which instructions are executed, and to hold and indicate the status of the computing system in relation to a particular program.

pseudo-master console. A subsystem-allocatable console that has system command authority like that of an MCS master console.

PSW. Program status word.

R

remote operations. Operation of remote sites from a host system.

roll mode. The MCS and SMCS console display mode that allows messages to roll off the screen when a specified time interval elapses.

roll-deletable mode. The console display mode that allows messages to roll off the screen when a specified time interval elapses. Action messages remain at the top of the screen where operators can delete them.

routing. The assignment of the communications path by which a message will reach its destination.

routing code. A code assigned to an operator message and used to route the message to the proper console.

S

shared DASD option. An option that enables independently operating computing systems to jointly use common data residing on shared direct access storage devices.

SMCS. SNA Multiple Console Support consoles are consoles that use SecureWay Communications Server to provide communication between operators and MVS as opposed to MCS consoles, which do direct I/O to the device.

software. (1) All or part of the programs, procedures, rules, and associated documentation of a data processing system. (2) Contrast with hardware. A set of programs, procedures, and, possibly, associated documentation concerned with the operation of a data processing system. For example, compilers, library routines, manuals, circuit diagrams. Contrast with **hardware**.

status-display console. An MCS console that can receive displays of system status but from which an operator cannot enter commands. See **full-capability console** and **message-stream console**.

subsystem-allocatable console. A console managed by a subsystem like JES3 or NetView used to communicate with an MVS system.

synchronous messages. WTO or WTOR messages issued by an MVS system during certain recovery situations.

SYSLOG. The system log data set.

system log (SYSLOG). In MVS, the system log data set that includes all entries made by the WTL (write-to-log) macro as well as the hardcopy log. SYSLOG is maintained by JES in JES SPOOL space.

sysplex. A multiple-MVS system environment that allows MCS, SMCS consoles or extended MCS consoles to receive messages and send commands across systems.

system console. In MVS, a console attached to the processor controller used to initialize an MVS system.

T

terminal. A device, usually equipped with a keyboard and some kind of display, capable of sending and receiving information over a link.

terminal user. In systems with time-sharing, anyone who is eligible to log on.

U

undelivered message. An action message or WTOR that cannot be queued for delivery to the expected console. MVS delivers these messages to any console with the UD console attribute in a system or sysplex.

V

virtual telecommunications access method (VTAM).

A set of programs that maintain control of the communication between terminals and application programs running under DOS/VS, OS/VS1, and OS/VS2 operating systems.

volume. (1) That portion of a single unit of storage which is accessible to a single read/write mechanism, for example, a drum, a disk pack, or part of a disk storage module. (2) A recording medium that is mounted and demounted as a unit, for example, a reel of magnetic tape, a disk pack, a data cell.

volume serial number. A number in a volume label that is assigned when a volume is prepared for use in the system.

volume table of contents (VTOC). A table on a direct access volume that describes each data set on the volume.

VTAM. Virtual telecommunications access method.

VTOC. Volume table of contents.

W

wait state. Synonymous with waiting time.

waiting time. (1) The condition of a task that depends on one or more events in order to enter the ready condition. (2) The condition of a processing unit when all operations are suspended.

warning line. The part of the console screen that alerts the operator to conditions requiring possible action.

wrap mode. The console display mode that allows a separator line between old and new messages to move down a full screen as new messages are added. When the screen is filled and a new message is added, the separator line overlays the oldest message and the newest message appears immediately before the line.

write-to-log (WTL) message. A message sent to SYSLOG or the hardcopy log.

write-to-operator (WTO) message. A message sent to an operator console informing the operator of errors and system conditions that may need correcting.

write-to-operator-with-reply (WTOR) message. A message sent to an operator console informing the operator of errors and system conditions that may need correcting. The operator must enter a response.

WTL message. Write-to-log message

WTO message. Write-to-operator message

WTOR message. Write-to-operator-with-reply message.

Index

Special Characters

- ? (question mark)
 - specified on parameter of system command 112
- * (asterisk)
 - specified on parameter of system command 112

Numerics

- 3277-2 display station
 - in console cluster 173
 - in message stream mode 72
 - PFK (program function key) display line 71
- 3290 information panel
 - as console cluster 181
- 9021 control unit 136
- 9121 control unit 136

