z/OS
Version 2 Release 4

MVS Planning: Operations

IBM

SA23-1390-40

Note

Before using this information and the product it supports, read the information in "Notices" on page 205.

This edition applies to Version 2 Release 4 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2020-04-28

Contents

Figures

Tables

About this document

This document contains planning information for MVS™ operations. It describes how to define and use multiple console support (MCS) consoles, SNA multiple console support (SMCS) consoles, HMC multiple console support (HMCS) consoles, and extended multiple console support (EMCS) consoles. It also describes how to manage messages and commands in an MVS single-system or sysplex environment.

Who should use this document

System programmers who plan MVS operations and persons who administer the security procedures for their installations should use this document. The document 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 document

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

The document is organized as follows:

- **Chapter 1, "Planning MVS operations," on page 1** describes setting operations goals for an MVS environment. It provides a brief introduction to MCS consoles, SMCS consoles, HMCS consoles, and EMCS consoles.
- **Chapter 2, "Defining 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 member of parmlib. It also provides information to plan for console recovery, console security, and system logging.
- **Chapter 3, "Managing messages and commands," on page 85** 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, "Message flooding," on page 121** describes how to use Message Flood Automation to handle message flooding situations. It describes how to set up message flooding policy in the MSGFLDxx member of parmlib.
- **Chapter 5, "Defining auto-reply policy for WTORs," on page 133** describes how to use the auto-reply policy for WTORs.
- **Chapter 6, "Planning for operation tasks," on page 137** describes how to plan for MVS operator tasks like initializing a system and operating MVS on a day-to-day basis.
- **Chapter 7, "Examples and MVS planning aids for operations," on page 153** 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.
- **Appendix A, "AUTOR00 parmlib member," on page 189** contains the contents of the parmlib member AUTOR00.
- **Appendix B, "Accessibility," on page 201** describes the major accessibility features in z/OS.

The glossary defines technical terms used in this document.

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 Internet Library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

Conventions and terminology used in this information

When this information refers to RACF® (Resource Access Control Facility) it is the IBM® security management product for its large server z/OS® operating system. You can substitute your security product in place of RACF if you are not using RACF.

How to send your comments to IBM

We invite you to submit comments about the z/OS product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

Important: If your comment regards a technical question or problem, see instead "If you have a technical problem" on page xv.

Submit your feedback by using the appropriate method for your type of comment or question:

Feedback on z/OS function
If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

Feedback on IBM Knowledge Center function
If your comment or question is about the IBM Knowledge Center functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Knowledge Center Support at ibmkc@us.ibm.com.

Feedback on the z/OS product documentation and content
If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: z/OS MVS Planning: Operations, SA23-1390-40
- The section title of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive authority to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of changes

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

Summary of changes for z/OS Version 2 Release 4 (V2R4)

The following changes were made to this information in z/OS Version 2 Release 4 (V2R4).

New

- For APAR OA58786, "Allowing a TSO/E user to issue the CONSOLE command" on page 160 is updated to allow the use of the OPERPARM keyword for the CONSOLE command .

Summary of changes in z/OS Version 2 Release 3 (V2R3)

The following changes were made to this information in z/OS Version 2 Release 3 (V2R3).

New

- For APAR OA54790, "Choosing how to define your console configuration" on page 15 is updated about password phrases.
- Various sections were changed that reflect the removal of shared mode and distributed mode.

Summary of changes in z/OS Version 2 Release 2 (V2R2)

The following changes were made to this information in z/OS Version 2 Release 2 (V2R2).

New

- MVS.MULTIPLE.LOGON.CHECK is a new security profile that is defined in the OPERCMDS class. When MVS.MULTIPLE.LOGON.CHECK is specified, an operator may log on to an unlimited number of consoles concurrently within a system or sysplex. For more information, see "Choosing how to define your console configuration" on page 15.
- TIMEOUT is a new CONSOLE statement keyword. When TIMEOUT is specified, the console will automatically log off after the specified number of minutes of inactivity is reached. For more information, see "Automatic Timeout Processing" on page 62.
- SUPSBY is a new CONSOLE statement keyword. When SUPSBY is specified, the designated console supports standby mode in addition to active mode and inactive mode. For more information, see "Utilizing standby mode" on page 46.
- Various sections were changed to document the new functional differentiation between shared mode and distributed mode.

Chapter 1. Planning 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 printer pool 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.

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 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 5.)

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 z/OS Communications Server to provide communication between operators and MVS instead of direct I/O to the console device.

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. Important messages that require action can be directed to an 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 153.

Figure 1 on page 3 shows a console configuration for an MVS system that also includes the system console, an SMCS console, NetView, and TSO/E.

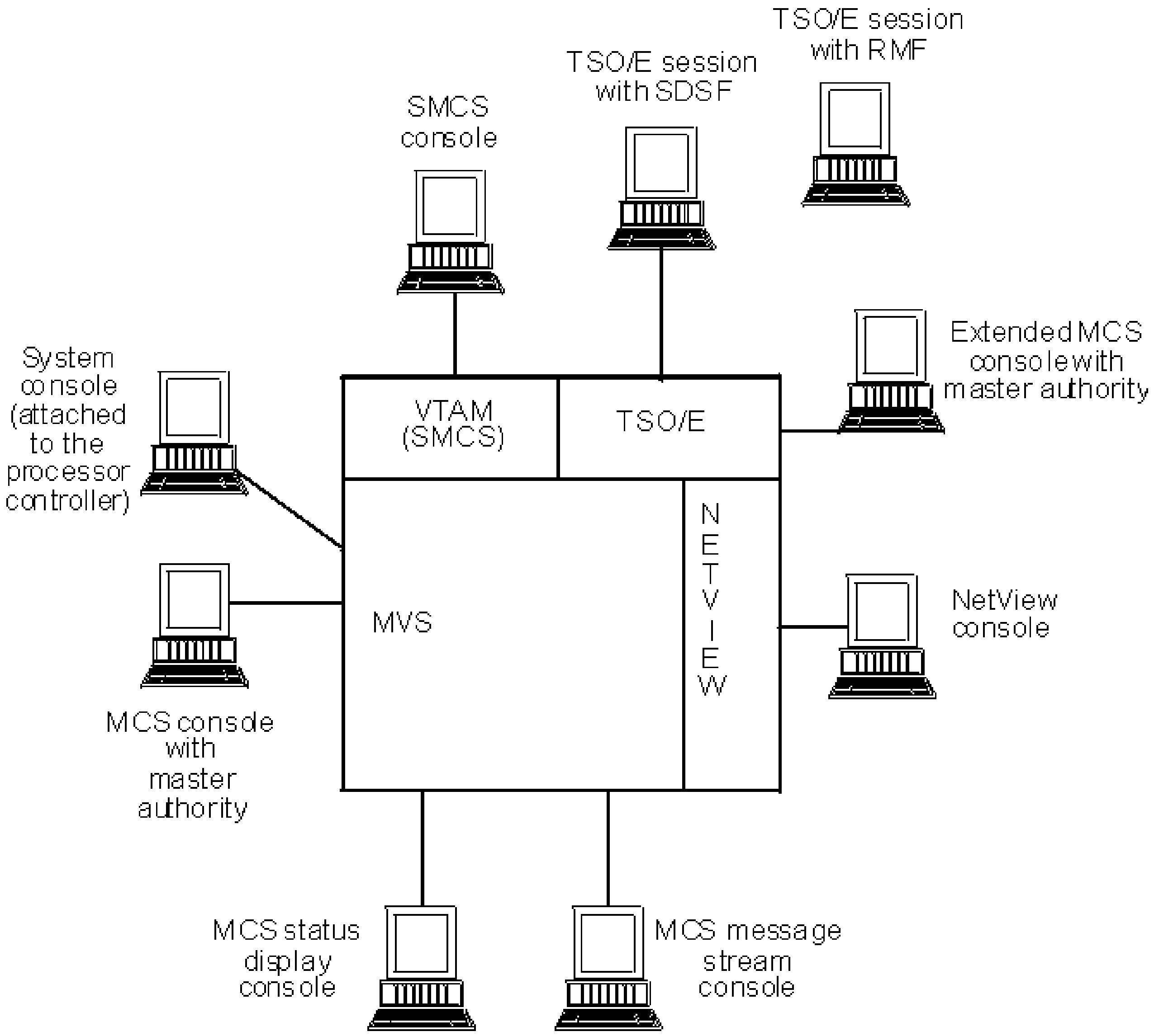

TSO/E session with RMF
TSO/E session with SDSF
SMCS console
Extended MCS console with master authority
System console (attached to the processor controller)
VTAM (SMCS)
TSO/E
NETVIEW
MVS
NetView console
MCS console with master authority
MCS status display console
MCS message stream console

Figure 1. Console Configuration for an MVS System

The system console function is provided as part of the Hardware Management Console (HMC). 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 on page 3 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 10.

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 8. For information about MVS operations and RMF, see "RMF and MVS operations planning" on page 9.

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 5.

The MCS consoles in Figure 1 on page 3 include the following:

- An MCS console with master authority 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 console with master authority.
- An MCS status display console. An operator can view system status information from DEVSERV, DISPLAY, 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 on page 4 shows a two-system sysplex, with three consoles attached:

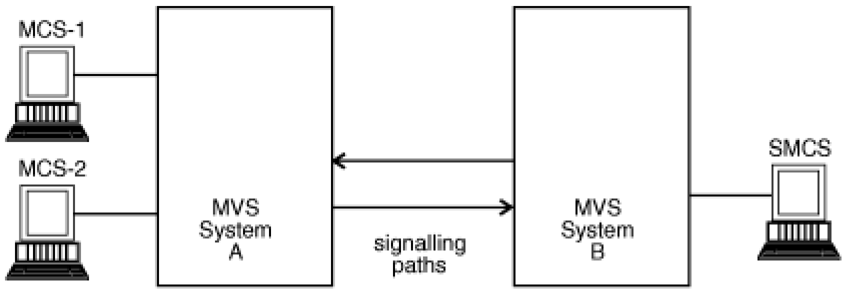


Figure 2. Sysplex Showing Console attachments

In Figure 2 on page 4, 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.

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 7, "Examples and MVS planning aids for operations," on page 153 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, HMCS and SMCS consoles in a system or sysplex

You can define up to 250 consoles including any subsystem-allocatable consoles for an MVS system. Of these, you can have up to 99 consoles concurrently active on an MVS system. In a sysplex, the limit is determined by the 99 console limit on each MVS system multiplied by the number of MVS systems in the sysplex. You can exceed this number in a system or sysplex by using extended MCS consoles. (See "Extended MCS consoles" on page 5.) Therefore, you should examine any product that uses subsystem-allocatable consoles to determine if it could use extended MCS consoles instead.

Subsystem allocatable consoles are defined in CONSOLxx and obtained and released using the IEAVG700 interface. Programs invoke IEAVG700 passing in the SCSR (subsystem console service routine) parmlist, which is mapped by IEZVG100. IBM highly recommends the use of extended MCS (EMCS) consoles rather than subsystem allocatable consoles.

There is no requirement to have an MCS, HMCS or SMCS console configured to each system. You can use command and message routing capabilities on one MCS, HMCS or SMCS console to control multiple systems in the sysplex. MCS, HMCS or SMCS consoles are not needed on all systems; but you should have at least one MCS, HMCS or SMCS console capable of operating the sysplex.

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 only console to receive synchronous messages, for example.

If you have only SMCS consoles, the hardware management console must be used in place of a NIP console. Consider creating an AUTOACT console group for the system console to provide unbroken communication from NIP to the activation of your SMCS consoles.

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 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.

In an application program, you can 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, the routing of messages and commands, and the format display of messages. "Defining console attributes for extended MCS consoles" on page 6 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 159 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 on page 6 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
Command authority for the console	AUTH	AUTH(INFO)	Only informational commands can be issued.
Routing codes for the console	ROUTCODE See Note 3	ROUTCODE(NONE)	No routing codes established for the console.
Levels of messages directed to the console	LEVEL See Note 3	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 See Notes 2 and 3.	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.

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

Console Attribute	**RACF OPERPARM Subkeyword**	**Default value**	**Meaning of Default**
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 76.
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).
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 See Note 4	DOM(NORMAL)	(NORMAL) See Note 5. (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 1	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 messages eligible for automation	AUTO See Notes 2 and 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.)
Receive messages directed to console id zero	INTIDS	N	Whether the console receives messages directed to console id zero.
Receive messages directed to unknown console ids	UNKNIDS	N	Whether the console receives messages directed to unknown console ids, such as one-byte id.
Receives the hardcopy message set	HC	N	Whether the extended console receives the hardcopy message set

Note:

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 messages to no longer be displayed on a console. Messages that are not displayed on a console will still be logged in SYSLOG and/or OPERLOG, and are viewable using facilities such as SDSF.

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

   ```
   VARY CN
   VARY CONSOLE
   CONTROL V,LEVEL
   LOGOFF (for SMCS consoles)
   ```
4. If the MCSOPER ACTIVATE request specified MSGDLVRY=NONE, the attribute specified or defaulted for DOM will be forced to DOM=NONE.
5. EMCS consoles defined with this attribute will not receive DOMs unless AMRF is active.

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 6. 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 and JES3 for processing, including:
  - Jobs on the input queue, output queue (JES2 only), and held queue (JES2 only)
  - 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 JES system without using JES command syntax.
- Enter MVS, JES2, and JES3 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.

- View output from the MVS Health Checker.
- 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 large enterprise 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*.

IBM OMEGAMON z/OS Management Console and MVS operations planning

The IBM OMEGAMON® z/OS Management Console (zMC) is a free offering from Tivoli® that uses the Tivoli Enterprise Portal (TEP) graphical user interface (GUI) [1] to provide an easy to use display of z/OS job and process availability information. Information about Coupling Facility connections and structures is also provided. Output from the z/OS Health Checker may also be viewed through the zMC.

The zMC comes pre-configured with a set of tabular and graphical displays (known as workspaces) and with a set of pre-configured *situations*. Situations are simple rules that can be established to look for conditions within the data being displayed and take some action, such as changing the color of the data which can visually alert operations personnel to various anomalous conditions. The TEP GUI is extremely flexible and may be easily customized to display the information in the manner desired. For example, within tabular displays, columns may be re-ordered left-to-right, and the data within a column may be sorted. Any of the data that is being displayed in a table may be displayed graphically, in a variety of formats. All of the pre-configured situations are easily modified and new situations may be easily created.

The zMC workspaces are designed to be easily integrated with other workspaces in the OMEGAMON XE for z/OS product if it is present.

For more information about the z/OS Management Console, see *IBM OMEGAMON for z/OS Management Console User's Guide*.

Tivoli OMEGAMON and MVS operations planning

The OMEGAMON product suite provides performance and availability monitoring for z/OS and for various products that run on z/OS such as CICS, IMS and DB2®. OMEGAMON XE for z/OS uses the Tivoli Enterprise Portal (TEP) graphical user interface (GUI) to present the performance and availability information that it has gathered. When multiple products that use the TEP are present, the dynamic workspace linking

[1] The Tivoli Enterprise Portal infrastructure is used by the Tivoli OMEGAMON XE product suite and by other Tivoli products such as Tivoli NetView for z/OS, Tivoli System Automation for z/OS and Tivoli Workload Scheduler for z/OS.

capabilities of the TEP make it possible to follow a trail that begins as a symptom in one product – for example with a communications problem reported by NetView – and move in-context from NetView's description of the problem, to CICS's description of the problem, to DB2's description of the problem, to z/OS's description of the problem, and so on.

For more information about OMEGAMON and the OMEGAMON suite of monitoring products, see *IBM Tivoli OMEGAMON XE on z/OS: 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.

Important: If automation is used to remotely IPL z/OS, the HMCS console (the "Integrated 3270 Console" located on the Hardware Management Console) ***must be disconnected*** from the z/OS partition that is to be IPLed, ***before*** the IPL is initiated. Failure to do so will prevent automation from receiving messages issued during the IPL and prevent automation from taking any appropriate actions.

Tivoli NetView for z/OS

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 100.)

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.

See *NetView Automation: Planning* for more information about how to coordinate activities for your MCS or SMCS consoles, NetView, and MPF.

Tivoli System automation for z/OS

System Automation for z/OS is a NetView application that automates console operations in a z/OS environment. System Automation for z/OS uses the message handling capabilities of MVS and NetView to initiate automation procedures. These automation procedures perform goal-oriented operator functions that manage MVS, JES2 or JES3, and program products like Websphere, DB2, CICS, IMS and Tivoli Workload Scheduler.

See *IBM Tivoli System Automation for z/OS User's Guide* for more information about how to plan console automation using System Automation and NetView.

Tivoli workload scheduler for z/OS

NetView and System Automation for z/OS can help you plan automated operations for z/OS 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 Tivoli Workload Scheduler (formerly the Operations Planning and Control (OPC) product) 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.

See *Tivoli Workload Scheduler for z/OS* for more information about how to plan the automation of your production workload processing using Tivoli Workload Scheduler for z/OS.

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 Processor Operations component of the System Automation for z/OS product uses NetView to allow a host z/OS system to automate operations at target systems. Using System Automation for z/OS, you can automate console functions remotely, like IPLing or power-on restarting a processor. You can initialize or monitor target systems and let NetView operators manage several target systems simultaneously from a host system.

Important: If automation is used to remotely IPL z/OS, the HMCS console (the "Integrated 3270 Console" located on the Hardware Management Console) ***must be disconnected*** from the z/OS partition that is to be IPLed, ***before*** the IPL is initiated. Failure to do so will prevent automation from receiving messages issued during the IPL and prevent automation from taking any appropriate actions.

In multisystem environments where remote operations is a goal, System Automation for z/OS and NetView provide a good way to manage operations. See *IBM Tivoli System Automation for z/OS User's Guide* for more information about the planning tasks for managing remote operations of systems using System Automation for z/OS and NetView.

ESCON, FICON, and operations planning

The I/O Operations component of the System Automation for z/OS product manages configuration changes among channels, ESCON Directors, control units, and devices. System Automation for z/OS can be used to control and display the entire I/O configuration, whether it be ESCON, FICON®, or non-ESCON or switched (via ESCON or FICON Directors) or non-switched. System Automation for z/OS 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 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, FICON, 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.

For more information, see *IBM Tivoli System Automation for z/OS User's Guide*

Reference

IBM Tivoli System Automation for z/OS User's Guide.

Chapter 2. Defining 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.

Console considerations for z/OS V1R8 and higher

In z/OS V1R8, the master console has been removed. The value of the master console has decreased over time. Historically, the master console had been used for important messages that required the highest level of authority to take action.

Table 2 on page 13 details the functions that were unique to the master console that are now available to other consoles.

Table 2. Functions that were unique to the master console that are now available to other consoles

Functions	z/OS V1R7 and lower releases	z/OS V1R8 and higher releases
Routing codes 1 and 2	The master console was forced to have routing codes 1 and 2 assigned to it and they could not be removed.	Routing codes 1 and 2 are no longer forced to any console. Make sure routing codes 1 and 2 are assigned to the appropriate console definitions in CONSOLxx.
Console id 0 (internal console ID (X'00000000'))	All messages targeted for console id 0 were delivered to the master console.	All messages targeted for console id 0 will go to any console that has the INTIDS routing attribute. INTIDS may be specified in CONSOLxx or through the VARY CN command.

Table 3 on page 13 details the functions that affected the master console that have been changed.

Table 3. Functions that affected the master console that have been changed

Functions	z/OS V1R7 and lower releases	z/OS V1R8 and higher releases
External interrupt key	The external interrupt key was used to switch the master console function to another console.	Since there is no longer a master console the external interrupt key on the hardware management console is not supported.
SYNCHDEST	The master console (*MSTCON*) could be specified in the SYNCHDEST group as a destination for synchronous messages.	The master console (*MSTCON*) is ignored. You need to explicitly specify by name any consoles you expect to receive synchronous messages. The system console remains the synchronous message destination of last resort.