A

- accessibility 191
- action message
 - deletion 73
 - display of information 117
 - limiting number received at a console 182
 - retention by action message retention facility 116
- action message retention facility
 - See AMRF
- activation
 - action message retention facility 117
 - command installation exit 122
 - general message processing exit IEAVMXIT 121
- allocation of storage for a run-time message file 130
- ALLOCxx member of parmlib concatenation 144
- alternate console
 - setting up alternate console chain 175
- alternate console group
 - changing a console member 46
 - defining 44
 - defining for the hardcopy output device 92
 - definition of an extended MCS console 46
 - reassigning to a console using VARY ALTGRP 46
 - selection of a master console 50
 - used in console recovery 44
 - used in hardcopy output device recovery 92
 - using CNGRPxx to define 44
- ALTERNATE keyword 47
- ALTGRP keyword 46
- ALTGRP subkeyword of OPERPARM 7
- AMRF (action message retention facility)
 - deactivation 117
 - description 116
 - display of an action message that is not retained 114, 182
 - retrieval of an action message 117
 - when JES3 uses an MCS console 119
- AMRF keyword 116
- AOC/MVS (Automated Operations Control/MVS) 11

- APPC (Advanced-Program-to-Program Communication) 136
- AUTH keyword 53
- AUTH subkeyword of OPERPARM 8
- AUTO subkeyword of OPERPARM 9
- automatable message
 - directing to an extended MCS console 108
- automated end of WTO messages 124
- automated message
 - handling with descriptor code 13 106
- Automated Operations Control/MVS
 - See AOC/MVS
- automatic LOGON 60
- automatic message deletion 72
- automatic mode 73
- automatic mode of message deletion
 - description 73
- automatic volume recognition
 - See AVR
- automation 11
 - directing an automated message to an extended MCS console 119
 - reissuing an automated message 119
 - selecting a message 119
 - use of descriptor code 13 for a message 106
- automation in a sysplex 120
- AVR (automatic volume recognition) 144

B

- boxing a device 146
- broadcast message
 - description 106

C

- cancellation
 - automatic message deletion 73
- change
 - console alternate in a group without IPLing 46
 - MMSLSTxx member 133
 - MPFLSTxx member of SYS1.PARMLIB 115
 - time interval for dynamic display 82
- CLOCKxx and the sysplex 142
- CLOCKxx member of SYS1.PARMLIB 142
- CMD keyword 85
- CMD parameter
 - USEREXIT option 122
- CMDDELIM keyword 88
- CMDSYS keyword 109
 - using with commands in a sysplex 101
- CMDSYS subkeyword of OPERPARM 8
- CNGRP keyword 45
- CNGRPxx member of SYS1.PARMLIB
 - activating 45
 - console recovery using 44
 - definition of an alternate console group 44

- CNLccccx member 133
- command
 - flow in a sysplex 101
 - flow in an MVS system 97
 - general characteristics 95
 - groups 53
 - summary of MVS commands to change CONSOLxx values 149
- command assignment
 - MCS console 53
- command association in a sysplex 109
- command delimiter
 - definition 88
- command flooding 98
- command group 52
 - figure 53
- command installation exit 122
- command prefix
 - system name as synonym for ROUTE command IEECMDPF samplib member 111
- command prefix facility
 - See CPF
- command profile in RACF 57
- COMMNDxx member of SYS1.PARMLIB
 - using with SET CNGRP command 45
- compilation of a message file for translation 130
- CON keyword
 - in PFKTABxx 85
 - on the CONSOLE statement of CONSOLxx 77
- CON=NONE operator prompt 138
- CONFIG command
 - routed by MSGRT command 82
- CONFIG OFFLINE command
 - FORCE operand 146
- considerations
 - for using shared DASD 148
- console
 - cluster 173
 - defining a subsystem-allocatable console 39
 - definition of an MCS console 38
 - display of a synchronous message 51
 - examples of console screens 70
 - extended MCS (multiple console support) 6
 - extending the limit for a system or sysplex 6
 - MCS (multiple console support) 2
 - message roll rate 179
 - NetView 11
 - NIP (nucleus initialization program) 138
 - of last resort used in recovery 47
 - operating mode 69
 - setting up alternate console chain 175
 - switching 54
 - switching attributes 48
 - system 136
 - system console as a UD console 48
 - with UD attribute as a console of last resort 48
- console cluster
 - example 173
 - example of CONSOLxx statements used for 180
- console command assignment 53
- console command authority
 - specification of authority for a console to issue a console control command 53
 - specification of authority for a console to issue a system control command 53
 - specification of authority for a console to issue an I/O control command 53
 - specification of authority for a console to issue an informational command 53
 - specification of master authority 53
- console consolidation 13
- console control command 53
 - description 53
- console display area 80
- console function
 - control of the format of messages and status information 82
 - controlling the use of an MCS console 69
 - defining out-of-line areas on an MCS console screen 80
 - definition of the time interval for dynamic status display 82
 - deleting messages manually from an MCS console screen 77
 - message display on a full-capability console 72
 - routing status display 81
 - specification of message number 78
 - specifying automatic message deletion for an MCS console 72
 - status display 79
 - temporary suspension of the screen roll 76
 - use of SEG to delete a group of messages 79
 - ways to define 68
- console ID
 - considerations using 40
- CONSOLE keyword
 - CON keyword 77
- console name
 - advantages using 40
 - considerations when naming an alternate console group 45
 - defining for an extended MCS console 7
 - defining in a sysplex 40
 - definition of a subsystem-allocatable console 40
 - definition of an MCS console 40
 - definition of the same console to a different system 41
 - for the system console 138
 - restrictions 42
- console operating mode
 - defining for consoles in console cluster 176
 - definition 69
- console recovery
 - alternate console groups used for 44
 - considerations 43
 - console of last resort 47
 - console switching of attributes 48
 - defining alternate console groups by function 43
 - for an extended MCS console 43, 46
 - for an MCS console 43
 - in a system or sysplex 43