Table 3. Functions that affected the master console that have been changed (continued)

Functions	z/OS V1R7 and lower releases	z/OS V1R8 and higher releases
Synchronous message destination default (no SYNCHDEST group defined)	The master console was used if available on the system where the message was issued, otherwise the system console on that system was used.	The system console on the system where the message was issued is used.

Table 4 on page 14 details the remaining functions that were affiliated with the master console and removed.

Table 4. Remaining functions that were affiliated with the master console and removed

Functions	z/OS V1R7 and lower releases	z/OS V1R8 and higher releases
Console switch and the SWITCH CN command	Console switch and/or the SWITCH CN command was used to switch a failing console to an alternate console.	The main purpose was to ensure the availability of the master console. Since the master console has been removed the need for console switch and SWITCH CN command has also been removed. This reduces the complexity of console definitions and the difficulties of finding the location of the master console after a console switch.
Hardcopy switch	Depending on your configuration of OPERLOG and SYSLOG, when one failed, hardcopy could automatically switch to the other.	There is no switching between hardcopy mediums, therefore to maintain the same level of functionality, run with both SYSLOG and OPERLOG as your hardcopy medium.
ALTGRP (alternate group)	The ALTGRP function was used in selecting an alternate console.	There is no longer a need for the ALTGRP (alternate group) function because of the elimination of console switch. **Note:** The Parmlib(CNGRPxx) member is still used for AUTOACT and SYNCHDEST. In V1R8 and higher the ALTGRP(x) keywords need to be removed from the CONSOLxx parmlib member.
NOCC and NOMCC	NOCC (no consoles condition) and NOMCC (no master console condition) were considered to be undesirable conditions. NOCCGRP specification in CONSOLxx specifies the name of the console group defined in CNGRPxx from whose members the system or sysplex can select a master console during a no consoles condition.	NOCC and NOMCC are no longer considered undesirable conditions and can be considered an acceptable running console environment. The specification of NOCCGRP is ignored.

Table 5 on page 15 details the functions changed to help run your sysplex.

Table 5. Changed function to help you run your sysplex.

Functions	z/OS V1R7 and lower releases	z/OS V1R8 and higher releases
SYSCONS (system console or hardware management console)	SYSCONS authority was installation defined.	SYSCONS always has master authority to insure there is always a master authority console available.

In V1R7 the external use of 1 byte console IDs was removed. If you compiled a program that used 1 byte console IDs, compilation would fail. But existing compiled programs that used 1 byte console IDs would continue to execute as before.

In z/OS V1R8 and higher releases, 1-byte console IDs (and EMCS migration ids) are eliminated. All messages targeted for 1 byte console IDs will go to any console that has the UNKNIDS routing attribute. UNKNIDS may be specified in CONSOLxx or through the VARY CN command.

Choosing how to define your console configuration

The CONSOLxx member of parmlib lets you define MCS consoles, HMCS 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, HMCS 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 because of the additional control provided by the extended MCS console interface.

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, HMCS 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 178 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, HMCS or SMCS consoles in an MVS system or sysplex can ensure that operators enter only the commands they are authorized to use.

An operator typically logs on to a single console. However, if you want to allow an operator to log on to multiple consoles concurrently within a system or sysplex, your security administrator can enable this. When the security profile MVS.MULTIPLE.LOGON.CHECK is defined in the OPERCMDS class, an operator may log on to multiple consoles. Defining this profile allows all operators to log on multiple times. There is no limit to the number of consoles to which an operator may log on. Operators are still required to provide a password while logging on to each console.

Consoles password phrase support becomes enabled on a system when the security profile is defined. There is no authority access checking from a user ID perspective.

The consoles function checks for the existence of a security profile in the OPERCMDS class to cover the MVS.CONSOLE.PASSWORDPHRASE.CHECK resource.

For example, the following RACF command can be used to define the profile:

```
REDEFINE OPERCMDS (MVS.CONSOLE.PASSWORDPHRASE.CHECK)
```

If the profile exists, the new LOGON panel display is revealed which will allow for either the new password phrase input or the standard eight (8) character passwords.

After enabling password phrases, active consoles need to be recycled to pick up the setting. If the console is not recycled, the 8-character password processing remains in effect for that console. There are several ways to recycle the console so the new password state is used:

- Place the console in standby mode (VARY CN(*),STANDBY) and then take the console out of standby mode by pressing the enter key on the console.
- Vary the console offline (VARY CN(*cnname*),OFFLINE) and then back online (VARY CN(*cnname*),ONLINE). Note that the online request must be made from another active console.
- Re-IPL the system.
- Note that SMCS consoles do not support standby, so they must be logged off and then reconnected to z/OS.

Note that during the process of an operator logging on, z/OS may issue messages referring to passwords. In these messages, passwords mean either passwords (8-byte variety) or password phrases.

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, HMCS and SMCS console configurations 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.

Using CONSOLxx

Reference

For complete information about CONSOLxx and any parmlib member, 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, refer to it to code your members.

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

- CONSOLxx, which defines console characteristics for each MCS, HMCS or SMCS console.
- CNGRPxx, which defines groups.
- PFKTABxx, which contains the program function key PFK tables for all MCS, HMCS and SMCS consoles. (For information on PFKTABxx, see “Defining PFKs and other command controls for consoles” on page 77.)
- 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 100.)
- MSGFLDxx which defines message flood automation policy. (For more information on MSGFLDxx, see Chapter 4, “Message flooding,” on page 121.)

CONSOLxx lets you define certain devices as consoles and specify attributes that determine how your operators can use MCS, HMCS 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 28 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, HMCS or SMCS console:

- 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 6 on page 17 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. The table also includes a topic reference for a description of each keyword.

Table 6. Summary of CONSOLE statement functions

Task	CONSOLE statement keyword	MVS command to change value	See topic
Defining a device as a console:			
Device number or SYSCONS, SUBSYSTEM, or SMCS	DEVNUM	1. Use the SETCON command or the sample program IEARELCN to delete the console. 2. Update the CONSOLxx parmlib member. 3. Use the SET CON=xx command to add the new console.	"Defining devices as MCS, HMCS, or SMCS consoles" on page 42
Console name	NAME	1. Use the SETCON command or the sample program IEARELCN to delete the console. 2. Update the CONSOLxx parmlib member. 3. Use the SET CON=xx command to add the new console.	"Defining devices as MCS, HMCS, or SMCS consoles" on page 42

Table 6. Summary of CONSOLE statement functions (continued)

Task	**CONSOLE statement keyword**	**MVS command to change value**	**See topic**
Kind of device	UNIT	1. Use the SETCON command or the sample program IEARELCN to delete the console. 2. Update the CONSOLxx parmlib member. 3. Use the SET CON=xx command to add the new console.	"Defining devices as MCS, HMCS, or SMCS consoles" on page 42
The VTAM® logical unit (LU) name (SMCS console only)	LU	VARY CN,LU	"Defining SMCS consoles" on page 33
Planning console security:			
When to automatically log off a user ID due to inactivity	TIMEOUT	VARY CN,TIMEOUT	"Automatic Timeout Processing" on page 62
Command authority level for the console	AUTH	VARY CN,AUTH	"Planning console security" on page 54
Override LOGON value on default statement	LOGON	VARY CN,LOGON	"Using RACF to control command authority and operator logon" on page 56
Specify the automatic activate group for the system console.	AUTOACT	VARY CN,AUTOACT	"Using AUTOACT with the system console" on page 140
Specifies when the system console is able to issue commands.	ALLOWCMD	None	*z/OS MVS Initialization and Tuning Reference*, CONSOLxx parmlib member
Controlling the console screen function:			
Input/output capability or console mode	USE	CONTROL V,USE	"Defining the USE attribute" on page 66
Message deletion mode of the console	DEL	CONTROL S,DEL	"Specifying automatic message deletion for MCS, HMCS or SMCS consoles" on page 67
Number of message lines that roll on the console screen	RNUM	CONTROL S,RNUM	Controlling the rolling rate
Number of seconds between message rolls or wraps	RTME	CONTROL S,RTME	Controlling the rolling rate
Conversational/ nonconversational message deletion	CON	CONTROL S,CON	"Manual deletion of messages" on page 71

Table 6. Summary of CONSOLE statement functions (continued)

Task	CONSOLE statement keyword	MVS command to change value	See topic
Number of lines to be deleted from the console screen using CONTROL E,SEG	SEG	CONTROL S,SEG	"Using SEG to delete groups of messages from the screen" on page 73
Defining status display areas of the console screen	AREA	CONTROL A	"Setting up out-of-line display areas on a console" on page 74
Monitoring selected events	MONITOR	MONITOR	"Displaying jobname, data set status, and TSO/E information" on page 76
Defining a PFK table for the console	PFKTAB	CONTROL N,PFK	"Defining PFKs and other command controls for consoles" on page 77
Defining the number of previously issued commands that can be retrieved	RBUF	1. Use the SETCON command or the sample program IEARELCN to delete the console. 2. Update the CONSOLxx parmlib member. 3. Use the SET CON=xx command to add the new console.	*z/OS MVS Initialization and Tuning Reference*, CONSOLxx parmlib member
Controlling message routing and message formatting:			
Routing codes for the console	ROUTCODE	• VARY CN,ROUT • VARY CN,AROUT • VARY CN,DROUT	"Defining routing codes" on page 92
Message levels for the console	LEVEL	CONTROL V,LEVEL	"Defining message levels for a console" on page 93
Message formats for console display	MFORM	CONTROL S,MFORM	"Controlling the format of messages and status information on console screens" on page 75
Receive messages directed to console id zero.	INTIDS	VARY CN,INTIDS	"Receiving messages that are directed to console ID zero" on page 95
Receive messages directed to unknown console ids.	UNKNIDS	VARY CN,UNKNIDS	"Receiving messages that are directed to unknown console IDs" on page 95
Controlling console operation in a sysplex:			

Table 6. Summary of CONSOLE statement functions (continued)

Task	CONSOLE statement keyword	MVS command to change value	See topic
System scope for messages that the console receives	MSCOPE	• VARY CN,MSCOPE • VARY CN,AMSCOPE • VARY CN,DMSCOPE	"Directing messages from other systems to a console in a sysplex" on page 94
System association for commands entered	CMDSYS	CONTROL V,CMDSYS	"Using CMDSYS on the CONSOLE statement" on page 96
Specifying the system where you want the console to be active	SYSTEM	VARY CN,SYSTEM	"Attaching consoles to particular systems in a sysplex" on page 45
Miscellaneous console operation:			
Console should support entering standby mode	SUPSBY	VARY CN,SUPSBY	"Utilizing standby mode" on page 46

Note:

1. The VARY command only changes the attributes of active consoles. Attempts to change attributes for inactive consoles are rejected. Two exceptions are the LU and LOGON attributes for SMCS consoles (not MCS). These can be changed for inactive consoles.
2. When a console is deactivated and then the console is reactivated, the console attributes used are what was specified in CONSOLxx when the system was IPLed, not the attributes that the console had when it was deactivated.

INIT, DEFAULT, and HARDCOPY statements

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

The INIT statement

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

- Specification of the console group parmlib member.
- Certain console screen functions for all consoles (activating the PFKTABxx member to control the PFK tables for MCS, HMCS 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 system commands to change some values defined on the INIT statement. See Table 7 on page 21, which summarizes console functions that you control on the INIT statement:

Note: In addition to any listed command, the SET CON=xx command can be used to change most of these values.

Table 7. Summary of INIT statement functions

Task			
Console function or attribute	**INIT statement keyword**	**MVS commands (in addition to SET CON=XX) to change most values**	**See topic**
Planning console recovery:			
Activating the CNGRPxx member that contains console group definitions	CNGRP	SET CNGRP	"Planning console groups" on page 47
Controlling the console screen function:			
Display of certain information for mount messages	MONITOR	• MONITOR • SETCON MONITOR	"Adding information to mount messages" on page 77
PFKTABxx member that contains PFK tables for consoles	PFK	SET PFK	"Setting up PFKs for consoles" on page 77
Defining the command delimiter for multiple command input	CMDDELIM	None	"Defining the command delimiter for full-capability consoles" on page 80
Specifying the VTAM APPLID that SMCS is to use on this system	APPLID	CONTROL M,APPLID	"Starting the SMCS application" on page 32
Specifying the VTAM GENERIC resource name that SMCS is to use for the sysplex	GENERIC	CONTROL M,GENERIC	"Starting the SMCS application" on page 32
Controlling message processing:			
Activating the message processing facility (MPF)	MPF	SET MPF	"MPF and MVS operations planning" on page 100
Activating the action message retention facility	AMRF	CONTROL M,AMRF	"Retaining messages" on page 102
Activating the IEAVMXIT message processing exit	UEXIT	CONTROL M,UEXIT	"Installation exits for messages and commands" on page 106
Maximum number of WTO buffers	MLIM	CONTROL M,MLIM	"Controlling WTO and WTOR message buffers" on page 109
Maximum number of WTOR buffers	RLIM	CONTROL M,RLIM	"Controlling WTO and WTOR message buffers" on page 109
Maximum number of write-to-log (WTL) buffers	LOGLIM	CONTROL M,LOGLIM	"Controlling write-to-log (WTL) message buffers" on page 114

Table 7. Summary of INIT statement functions (continued)

Console function or attribute	INIT statement keyword	MVS commands (in addition to SET CON=XX) to change most values	See topic
Activating the MVS message service (MMS) for message translation	MMS	SET MMS	"Handling translated messages" on page 114
In a sysplex, controlling the aggregation of messages returned by the ROUTE *ALL or ROUTE *systemgroupname* command	ROUTTIME	CONTROL M,ROUTTIME	"Aggregating messages returned to the ROUTE command" on page 111
Specifying the MSGFLDxx parmlib member.	MSGFLD	SET MSGFLD	Chapter 4, "Message flooding," on page 121
Controlling component tracing options			
Specifying the Parmlib member that contains tracing options for the operations services (OPS) component	CTRACE	TRACE CT	*z/OS MVS Initialization and Tuning Reference*

The DEFAULT statement

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

Note: In addition to any listed command, the SET CON=xx command can be used to change most of these values.

Table 8 on page 22 summarizes console functions that you can control using the DEFAULT statement:

Table 8. Summary of DEFAULT statement functions

Task	DEFAULT statement keyword	MVS commands (in addition to SET CON=XX) to change most values	See topic
Controlling console security:			
Operator logon to MCS, HMCS and SMCS consoles	LOGON[1]	None	"Using RACF to control command authority and operator logon" on page 56

Table 8. Summary of DEFAULT statement functions (continued)

Task	DEFAULT statement keyword	MVS commands (in addition to SET CON=XX) to change most values	See topic
Controlling the console screen function:			
Freezing the display of messages on MCS, HMCS or SMCS console screens	HOLDMODE	1. Update the CONSOLxx parmlib member. 2. Use the SET CON=xx command to change the value.	"Temporarily suspending the screen roll" on page 70
Controlling message routing:			
Assigning routing codes for messages without any specified target	ROUTCODE	1. Update the CONSOLxx parmlib member. 2. Use the SET CON=xx command to change the value.	"Handling messages without routing codes" on page 93
Assigning the name of a console group to receive synchronous messages	SYNCHDEST	Activate another CNGRPxx member (SET CNGRP) that defines the same console group but with different console members.	"Display of synchronous messages" on page 48
Controlling message processing:			
Maximum number of REPLY ids	RMAX	CONTROL M,RMAX (Note that this command can only be used to increase the value.)	"Controlling WTO and WTOR message buffers" on page 109

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 SYSLOG and/or OPERLOG as the hardcopy medium

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

For information about hardcopy processing, see "Hardcopy processing" on page 80.

Note: In addition to any listed command, the SET CON=xx command can be used to change most of these values.

Table 9 on page 23 summarizes console functions you can control using the HARDCOPY statement:

Table 9. Summary of HARDCOPY statement functions

Task	HARDCOPY statement keyword	MVS commands (in addition to SET CON=XX) to change value	See topic
Controlling logging and system recording:			
Hardcopy medium **Note:** If the Hardcopy function is important to you, you should use both SYSLOG and OPERLOG.	DEVNUM	• VARY OPERLOG,HARDCPY • VARY SYSLOG,HARDCPY	"Hardcopy processing" on page 80

Table 9. Summary of HARDCOPY statement functions (continued)

Task	**HARDCOPY statement keyword**	**MVS commands (in addition to SET CON=XX) to change value**	**See topic**
Routing codes for the hardcopy message set	ROUTCODE	• VARY OPERLOG|SYSLOG,HARDCPY,AROUT • VARY OPERLOG|SYSLOG,HARDCPY,DROUT • VARY OPERLOG|SYSLOG,HARDCPY,ROUT	"Hardcopy processing" on page 80
Hardcopy of commands by level	CMDLEVEL	VARY OPERLOG|SYSLOG,HARDCPY,cmdlevel	"Hardcopy processing" on page 80
Defining year format in SYSLOG	HCFORMAT	None	"Hardcopy processing" on page 80

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, HMCS and SMCS consoles defined by CONSOLE statements are active, and the values specified for INIT and DEFAULT , and HARDCOPY control console operations for the system. Operators can use the CONTROL, MONITOR, SET, and VARY commands to change many of the definitions after the system is active.

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 on the CONSOLE statement has sysplex scope. NAME specifies a unique name that identifies the console within 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, UNIT for CONSOLE and all keywords for HARDCOPY have system scope. The device type (UNIT) for the console applies 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.

See Table 17 on page 153 and Table 18 on page 157 to check which keywords on each CONSOLxx statement are system or sysplex in scope.

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.
3. 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 24, 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 on page 25, 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).

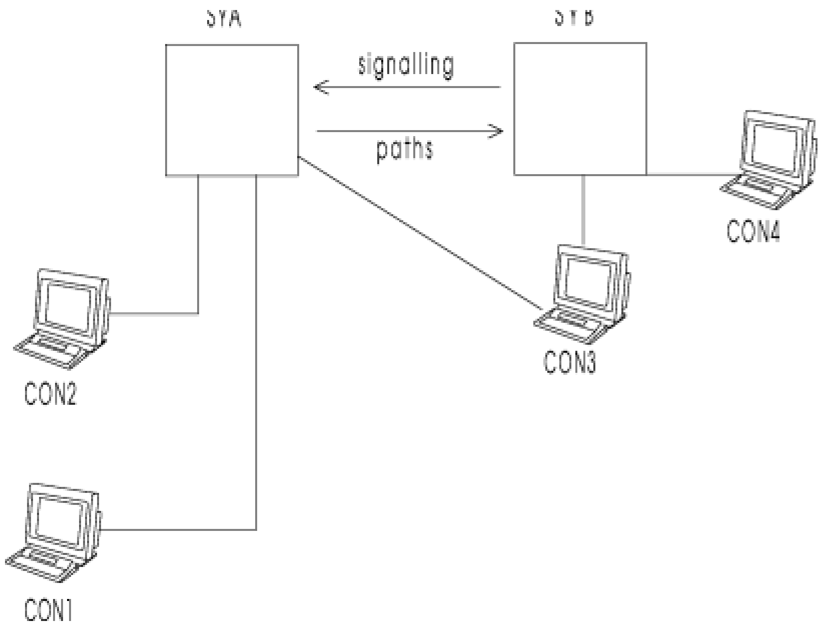
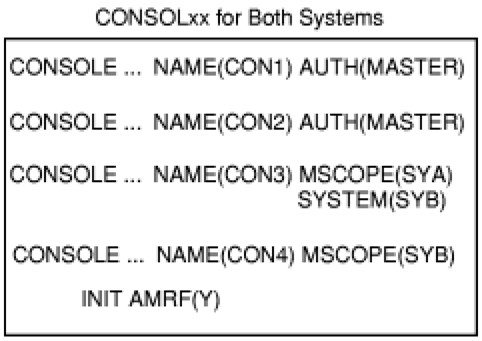


Figure 3. Console Configuration in a Sysplex with Two Systems and Four MCS Consoles

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.
- A console can be active on only one system at a time. In Figure 3 on page 25, 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 are 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 SYS3

For more information about using system symbols in parmlib members, including lists of valid system symbols, see the topic 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 25, Figure 4 on page 26, 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.