- console recovery (*continued*)
 - no-consoles condition 50
 - no-master-console condition 49
 - RESET CN command 49
 - role of master console 49
 - specifying for the hardcopy output device 92
 - use of an alternate console group 43
 - using SYS1.PARMLIB members for 44
 - console security 52
 - using MCS 52
 - using RACF 54
 - using the command installation exit 68
 - CONSOLE statement 15
 - ALTGRP keyword 46
 - AREA keyword 81
 - AUTH keyword 53
 - CMDSYS keyword 109
 - DEL keyword 72, 78
 - DEVNUM keyword 38
 - LEVEL keyword 106
 - MFORM keyword 83
 - MONITOR keyword 84
 - MSCOPE keyword 108
 - MSGRT keyword 81
 - NAME keyword 38
 - PFKTAB keyword 86
 - RNUM keyword 73
 - ROUTCODE keyword 105
 - RTME keyword 73
 - SEG keyword 79
 - summary of console function 15
 - UD keyword 107
 - UNIT keyword 38
 - USE keyword 71
 - UTME keyword 82
 - console switching 48
 - as the result of the SWITCH CN command 48
 - for an extended MCS console 48
 - for an MCS console 48
 - using the SWITCH CN command 48
 - when a console fails 48
 - CONSOLxx member of SYS1.PARMLIB 13
 - console recovery using 44
 - CONSOLE statement 15
 - DEFAULT statement 16
 - defining an MCS console configuration 14
 - defining in a sysplex 20
 - defining problem determination mode routing for the system console 139
 - DEL keyword
 - example 74
 - DEVNUM keyword
 - example 54
 - example of for a console cluster 180
 - example of PFK parameter 184
 - HARDCOPY statement 16, 88, 89
 - INIT statement 16
 - MFORM parameter
 - example 74
 - RTME parameter
 - example 74
 - CONSOLxx member of SYS1.PARMLIB (*continued*)
 - scope of keywords in a sysplex 20
 - specification of an alternate console group 46
 - summary of keywords 149
 - summary of MVS commands to change keyword values 149
 - system use 92
 - UEXIT parameter
 - example 121
 - control
 - format of message 82
 - JES3 through an MCS console 119
 - message deletion 73
 - message in a console cluster 173
 - messages processed by MPF 114
 - NetView through an MCS console 11
 - shared DASD 146
 - status display 79
 - WTL message 128
 - WTO and WTOR message 123
 - CONTROL access authority used by RACF 56
 - conversational mode
 - definition 77
 - message deletion 77
 - PFK (program function key) 85
 - CPF (command prefix facility) 110
 - using in a sysplex 101
 - critical eventual action message
 - description 106
 - CSVAPF macro
 - defining RACF profiles for 154
 - CSVLYLPA macro
 - defining RACF profiles for 170
 - CSVLYNEX macro
 - defining RACF profiles for 158
 - CSVLYNL macro
 - defining RACF profiles for 164
- ## D
- DASD (direct access storage device)
 - shared 146
 - data definition statement 143
 - data set status
 - display 84
 - DD (data definition statement) 143
 - DEFAULT statement 16, 18
 - HOLDMODE keyword 76
 - LOGON keyword 59
 - RMAX keyword 123
 - ROUTCODE keyword 105
 - summary of console function 19
 - SYNCHDEST keyword 51
 - definition
 - command assignment to a PFK (program function key) 181
 - command for PFK (program function key) 85
 - console configuration for a sysplex 5
 - console name 40
 - extended MCS console 6
 - MCS console 38