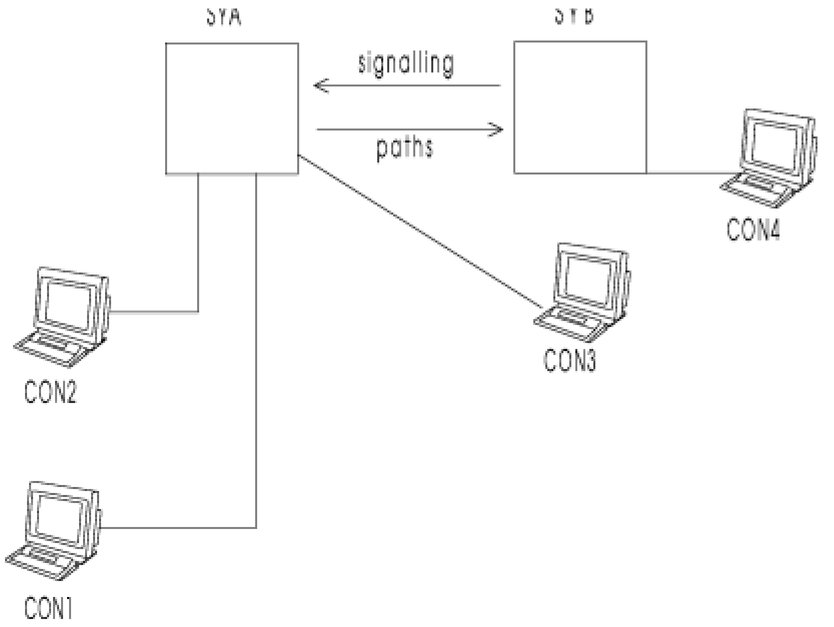
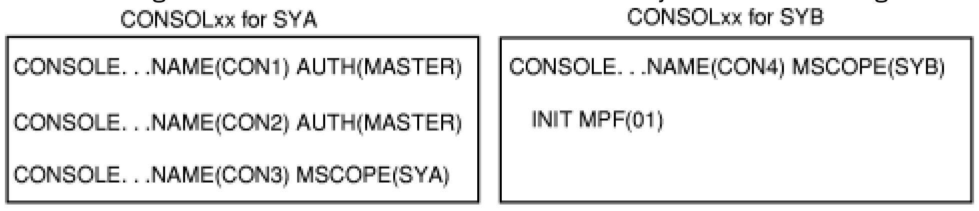


Figure 4. Console Configuration in a Sysplex with Two Systems and Four MCS Consoles

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

```
CONSOLE. . .NAME(CON1) AUTH(MASTER)

CONSOLE. . .NAME(CON2) AUTH(MASTER)

CONSOLE. . .NAME(CON3) MSCOPE(SYA)
```

CONSOLxx for SYB

```
CONSOLE. . .NAME(CON4) MSCOPE(SYB)

   INIT MPF(01)
```

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 CONSOLxx keywords with sysplex scope, keyword values apply to all the systems in the sysplex. For example, the RMAX keyword in CONSOLxx defines the maximum number of replies for the sysplex. It is specified, or defaulted to, by the first system to enter the sysplex. Subsequent CONSOLxx RMAX values will be ignored and the value can only be altered through command (K M,RMAX). Since the first system governs the value it is important to understand the scope of the CONSOLxx keywords.

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 on page 27, 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):

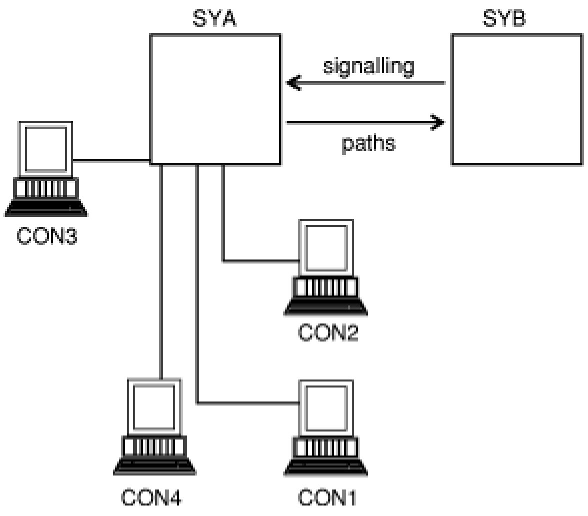
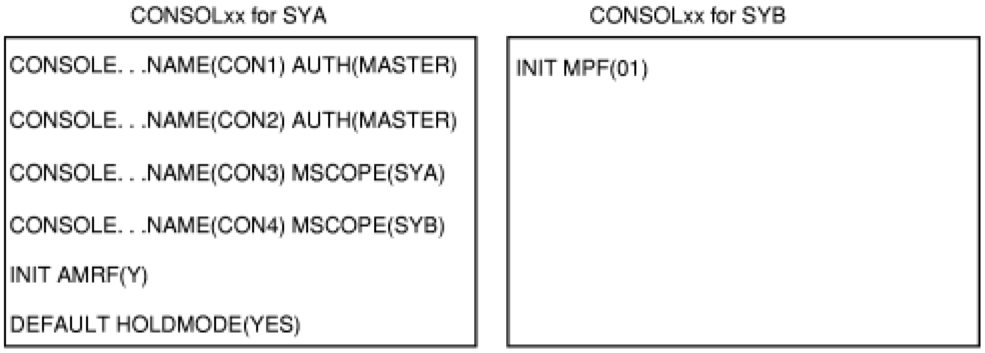


Figure 5. Console Configuration in a Sysplex with Four MCS Consoles Attached to One System

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

```
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)
```

CONSOLxx for SYB

```
INIT MPF(01)
```

The first system to join the sysplex (SYA) is the focal point of console operations for the sysplex configuration in Figure 5 on page 27. 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 5 on page 27 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, SMCS consoles, or extended MCS consoles (EMCS) are ways to resolve the problem.

SMCS console considerations

SMCS consoles are MCS consoles that use z/OS Communications Server SNA and TCP/IP 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.
- z/OS Communications Server must be active for SMCS to be active. The system console and MCS consoles do not rely on z/OS Communications Server, and these can be used before z/OS Communications Server 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.

Because an SMCS console is connected through a network and uses z/OS Communications Server services, the z/OS Communications Server 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 29 shows a sample LOGON mode table entry. Table 10 on page 29 defines the keywords in the LOGON mode table entry, and the values that SMCS expects. Table 11 on page 30 provides details about the values to specify for the PSERVIC keyword.

```
**********************************************************************
*                                                                    *
*        DYNAMIC LOGMODE ENTRY FOR SNA                               *
*        3270 DISPLAYS (APPLIES TO QUERIABLE TERMINALS)              *
*                                                                    *
**********************************************************************
DYNSNA   MODEENT LOGMODE=DYNSNA,COS=INTERACT,APPNCOS=#INTER,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'90',
               COMPROT=X'3080',
               RUSIZES=X'87F8',  * OUTBOUND 3840     INBOUND 1024
               TYPE=1,
               PSERVIC=X'028000000000000000000300'
```

Figure 6. Sample LOGON Mode Table Entry

Table 10. Keyword Definitions

Keyword	Definition	Local Non-SNA Value	SNA Value
FMPROF	Function Management Profile	X'02'	X'03'
TSPROF	Transmission Services Profile	X'02'	X'03'
PRIPROT	Primary LU Protocol	X'71'	X'B1'
SECPROT	Secondary LU Protocol	X'40'	X'90'
COMPROT	Common LU Protocol	X'2000'	X'3080'

Table 10. Keyword Definitions (continued)

Keyword	Definition	Local Non-SNA Value	SNA Value
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 11 on page 30 for values.

Table 11. PSERVIC Values for SNA Devices. Note that failing to follow the recommended setting for bytes 2, 9 and 10 may result in the SMCS console session being established with default console attributes such as RTME.

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.
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).

Table 11. PSERVIC Values for SNA Devices. Note that failing to follow the recommended setting for bytes 2, 9 and 10 may result in the SMCS console session being established with default console attributes such as RTME. *(continued)*

Byte	Value	Definition
9,10	Possible values: • 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: • 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: • 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)
```

You can change the APPLID after the system is active, but only when an APPLID was specified in CONSOLxx during IPL. If you omit APPLID, you can run the **SET CON=** command to specify an APPLID at any time.

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 an 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 an SMCS console definition follows:

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

SMCS consoles are not associated with a particular system. An 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. 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 or SYSTEM parameters on the CONSOLE statement. Also, the only acceptable value for the USE keyword is FC.

All consoles require the NAME parameter. If NAME is not specified, or is not valid, the CONSOLE statement is rejected. 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.

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 an 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 explicitly 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

You can use the VARY CN command to change the LOGON value of an SMCS console or an active MCS console after the system is active:

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

The change will take effect immediately.

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

For an active console, LOGON can be combined with other parameters on the VARY command; for an inactive SMCS console, LOGON can only be combined with LU.

Changing the predefined LU

The VARY CN command can also change the predefined LU of an SMCS console:

```
     VARY CN(consname),LU=luname
```

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

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

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

For an active console, LU can be combined with other parameters on the VARY command; for an inactive console, LU can only be combined with LOGON.

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.

An operator typically logs on to a single console. However, if you want to allow an operator to log on to multiple consoles concurrently within a system or sysplex, your security administrator can enable this. When the security profile MVS.MULTIPLE.LOGON.CHECK is defined in the OPERCMDS class, an operator may log on to multiple consoles. Defining this profile allows all operators to log on multiple times. There is no limit to the number of consoles to which an operator may log on. Operators are still required to provide a password while logging on to each console.

Consoles password phrase support becomes enabled on a system when the security profile is defined. There is no authority access checking from a user ID perspective.

The consoles function checks for the existence of a security profile in the OPERCMDS class to cover the MVS.CONSOLE.PASSWORDPHRASE.CHECK resource.

For example, the following RACF command can be used to define the profile:

```
REDEFINE OPERCMDS (MVS.CONSOLE.PASSWORDPHRASE.CHECK)
```

If the profile exists, the new LOGON panel display is revealed which will allow for either the new password phrase input or the standard eight (8) character passwords.

After enabling password phrases, active consoles need to be recycled to pick up the setting. If the console is not recycled, the 8-character password processing remains in effect for that console. There are several ways to recycle the console so the new password state is used:

- Place the console in standby mode (VARY CN(*),STANDBY) and then take the console out of standby mode by pressing the enter key on the console.
- Vary the console offline (VARY CN(*cnname*),OFFLINE) and then back online (VARY CN(*cnname*),ONLINE). Note that the online request must be made from another active console.
- Re-IPL the system.
- Note that SMCS consoles do not support standby, so they must be logged off and then reconnected to z/OS.

Note that during the process of an operator logging on, z/OS may issue messages referring to passwords. In these messages, passwords mean either passwords (8-byte variety) or password phrases.

Commands

Certain commands should be restricted only to users who need to issue those commands. MVS commands, RACF access authorities, and resource names in *z/OS MVS System Commands* lists all of the MVS commands that can be issued and the resource names that you can use to protect them.

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 authority, and it is a very good idea to use a security product to limit access to the commands. See MVS commands, RACF access authorities, and resource names in *z/OS MVS System Commands*.

Actions to take for inactivity

Consider the action that the system should take if the operator does not interact with the console for a specified period of time. You can specify the TIMEOUT function on the console definition. If there is no input activity within the number of minutes that are specified with TIMEOUT, a **LOGOFF** command is issued for the console.

Note: Input activity includes the pressing of an attention key such as enter, PFK, PA1, or PA2.

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 54 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 54 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 z/OS 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 a NIP console must be used to 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 on page 37.
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"
5650-ZOS (C) Copyright IBM Corp. 2001
```

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, HMCS, SMCS, or Subsystem console defined in CONSOLxx. In a sysplex, deleting a console definition releases the console ID and console name 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 can also remove EMCS console definitions. See “Removing extended MCS console definitions from a configuration” on page 40 for more information.

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 and console name with the inactive console. You can delete the console definition making the console ID and console name available for reuse. When you add a new console, the system reassigns the console id.

There are two ways in which you can delete a console definition. The first, and preferred way, is to use the `SETCON DELETE` command. For example, to delete an SMCS console named SMCSSY1, issue:

```
SETCON DELETE,CN=SMCSSY1
```

For more information about the `SETCON DELETE` command, see *z/OS MVS System Commands*.

The second way to remove a console definition is to assemble and link the sample source code for program `IEARELCN` in SYS1.SAMPLIB. “Sample invocation of IEARELCN” on page 38 describes the sample job for invoking the console service. The programming environment, and the return and reason codes for invoking the console service are described in the sample program prologue.

The following restrictions for removing a console definition apply:

- Dynamic I/O reconfiguration cannot be performed for a device that is defined as an MCS console. If you want to change the I/O configuration of the device, you must first delete the console definition. After the definition has been removed, the device can be dynamically reconfigured. 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*.)
- If an HMCS console is in Standby mode, a `VARY CONSOLE OFFLINE` or `RESET CONSOLE` must first be issued before attempting to delete the console definition.

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
```

xxxxxxxx: 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.

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Meaning and Action**
04	14	Caller is holding locks. Ensure that caller is not holding any locks.
04	18	Caller is not in task mode. Ensure that caller is running in task mode.
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.
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.
08	24	This reason code is for IBM internal diagnostic purposes only. Record it and supply it to the appropriate IBM support personnel.
08	28	This reason code is for IBM internal diagnostic purposes only. Record it and supply it to the appropriate IBM support personnel.
08	2C	This reason code is for IBM internal diagnostic purposes only. Record it and supply it to the appropriate IBM support personnel.
08	30	This reason code is for IBM internal diagnostic purposes only. Record it and supply it to the appropriate IBM support personnel.
08	34	This reason code is for IBM internal diagnostic purposes only. Record it and supply it to the appropriate IBM support personnel.

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Meaning and Action**
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.

Removing extended MCS console definitions from a configuration

You can delete the definition of any extended MCS console, thus freeing the ID and console name that had been assigned to the extended MCS console. The system then can reuse that ID for a newly-defined extended MCS console.

There are two ways in which you can delete a console definition. The first, and preferred way, is to use the `SETCON DELETE` command. For example, to delete an extended MCS console named EMCSSY1, issue:

```
SETCON DELETE,CN=EMCSSY1
```

For more information about the `SETCON DELETE` command, see *z/OS MVS System Commands*.

The second way to remove a console definition is to assemble and link the sample source code for program `IEARELEC` in SYS1.SAMPLIB. "Sample invocation of IEARELEC" on page 41 describes the sample job for invoking the console service. The programming environment, and the return and reason codes for invoking the console service are described in the sample program prologue.

The following restrictions for removing an extended MCS console apply:

- The extended MCS console must be inactive.
- The console ID of a removed extended MCS console can be reused once it has been deactivated and removed. It is safe to use the console ID to process a command response, but you should avoid saving the console ID for later processing. Therefore, you should use the console name to direct messages to specific consoles. If the console ID is used, messages may end up going to unintended consoles.
- Do not remove or change the definition of the following system-used extended MCS consoles:
  - The ROUTE command console, named *ROUTExx, where xx is the value of &SYSCLONE
  - The OPERLOG console, named *OPLOGxx, where xx is the value of &SYSCLONE
  - The SYSLOG console, named *SYSLGxx, where xx is the value of &SYSCLONE
  - The DIDOCS EMCS console, named *DICNSxx, where xx is the value of &SYSCLONE
  - The System REXX consoles, named *AXTnnxx and *AXRnnxx, where nn is the worker address space number and xx is the value of &SYSCLONE
  - The IOS EMCS console, named SYSIOSRS
  - The JES3 DLOG console, named JES3DLOG

Sample invocation of IEARELEC

SYS1.SAMPLIB provides a sample program in member IEARELEC to remove an inactive extended MCS console.

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

xxxxxxxx: is the name of the console whose definition is to be removed. The use of wildcards is supported.

Environment

You can also invoke the console definition removal service (CNZM1ERF) from an authorized program. CNZM1ERF 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 CNZM1ERF 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 parameter list.

Parameter list contents

Bytes 0-7
Console name or wildcard pattern

Bytes 8-15
Command and response token (CART) to be associated with any message issued by the EMCS Console Removal Service.

If a CART is not needed, specify binary zeros as input.

Bytes 16-23
The console name of the console (CONSNAME) that will receive any message issued by the EMCS Console Removal Service.

If a CONSNAME is not needed, specify binary zeros as input.

Return and reason codes

When control returns from the extended MCS console removal service (module CNZM1ERF), 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	400	All extended MCS consoles matching the wildcard pattern are active. No console definitions were removed.

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning and Action
04	404	One or more extended MCS consoles matching the wildcard pattern are active. Console definitions were removed for the consoles that were not active.
04	408	No extended MCS consoles were found matching the wildcard pattern. No console definitions were removed.
04	40C	There was not enough available storage to complete the request. Only the extended MCS consoles listed in message CNZ4002I had their console definitions removed.
08	800	Caller is not authorized. Caller is not in supervisor state.
08	804	Caller is not authorized. Caller is not in key zero.
08	808	Caller is in cross memory mode.
08	80C	Caller is running in SRB mode.
08	810	Requested console is active.
08	814	Requested console is not an extended MCS console.
08	818	Requested console is not defined.
08	820	Incorrect input was specified.
10	1000	An unexpected ABEND occurred.
10	1004	Storage could not be obtained.
10	1008	Resources could not be obtained.
10	100C	Resources could not be released.
10	1010	Unexpected return code from CONVCON.

Defining devices as MCS, HMCS, 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. NAME is required for ALL consoles. If NAME is not valid, the system rejects the CONSOLE statement.

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. NAME is required.

You can specify DEVNUM(SYSCONS) to define the system console in CONSOLxx. See "The system console and CONSOLxx" on page 140.

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

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

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

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

Do not specify the UNIT or SYSTEM keywords on the CONSOLE statement for HMCS and 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.

Input/output devices are also called display consoles. 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 66.

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 48.)

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*.

Using console names

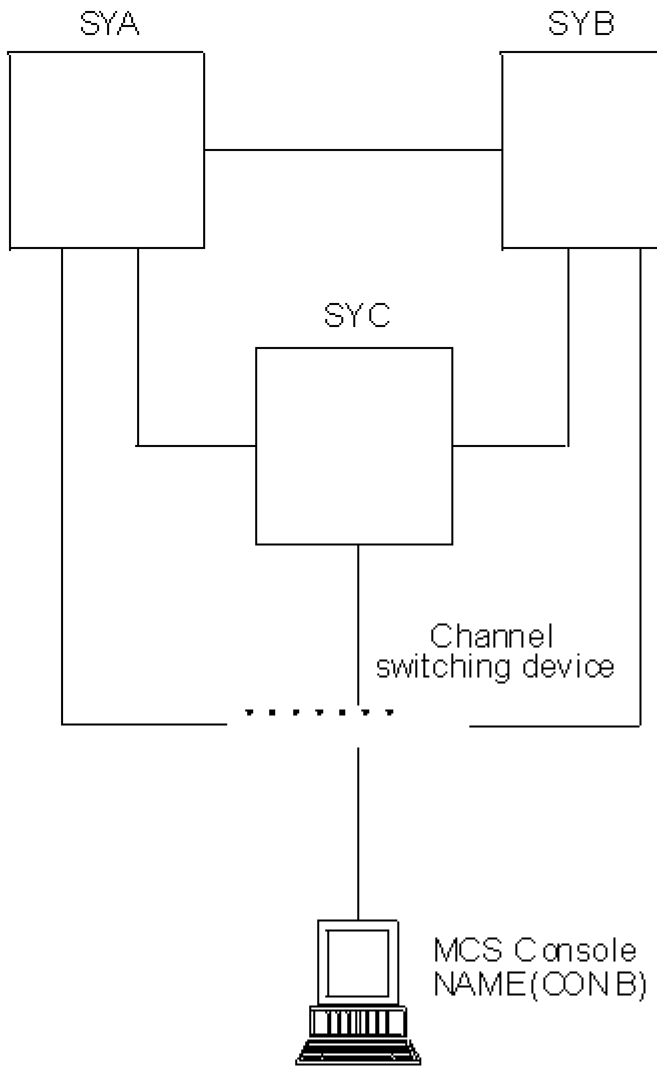
Define each console by device number and device unit on the CONSOLE statement and name each MCS console. Console names are required for all CONSOLE statements in CONSOLxx.

Using console names in a sysplex

When defining consoles for a sysplex, names are required for MCS, HMCS, SMCS, 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 25.

In a sysplex, the console name and console ID uniquely identify the console to the sysplex for the life of the sysplex or the removal of the console by `SETCON DELETE`, IEARELCN or IEARELEC. When a console is removed, the binding of the console name to the console ID is broken. If the console ID is re-used by MVS, it may be bound to a different console name and represent a different console. A console name will always represent the same console in CONSOLxx; a console ID may not. Programs that need to communicate with specific consoles should always use console names to avoid communicating with an unintended console.

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 CONB 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(CONB)
```