- definition (*continued*)
 - MCS console configuration 13
 - PFK (program function key) table 181, 184
 - routing code for consoles 104
 - routing codes for a console 175
 - same named console to a different system 41
 - single alternate console 47
 - where to send an undelivered action message 107
- DEL keyword 72, 78
- deletion
 - action message for which action has been taken 72
 - appropriate deletion specification 73
 - description of automatic message deletion 72
 - messages 72
 - non-action message 72
- descriptor code
 - descriptor code 1 106
 - descriptor code 11 106
 - descriptor code 12 106
 - descriptor code 13 106
 - descriptor code 2 106
 - descriptor code 3 106
 - during message suppression 115
 - message presentation 115
- device
 - allocation 143
 - assignment 144
 - boxing 146
 - operator considerations 146
- device allocation request
 - operator action 144
- device that MVS can use as a console 39
- DEVNUM keyword 38
 - on the CONSOLE statement 38
 - on the HARDCOPY statement 38
- devnum parameter of a system command 145
- direct access storage device
 - See DASD
- direction of a command from a console to another
 - system in a sysplex 109
- direction of a command response to a specific console
 - example in a console cluster 179
- direction of an automatable message to an extended
 - MCS console 108
- direction of messages from other systems to a console
 - in a sysplex 108
- disability 191
- display
 - action messages awaiting action 117
 - active job 82
 - data set status 84
 - job information 84
 - MSGRT routing instruction 82
 - TSO information 84
- display area
 - definition 71, 80
 - setting up for console cluster 176
- DISPLAY command
 - PFK (program function key)
 - example of output 184

- DISPLAY command (*continued*)
 - R parameter
 - examples 118
 - routed by MSGRT command 82
 - display console 39
 - display of a synchronous message 51
 - DOM subkeyword of OPERPARM 8
 - dynamic device allocation 143
 - dynamic status display
 - description 80

E

- entry area
 - definition 71
 - example 81
- error code 143
- error recovery 143
- ES/9000 control unit 9021
 - problem determination mode 139
 - system console attached 136
- ES/9000 control unit 9121
 - problem determination mode 139
 - system console attached 136
- ESCON Manager and MVS operations planning 12
- eventual action message
 - description 106
- example of defining a console configuration for a
 - sysplex environment 185
- examples and planning aids for MVS operations
 - planning 149
- examples of commands
 - CONFIG command 148
 - DISPLAY PFK 184
 - DISPLAY R 118
 - MSGRT command 82
- extended MCS console
 - console attributes 6
 - defining console attributes 7
 - definition 6
 - direction of an automatable message 108
 - example of defining 152
 - MCSOPER macro 7
 - OPERPARM segment used with 7
 - planning 6
 - providing recovery for 46
 - RACF ADDUSER command 7
 - RACF ALTUSER command 7
 - security 7
 - system console 138
 - TSO/E CONSOLE command 7
 - using the automation attribute with 9
 - using to extend the limit for a system or sysplex 6
- extended MCS consoles
 - increasing the 99 console limit 5
 - MCSOPER and OPERPARM 9

F

- flagged message
 - removed under automatic mode 73

- FORCE operand of CONFIG command
 - cautions about using 146
- FORCE operand of VARY command
 - cautions about using 146
- format of message
 - control 82
- full-capability console 69
 - definition 69
 - example of console screen 70
 - in console cluster 173

G

- general characteristics of messages and commands 95

H

- hardcopy log
 - considerations in a sysplex 98
 - definition 89
 - TRACK display 178
- hardcopy medium
 - controlling 90
 - disabling 93
- hardcopy message set
 - characteristics 89
 - controlling 90
 - definition 89
 - selection criteria
 - fixed 89
 - installation controllable 89
- hardcopy output device recovery
 - alternate console groups used for 92
- HARDCOPY statement 16, 19
 - definition 88
 - in CONSOLxx 89
 - summary of console function 19
- Hardware Configuration Definition
 - See HCD
- hardware console 136
- hardware malfunction 145
- HCD (Hardware Configuration Definition) 38
- HOLDMODE keyword 76
- hot I/O detection 145
 - operator considerations 145

I

- I/O control command 53
 - description 53
- IEA180 message 86
- IEASYSxx member of SYS1.PARMLIB 87, 137
 - CON=NONE prompt 138
 - specification of CLOCKxx and IEFSSNxx information 142
- IEASYSxx member of SYSn.IPLPARM 137
- IEAVMXIT installation exit 121
 - description 121
 - status when MPF is off 114
- IEE041I message 79

- IEE379I message 148
- IEE612I message 143
- IEE719I message 148
- IEECMDPF samplib member
 - system name as synonym for ROUTE command 111
- IEEGSYS SYS1.SAMPLIB member
 - description 110
- IEF238D message 144, 146
- IEFSSNxx member of SYS1.PARMLIB 142
- IMSI (initialization message suppression indicator) 137
- informational command 53
 - description 53
- informational message
 - description 106
 - handling with descriptor code 12 106
 - handling with descriptor code 13 106
 - sent to consoles with the UD attribute 48, 96
- INIT statement 16
 - AMRF keyword 116
 - CMDDELIM keyword 88
 - CNGRP keyword 45
 - definition 16
 - LOGLIM keyword 128
 - MLIM keyword 123
 - MMS keyword 133
 - MONITOR keyword 85
 - MPF keyword 114
 - NOCCGRP keyword 50
 - PFK (program function key) keyword 86
 - RLIM keyword 123
 - summary of console function 17
 - UEXIT keyword 121
- initialization
 - definition 136
 - from shared device 148
 - using LOADxx for 137
- initialization of an extended MCS console session 7
- initialization of the MVS system 136
- input/output definition file
 - See IODF
- installation exit routine
 - considerations in a sysplex 98
 - description of use in processing a command 122
 - description of use in processing a message 121
- instruction line
 - definition 71
 - example 81
- interaction with system function 143
- IODF (input/output definition file) 137
- IOS109I message 145
- IOS120A message 148