For SYB:

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

For SYC:

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

CONB can be active on only one system in the sysplex at a time. If CONB is active on SYA and SYA fails, the operator can activate CONB on SYB or SYC. The newly activated CONB console uses the definition from its current system

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
- LOGON
- LOGOFF
- OPERLOG
- SYSIOSRS
- SYSLOG
- UNKNOWN

Note about SYSIOSRS

Console name SYSIOSRS is reserved for system use.

The IOSAS address space must have an associated “trusted” userid defined 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*

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 140.

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).

Utilizing standby mode

To activate a console, you typically must issue a VARY command from another console. If another console is not available and you are activating an HMCS console, you also have the option of accessing the HMC and attaching the Integrated 3270 icon to the LPAR icon. After pressing a key that generates an attention, the console is active. These console activation methods consume system services and resources such as storage, queues, CPU cycles, and lock usage, many of which are potentially unavailable during emergency situations. To reduce the amount of system services and resources that are used, and to better prepare for emergency situations, consider placing consoles that you are not actively using in standby instead of inactive mode.

Standby mode is available for HMCS and MCS display consoles in full capability (FC) mode. SMCS, EMCS, subsystem consoles, the system console, printer consoles, and consoles in status display (SD) or message stream (MS) mode are not supported.

When a console is in standby mode, z/OS no longer processes I/O on it. However, the console has already undergone the open process, and ENQs and control blocks that represent the console are already obtained. So, the overhead that is required to switch the console back into active mode is greatly reduced, which is especially beneficial during critical situations.

Note: Consoles cannot initialize in standby mode because the open process requires an online device. You must activate a console before switching it to standby mode.

The SUPSBY option on the CONSOLE statement designates whether a specified console should support standby mode. If the console is not eligible for standby mode, N/A will display. After a console is set to support standby mode, you can use various methods to switch it between active and standby modes.

Switch from active mode to standby mode

Use either of the following two methods to switch your console from active mode to standby mode:

- Drop the session.
- Issue the `VARY CN(`*`name`*`),STANDBY` command

Switch from standby mode to active mode

Use either of the following two methods to switch your console from standby mode to active mode:

- Press an attention key such as enter or PA1.
- Issue the `VARY CN(`*`name`*`),ONLINE` command.

Note: If the console requires authentication, you must log on again after the console is reactivated.

Consoles in standby mode count toward the limit of 99 active consoles per system.

Standby support is similar to the Open Systems Adapter (OSA) Express Integrated Console Controller (ICC) feature on MCS consoles called "Defer Host Disconnect" (DHD). However, when implementing DHD, z/OS continues to process I/O on the console while the session is disconnected. OSA just claims that the I/O was successful. Using the standby mode instead is an efficient choice because, with standby, no I/O goes to the console and fewer system resources are utilized.

Planning console recovery

The following sections discuss how to plan for console recovery.

Recovery considerations

When you plan console recovery for a system or a sysplex, first consider all the MCS consoles, HMCS consoles, SMCS consoles, and extended MCS consoles you have defined for your console configuration. (For MCS, HMCS 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 defining multiple instances of the consoles that you need to operate your installation.

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 inactive console is an SMCS console, logon to the console again.

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

Planning console groups

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

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

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

- CONSOLE AUTOACT to define the group of consoles that determine the usage of the system console. (See "Using AUTOACT with the system console" on page 140.)
- DEFAULT SYNCHDEST to define a group of consoles able to display synchronous messages. (See "Display of synchronous messages" on page 48.)

When you define group names in CNGRPxx, do not use the names of MCS, HMCS or SMCS consoles defined in CONSOLxx or EMCS console key names displayed on the DISPLAY CONSOLES,KEY command as a group name.

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 a 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 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*.

Display of synchronous messages

Synchronous messages are WTO or WTOR messages that can be issued during initialization or recovery situations. When a synchronous WTOR is issued, message CNZ4215W is issued to all active MCS consoles on the system issuing the synchronous WTOR. The highlighted WTO will indicate:

- the message id of the synchronous WTOR for which a reply is needed
- on which console the synchronous WTOR is currently being displayed
- if the message is being managed by the z/OS auto-reply function.

You can define a full-capability MCS console, or the system console, as members of a console group in CNGRPxx 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 console group whose members can receive a synchronous message.

If a reply is not provided within approximately 125 seconds, the synchronous WTOR is moved to the next console in the SYNCHDEST list.

MVS searches for an eligible console based on the order of the console names specified in the group. The moving of the WTOR to different consoles will continue until a reply is provided or until the SYNCHDEST list is exhausted. If all consoles in the SYNCHDEST list have been tried, the WTOR is displayed on the system console and will remain there until a reply is provided. You can specify valid MCS console names as members of the group. You can also specify *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 a console group on SYNCHDEST or none of the consoles on SYNCHDEST are active, the system that issues the message tries to display the message 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.

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.

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.

Important: You must respond to synchronous WTOR messages promptly. Synchronous WTOR messages (such as IOS115I and IEA367A) prevent the system from updating its status on the sysplex couple data set. This, in turn, can lead to Sysplex Failure Management (SFM) deciding that the system is not responding normally, and removing it from the sysplex.

If you have a standard response to synchronous WTOR messages such as these, you may want to automate the reply to these messages to avoid delay and avoid SFM partitioning the system out of the sysplex.

Note that MPF and typical automation products are unable to automate a response to these messages. However, Auto-reply is able to automate these messages.

System console automatic activation

The AUTOACT keyword for the system console specifies a console group. The consoles in this group can "replace" the system console. The AUTOACT support is especially useful if you have no MCS consoles (for example, all of them are SMCS consoles). With this support, the system console can be used during NIP, activated automatically at the end of NIP, and deactivated when an SMCS console activates. See “Using the AUTOACT console group” on page 138 for more information.

Recovery for consoles

Several errors can directly affect operation of the display consoles used by operators. Symptoms may be obvious, such as the screen suddenly fails, messages describe the error, or the keyboard locks. In other cases, the error might not be immediately apparent. Prepare recovery actions for the following.

System problems

When a system error occurs, one or more of the following can happen:

- The screen blanks out, and then an error message appears in the message area. See “Recovery actions for an error message in the message area ” on page 49.
- There is an abnormal lack of console activity. See “Recovery actions for a lack of console activity” on page 50.
- Messages IEA405E and IEA404A appear, to indicate that console messages are backed up. See “Console message backups ” on page 50.

Console hardware errors

When a console hardware error occurs, one or more of the following can happen:

- Error messages are centered on the screen. The remainder of the screen is blank. See “Recovery actions for error messages centered on the screen ” on page 53.
- The screen blanks out, but no error message appears.
- The screen appears normal, but the keyboard is locked and the operator cannot enter commands. See “Recovery actions for a locked keyboard” on page 53.

See also the section on how to process operating system messages at the system console. See “Actions to see system messages at the system console” on page 54.

System programming problems

Recovery actions for an error message in the message area

An error message in the message area at the top of the screen indicates that a recoverable system error occurred.

1. Perform the action specified by the error message.
2. Perform a cancel action. This may be pressing the PA2 key. This should restore the screen.
3. Review the messages to make sure that no messages were lost during error recovery.

Recovery actions for a lack of console activity

A lack of console activity can be no messages appearing on the console or no response by the system to commands. Causes can be simply a low level of system activity or a problem in message processing in the operating system.

1. If a full-capability console appears inactive, check the system response by requesting a display of the time. Enter a DISPLAY T command.

   The system should respond within a few seconds with the time and date.
2. Take the console out of hold mode (see "Temporarily suspending the screen roll" on page 70 for more details).
3. If it does not, enter a CONTROL C,D command to cancel any status displays being presented on the unresponsive console.

   The console should return to normal activity.
4. If it does not, assume the console has some other problem. Do the following:
   a. Enter a CONTROL Q command to delete messages that are queued to the console, but have not yet been displayed. These messages have already been through hardcopy processing.
   b. Check for a console hardware error.
   c. If possible, bring up another console and enter a RESET CN command for the unresponsive console.
   d. Notify the system programmer.
   e. If necessary, reIPL the system, following normal procedures.

Console message backups

The operating system places WTO and WTOR messages in buffers in virtual storage. The WTO buffers hold messages the system has not yet displayed at eligible consoles. Each WTOR buffer holds one WTOR message that the system has displayed but that an operator has not replied to. The MLIM and RLIM parameters on the INIT statement in the CONSOLxx parmlib member specify the maximum numbers of buffers. The RMAX parameter on the default statement in CONSOLxx also affects the number of WTOR buffers, because the system cannot have more WTOR buffers than the largest reply identifier value. If the installation does not code these parameters, the system defaults are:

- For WTO messages, 1500 buffers
- For WTOR messages, 10 buffers

  Note: It is suggested that you specify a minimum of 99 buffers for WTOR messages.