J

- JES2 (job entry subsystem)
 - automatic command facility 180
 - commands used in console cluster 179, 180, 182, 183
 - CONDEF initialization statement 110
 - initialization data set 180

JES2 (job entry subsystem) *(continued)*
 use of the command prefix facility 110
JES3 complex
 control through an MCS console 119
 DSP name as keyname 119
job information
 display 84

K

KEY keyword
 on the PFKTABxx member 85
KEY subkeyword of OPERPARM 8
keyboard 191

L

L= operand
 using in a sysplex 103
 using with commands 111
LEVEL keyword 106
LEVEL subkeyword of OPERPARM 8
load
 process 143
LOAD command function 137
LOAD command parameter function
 IMSI (initialization message suppression
 indicator) 137
 on system console 137
LOADxx member 136
LOADxx member of SYS1.PARMLIB 137
LOADxx member of SYSn.IPLPARM
 specifying on the system console frame 137
LOGCMDRESP subkeyword of OPERPARM 8
LOGLIM keyword 128
LOGON in conjunction with a RACF profile 59

M

management of messages and commands 95
master console
 cluster 173
 no-consoles condition 50
 no-master-console condition 49
 respecification 54
 role in recovery 49
 routing a message 105
 selecting from an alternate console group 50
 sending a message 105
 specification 53, 54
 specification in a sysplex 54
 switching assignment to another console 54
master console command 53
 description 53
MCS (multiple console support console) console
 configuration
 definition for a sysplex 5
MCS (multiple console support) command group 53
MCS (multiple console support) console
 defining 38
 definition 3

MCS (multiple console support) console *(continued)*
 definition of a console name 40
 devices MVS can use 39
MCS (multiple console support) console authority 52
MCS console
 extending the limit for a system or sysplex 6
MCS console cluster
 definition 13
 example 173
MCS console configuration 13
 defining in a multisystem environment 13
 use of an MCS console cluster 13
 using CONSOLxx 14
MCSOPER macro 7, 49
message
 control of status display 79
 control of the format 82
 deletion 72
 description of automatic message deletion 72
 descriptor code 13 106
 flow in a sysplex 98
 flow in an MVS system 97
 general characteristics 95
 processed by IEAVMXIT 121
 processed by MPF 114
 processed by user exit 121
 reissuing an automated message 106
 routing 104
 selecting for automation 119
 suppressed by MPF 115
 synchronous 51
message and command flow
 in a sysplex 98
 in a system 97
message area
 definition 71
 example 81
message deletion specification 72
 definition 73
message format
 changing 82
message level
 assigning to a console 106, 176
 relationship with routing code 107
 switching for a console 48
MESSAGE macro in JES3 118
message number
 aid in manual message deletion 79
 display of consecutive numbers 79
 stopped by automatic message deletion 79
 when not recommended 79
message presentation
 specifying 115
message roll rate
 specification 179
 temporary suspension 76
message routing 104
message routing code 104
 definition 105
 for an extended MCS console 105
 for messages without any defined 105

- message scope
 - defining in a sysplex 108
 - switching for a console 48
- message stream console
 - definition 70
 - in console cluster 173
- message suppression
 - for automation 115
- message translation 128
 - creating a VSAM linear data set 130
 - run-time message file 130
 - using the MVS message compiler 129
- message translation service
 - See MMS
- message-stream console
 - example of console screen 70
- messages
 - commands 95
 - managing system commands 95
 - managing system messages 95
- MFORM keyword 83
- MFORM subkeyword of OPERPARM 8
- MIGID subkeyword of OPERPARM 8
- MLIM keyword 123
- MMS (MVS message service) 128
 - installation exit used to control 132
 - using SYS1.PARMLIB to control 132
- MMS keyword 133
 - on the INIT statement of CONSOLxx 133
- MMSLSTxx member 132
 - activating 133
- MONITOR keyword
 - on the CONSOLE statement of CONSOLxx 84
 - on the INIT statement of CONSOLxx 85
- MONITOR subkeyword of OPERPARM 8
- mount
 - characteristics 147
 - characteristics for shared DASD 147
 - message 85
- MOUNT command 144, 145, 147
 - data definition statement 143
- mounting a volume
 - operator considerations 145
- MPF (message processing facility)
 - changing messages suppressed 114
 - description 114
 - displaying MPF options in effect 114
 - ending message suppression 114
 - messages unable to be suppressed by 115
 - planning for MVS 114
 - retention of a message 117
 - selecting a message for automation 119
 - specification of automation for a message 119
 - specification of command installation exit 122
 - specification of message presentation 115
 - specification of message processing exit 122
 - through MPFLSTxx 115
 - use of NetView 11
- MPF keyword 114
- MPFLSTxx member of SYS1.PARMLIB
 - CMD parameter
 - USEREXIT option 122
 - MSGCOLR statement 115, 116
 - msgid parameter
 - AUTO option 119
 - RETAIN option 117
 - SUP option 116
 - USEREXIT option 122
- MPFLSTxx member of SYS1.PARMLIB of SYS1.PARMLIB
 - changing 115
- MSCOPE keyword 108
- MSCOPE subkeyword of OPERPARM 8
- MSGCOLR statement 115, 116
- msgid parameter 119
 - AUTO option 119
 - RETAIN option 117
 - SUP option 116
 - USEREXIT option 122
- MSGRT command
 - determination of current routing instruction 82
 - examples 82
- MSGRT keyword 81
- multiple console support 2
- MVS command profile summary 62
- MVS message compiler 129
 - invoking 131
- MVS operations planning
 - automation 11
 - console function 68
 - examples and planning aids 149
 - recovery 43
 - remote operation 11
 - security 52
- MVS single system
 - consolidation of consoles 13
 - use of an MCS console cluster 13
- MVS system environment
 - defining a console configuration for 13
 - maximum number of MCS consoles defined for 5
- MVS.MCSOPER.console-name profile in OPERCMDS class
 - defining 153
- MVS.UNKNOWN profile 68

N

- NAME keyword 38
- NetView 11
 - consolidating consoles using 13
 - use of MPF to suppress a message 11
- NetView console 11
- NIP console
 - as master console 138
 - definition 137
- nn on the console warning line 71
- no-consoles condition 50
 - selecting a master console from an alternate console group 50
 - subsystem console 51

- no-master-console condition 49
- NOCCGRP keyword 50
- nonconversational mode
 - definition 77
 - message deletion 77
 - PFK (program function key) 85
- Notices 193
- number of messages waiting to be displayed 71

O

- OLTEP (online test executive program) 144
- online test executive program
 - See OLTEP
- OPC/ESA (Operations Planning and Control/ESA) 11
- operating environment
 - AOC/MVS 11
 - APPC (Advanced Program-to-Program Communication) 136
 - considerations 2
 - example of console configuration 4
 - in a sysplex 5
 - multiple console support 2
 - NetView 11
 - OPC/ESA 11
 - remote operation 11
 - RMF (Resource Measurement Facility) 10
 - SDSF (System Display and Search Facility) 10
 - single system 2
- operations goals
 - control of operating activity and function 1
 - increasing system availability 1
 - simplifying the task of the operator 1
 - single point of control 2
 - single system image 2
 - streamlining message flow and command processing 1
- operations log
 - purpose 91
- Operations Planning and Control/ESA
 - See OPC/ESA
- operator information area
 - definition 71
 - example 81
- operator reply
 - control using RLIM and RMAX value 124
- OPERLOG (operations log) 91
- OPERPARM segment 7
- optional LOGON 61
- out-of-line display area 80
- output-only console 39
 - definition 70

P

- PFK (program function key)
 - defining commands for 181
 - definition of command 85
 - in conversational mode 85
 - summary of keys for console cluster 184

- PFK (program function key) display line
 - definition 71
- PFK (program function key) keyword
 - on the PFKTABxx member 85
- PFK (program function key) table
 - definition 85
 - example of defining command 87
- PFK keyword 86
 - on the INIT statement in CONSOLxx 86
- PFK table
 - example of defining 184
 - example of defining for a console in a console cluster 181
- PFKTAB keyword 86
- PFKTAB parameter
 - example 184
- PFKTABxx member of SYS1.PARMLIB
 - CMD keyword 85
 - CON keyword 85
 - description 85
 - example of coding 87
 - example of defining for a console cluster 181
 - example of in a console cluster 183
 - KEY keyword 85
 - PFK (program function key) keyword 85
 - PFK keyword 85
 - TABLE keyword 85
- planning
 - AOC/MVS 11
 - basic operator procedure 135
 - considerations for using consoles in a sysplex 20
 - console function 68
 - console recovery 43
 - for extended MCS console use 6
 - MCS console configuration 13
 - NetView and MVS operation 11
 - OPC/ESA 11
 - operations goals 1
 - remote operation 11
 - RMF and MVS operation 10
 - SDSF and MVS operation 10
- printing the system log 92
- problem determination mode 139
 - changing console attributes for the system console 140
 - CONSOLxx and routing values 139
 - messages received on the system console when in 140
 - messages received when in problem determination mode 139
 - using VARY, CONTROL, or MONITOR commands when in 140
- profile
 - RACF (Resource Access Control Facility) 55
- program status word
 - See PSW
- PSW (program status word) 143

R

RACF (Resource Access Control Facility)
console security 54
definition of resource profile for command 57
definition of resource profile for console 59
definition of user profile for console 57
definition of user profile for operator 57
LOGON requirements using 59
summary of the MVS command profile 62
using to define an extended MCS console 152
using with CSVAPF macro 154
using with CSVDYLPA macro 170
using with CSVDYNEX macro 158
using with CSVDYNL macro 164
using with SET PROG command 154, 158, 164, 170
using with SETPROG command 154, 158, 164, 170
RACF access authority used with command profile 56
RACF ADDUSER command
used with extended MCS console 7
RACF ALTUSER command
used with extended MCS console 7
RACF profile 55
authority 56
resource 56
READ access authority used by RACF 56
recovery
boxing 146
error 143
hot I/O 145
remote operation
TSCF (Target System Control Facility) 11
removal of message from the screen 72
reply
WAIT for shared DASD 148
required LOGON 61
reserve/release 146
RESET CN command 49
resource allocation 143
Resource Measurement Facility
See RMF
respecification of the master console 54
retention of an action message 116
RLIM keyword 123
RMAX keyword 123
RMF (Resource Measurement Facility) 10
RNUM keyword 73
roll mode 73
for consoles in a console cluster 179
roll mode of message deletion
for consoles in a console cluster 182
roll-deletable mode 73
for consoles in a console cluster 179
roll-deletable mode of message deletion
defining 74
ROUTE CODE keyword
on the CONSOLE statement of CONSOLxx 105
on the DEFAULT statement of CONSOLxx 105
ROUTE CODE subkeyword of OPERPARM 8
ROUTE command
description 109

ROUTE command (*continued*)
using in a sysplex 102
routing
command 109
message 105
message by routing code 104
routing code
assigning to a console 105, 175
description 105
handling messages without 105
relationship with message level 107
switching for a console 48
RTME keyword 73
run-time message file
creating a VSAM linear data set for 130

S

scope of CONSOLxx keywords in a sysplex
examples for a console configuration 23
summary 21
scratch volume 144, 145
screen format
example 81
SDSF (System Display and Search Facility) 10
security
planning console 52
SEG keyword 79
selection of a master console from an alternate console
group 50
selector pen
defining commands for 86
deleting messages using 77
SET CNGRP command 45
SET PROG command
example of defining using RACF 154, 158, 164, 170
SETPROG command
example of defining using RACF 154, 158, 164, 170
shared DASD
operating guidelines 148
operator considerations 147
option 146
sharing system commands
in COMMNDxx 113
in multiple systems 113
overview 111
planning 112
shortcut keys 191
SMCS
consoles in a system or sysplex 5
specification
console operating mode 176
message level 106, 176
new master console 54
PFK (program function key) 181
routing code 175
shared DASD mount characteristics 147
TRACK display in console cluster 177
specification of automatic message deletion 73

- specification of color
 - through an installation exit 115
 - through MPFLSTxx 115
- specification of highlighting
 - through an installation exit 115
 - through MPFLSTxx 115
- specification of intensity
 - through an installation exit 115
 - through MPFLSTxx 115
- specification of the master console 53
- specification of the master console using CONSOLxx 54
- START command
 - DEALLOC procedure 147
- started cataloged procedure
 - data definition statement 143
- status display
 - control 79
 - description 79
- status display console
 - definition 70
 - example of console screen 70
 - in console cluster 173
- STORAGE subkeyword of OPERPARM 8
- subsystem processing
 - considerations in a sysplex 98
- subsystem-allocatable console 13
 - using names for 40
- summary of CONSOLxx keywords 149
- summary of MVS message and command processing 134
- suppression
 - job and system name for message 83
 - message from a console 115
- suspension
 - message roll rate 76
- SWITCH CN command 48
- symbolic group name 145
- SYNCHDEST keyword 51
- synchronous message
 - considerations in a sysplex 51
 - console interruption 52
 - displaying 51
 - issued on a 3270-X device 39
 - issued on an MCS console 51
 - issued on the master console 51
 - issued on the system console 51
- SYS1.PARMLIB 14
 - CONSOLxx member
 - HARDCOPY statement 89
- SYSLOG (system log) 92, 93
- sysplex 98
 - automated message 108
 - automation in 120
 - command association for a system 109
 - direction of a console command to another system 109
 - direction of messages from other systems to a console 108
 - example of CONSOLxx keyword scope for an MCS console 23

- sysplex (*continued*)
 - example of defining a console configuration 185
 - maximum number of MCS consoles defined for 5
 - no-consoles condition 49, 50
 - operating environment 5
 - scope of CONSOLxx keywords 20
 - specification of the master console 54
 - use of a console name 40
 - use of CONSOLxx 20
 - use of the extended MCS console 6
 - use of the MCS console 14
- sysplex operating environment 5
- system
 - activity recording 90
 - monitoring 143
- system command group 52
- system console
 - activating problem determination mode 139
 - changing console attributes when in problem determination mode 140
 - defining in CONSOLxx 138
 - defining using DEVNUM on the CONSOLE statement 39
 - definition 136
 - example of console in problem determination mode 140
 - example of in an MCS console configuration 4
 - in an MVS console configuration 4
 - in problem determination mode 139
 - LOAD command parameter function 137
 - messages received during normal operations 139
 - messages received during problem determination mode 140
 - naming 138
 - naming restrictions 138
 - using CONSOLxx to define problem determination mode routing values 139
 - using VARY, CONTROL, or MONITOR commands for problem determination mode 140
- system control command 53
 - description 53
- System Display and Search Facility
 - See SDSF
- system group name
 - setting up for ROUTE command use 110
- system log
 - purpose 92
- system name
 - synonym for ROUTE command 111
- system symbols
 - using in commands 112

T

- TABLE keyword
 - on the PFKTABxx member 85
- tape
 - blank 145
 - labeled 145
 - unlabeled 145
- tapemark 145

- Target System Control Facility
 - See TSCF
- temporary suspension of the screen roll using HOLDMODE 76
- time interval for dynamic display
 - changing 82
- time stamp
 - adding to message 84
- TRACK command
 - routed by MSGRT command 82
 - setting up TRACK display 177
- translation
 - handling messages 128
- treatment of message for translation 128
- TSCF (Target System Control Facility) 12
- TSO SUBMIT command 143
- TSO/E (time sharing option/extended)
 - display of information 84
- TSO/E CONSOLE command
 - initialization of an extended MCS console session using 7
- TSO/E LOGON command
 - data definition statement 143

U

- UD keyword
 - on the CONSOLE statement 107
- UD subkeyword of OPERPARM 8
- UEXIT keyword
 - on the INIT statement of CONSOLxx 121
- undelivered action message
 - sent to consoles with the UD attribute 48
 - where to send 107
- undelivered WTOR message
 - sent to consoles with the UD attribute 48
- UNIT keyword 38
- unlabeled tape 145
- UPDATE access authority used by RACF 56
- use
 - console cluster 173
 - X option of MFORM 83
- USE attribute of VATLSTxx 144
- USE keyword 71
- use of location operand to route status display 82
- UTME keyword 82

V

- VARY ALTGRP command 46
- VARY CN command
 - examples 54
- VARY CN,ACTIVATE command 139
- VARY CN,DEACTIVATE command 139
- VARY CN(*),ACTIVATE 139
- VARY CN(*),DEACTIVATE 139
- VARY command 146, 147, 148
- VARY HARDCPY command
 - using to modify hardcopy message set 89
- VATLSTxx member
 - at IPL time 144

- VATLSTxx member *(continued)*
 - controlling mounting of volumes 144
 - when the VARY command is issued 144
- volume
 - characteristics 147
 - mounting 144, 147

W

- wait state
 - restartable 145
- warning line
 - definition 71
 - example 81
 - nn 71
- wildcards
 - using in commands 112
- wrap mode 74
- WRITELOG command
 - to force printing of SYSLOG 92
- WTL message
 - specifying buffers through CONSOLxx 128
- WTO message
 - controlled by user exit IEAVMXIT 121
 - specifying buffers through CONSOLxx 123
- WTO messages
 - controlling automated end 124
- WTOR message
 - controlled by user exit IEAVMXIT 121
 - description 121
 - specifying buffers through CONSOLxx 123

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

z/OS
MVS Planning: Operations

Publication No. SA22-7601-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<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"/>				
Complete	<input type="checkbox"/>				
Easy to find	<input type="checkbox"/>				
Easy to understand	<input type="checkbox"/>				
Well organized	<input type="checkbox"/>				
Applicable to your tasks	<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



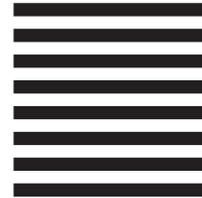
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

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, NY
12601-5400



Fold and Tape

Please do not staple

Fold and Tape



Program Number: 5694-A01



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SA22-7601-02