In a sysplex, the first system that joins determines the RMAX value. If no RLIM value is set, RLIM is set to RMAX for all systems in the sysplex.

Messages back up when the system cannot free buffers for new messages because the buffers contain old messages.

For more information on changing or displaying the number of allowed WTO (write-to-operator) or WTOR (write-to-operator-with-reply) message buffers, see the topic named "Changing or Displaying the Number of Allowed WTO and WTOR Message Buffers" in *z/OS MVS System Commands*.

You can use Message Flood Automation to help reduce the possibility that a message flood will cause a console message back-up. See Chapter 4, "Message flooding," on page 121.

WTO buffer backup

When WTO message buffer use reaches 80% of the limit, the system issues the following message. The system also takes out of hold mode any consoles in hold mode with messages queued.

```
IEA405E WTO BUFFER SHORTAGE - 80% FULL
```

If the problem continues and WTO buffer use reaches its limit, the system issues the following message:

```
IEA404A SEVERE WTO BUFFER SHORTAGE - 100% FULL
```

WTOR buffer backup

When WTOR message buffer use reaches 80% of the limit, the system issues the following message:

```
IEA230E WTOR BUFFER SHORTAGE - 80% FULL
```

If the problem continues and WTOR buffer use reaches its limit, the system issues the following action message:

```
IEA231A WTOR BUFFER SHORTAGE CRITICAL - 100% FULL
```

Notes about console message backups

- All lines of an out-of-line multiple-line status display that have not been presented occupy message buffers. Therefore, the operator should erase these displays when they are no longer needed to free the message buffers.
- During system initialization, the system does not use the MLIM and RLIM parameter values in the CONSOLxx parmlib member until either the system log or a console becomes active or processing by the nucleus initialization program (NIP) completes. After NIP processing, consoles other than the initialization console become active and buffer space becomes important.
- When activating an extended MCS console, specify the optional alert percentage and an event control block (ECB) address on the MCSOPER macro. Then, when the message data space reaches the specified percentage, the system will post the ECB.

Recovery actions for a WTO buffer shortage

1. Determine why the buffers are full and correct the problem. Possible reasons are:
   - A console is not ready because:
     - An I/O error occurred.
     - One or more consoles have a slow roll time.
     - The console is in roll deletable (RD) mode but the screen is filled with action messages.
   - The buffer limit is too low to handle the message traffic in the system. Either the MLIM value in the CONSOLxx parmlib member is too low, or the system default is not sufficient.
2. Also, do the following:
   a. Enter a DISPLAY CONSOLES,BACKLOG command. The display lists all MCS and SMCS consoles that have any outstanding WTO messages.
   b. Look in the display for the CURR= and LIM= values for messages where

      CURR=aaaa
      The number of WTO buffers in use

      LIM=bbbb
      The limit of WTO buffers
   c. If the buffer limit is not adequate, issue a CONTROL M,MLIM command to increase the WTO buffer limit for the duration of the IPL.
   d. Issue K S,DEL=R to ensure that the console is in Roll mode.
   e. Issue a K Q to delete the queue of messages on the console. You may have to issue the K Q command multiple times to clear the backlog. You should issue the K Q command for each console that has a backlog.
3. Notify the system programmer of the buffer problem, if it was not caused by hardware. The system programmer may want to increase the buffer limit before the next IPL.

When the number of buffers in use drops below 60% of the limit specified at IPL time, the system issues the following message:

```
IEA406I WTO BUFFER SHORTAGE RELIEVED
```

Recovery actions for a WTOR buffer shortage

Enter a DISPLAY R,R command to see the accumulated WTOR messages.

- Reply to the outstanding WTOR messages.
- Cancel jobs that are currently issuing WTOR macros.
- Enter the CONTROL M,RLIM command to increase the number of WTOR buffers.

Note: The RLIM value cannot be higher than the RMAX value (RMAX is the highest possible reply ID). If you need to increase RLIM higher than RMAX, consider entering the CONTROL M,RMAX command to increase RMAX. To determine the current values of RLIM and RMAX, issue the K M,REF command.

The value of RMAX controls the number of digits in all reply IDs. For example, increasing RMAX to 100 (or higher) causes all WTORs to have 3-digit reply IDs. This might affect automation routines. Check with the system programmer before increasing RMAX higher than 99.

If the shortage recurs, have the system programmer increase the value for RLIM or RMAX in the CONSOLxx parmlib member. It is suggested that you use a RMAX value of 9999.

Recovery actions for command flooding indications

Command flooding occurs when too many MVS commands are issued at one time, possibly because a program has issued too many MGCRE macros. The first indication of command flooding is message

IEE822E COMMANDS ARE AT 80% OF LIMIT IN COMMAND CLASS cc, followed by message IEE806A COMMANDS EXCEED LIMIT IN COMMAND CLASS cc. The CMDS command can be used to correct this situation.

1. Determine why the command flooding occurred.

   Issue the CMDS SHOW command. This displays all of the executing commands as well as the commands that are waiting for execution with the time that the command started execution and the job that issued the commands.
2. If most of the commands in the CMDS SHOW output appear to be from the same job or automation program, the job may be in a loop issuing commands, or the job may have legitimately issued a large number of commands. For example, a program may have issued a large number of VARY commands for many devices. If the commands were issued legitimately, and it appears that the commands are being processed, the commands may be allowed to complete execution.

   If the commands appear to have been issued in error, take one of the following actions:

   - Use the CANCEL command to cancel the job that is issuing the commands.
   - Use the CMDS REMOVE command to remove the commands that are waiting for execution. CMDS REMOVE cannot remove commands that are already executing.
3. If it appears that an executing command has been running for a long time, it may be hung due to a resource deadlock or other required action. Issue CMDS DUMP to obtain diagnostic data about why the command is hung. The CMDS ABEND command can be used to ABEND the command that is hung. CMDS ABEND should be used with extreme caution and should be used only as a last resort, as the system could be left in an inconsistent state.

Console hardware errors

Recovery actions for error messages centered on the screen

If a console hardware error occurs, one of the following pairs of messages can appear centered on the screen:

```
IEE170E RETRYABLE ERROR. RECENT ACTION MAY NEED TO BE REPEATED
IEE170E PRESS THE CANCEL KEY TO RESTORE THE SCREEN
IEE171E CONDITIONAL ERROR. RECENT ACTION MAY NEED TO BE REPEATED
IEE171E PRESS CANCEL TO CONTINUE
```

1. Perform a cancel action, which may be pressing the PA2 key.

The cancel action should restore most of the screen, including messages displayed in line in the message area, the instruction line, and the warning line. However, the system blanks out the entry area and the PFK line, positions the cursor to the first data entry position, and ends message numbering, if active.

Note: If you do not perform a cancel action, the system rewrites the screen (same effect as cancel) after about 30 seconds.

If the keyboard input for the cancel action results in a console hardware error, the system sees this error as a permanent I/O error and deactivates the console.

Recovery actions for a locked keyboard

You may find that you cannot enter commands through a console that otherwise appears normal.

Note: Inhibited input, with or without keyboard locking, can also occur for the following system programming problems. See "System programming problems" on page 49.

- The system enters a wait state.
- The system is in a disabled loop.

- A problem occurs in message processing in the operating system.

1. Look for software problems first, based on messages and logrec data set error records. Many software problems can inhibit console input.
2. If no software problems are found, try to restore the screen by performing a cancel action, which may be pressing the PA2 key.
3. Contact hardware support to fix the error in the failed console.

Actions to see system messages at the system console

To display operating system messages, select the desired CPC or Image object(s), and then start the Operating System Messages task. This task will provide a tabbed interface with tabbed page for each active operating system message interface. Each of these pages contains an entry field where commands and responses can be entered.

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.

An operator typically logs on to a single console. However, if you want to allow an operator to log on to multiple consoles concurrently within a system or sysplex, your security administrator can enable this. When the security profile MVS.MULTIPLE.LOGON.CHECK is defined in the OPERCMDS class, an operator may log on to multiple consoles. Defining this profile allows all operators to log on multiple times. There is no limit to the number of consoles to which an operator may log on. Operators are still required to provide a password while logging on to each console.

Consoles password phrase support becomes enabled on a system when the security profile is defined. There is no authority access checking from a user ID perspective.

The consoles function checks for the existence of a security profile in the OPERCMDS class to cover the MVS.CONSOLE.PASSWORDPHRASE.CHECK resource.

For example, the following RACF command can be used to define the profile:

```
REDEFINE OPERCMDS (MVS.CONSOLE.PASSWORDPHRASE.CHECK)
```

If the profile exists, the new LOGON panel display is revealed which will allow for either the new password phrase input or the standard eight (8) character passwords.

After enabling password phrases, active consoles need to be recycled to pick up the setting. If the console is not recycled, the 8-character password processing remains in effect for that console. There are several ways to recycle the console so the new password state is used:

- Place the console in standby mode (VARY CN(*),STANDBY) and then take the console out of standby mode by pressing the enter key on the console.
- Vary the console offline (VARY CN(*cnname*),OFFLINE) and then back online (VARY CN(*cnname*),ONLINE). Note that the online request must be made from another active console.
- Re-IPL the system.
- Note that SMCS consoles do not support standby, so they must be logged off and then reconnected to z/OS.

Note that during the process of an operator logging on, z/OS may issue messages referring to passwords. In these messages, passwords mean either passwords (8-byte variety) or password phrases.

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" on page 55 describes the AUTH attribute and command groups. "Using RACF to control command authority and operator logon" on page 56 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 35.

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 can 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 control 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 JES 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, HMCS or SMCS console, use the following keyword on the CONSOLE statement.

AUTH
Defines the command authority for an MCS, HMCS 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

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 86.

Assigning a console master authority

To ensure that you have the ability to operate your installation at all times, you should define multiple consoles with master authority. Note that the system console is forced to have master authority.

For example, to assign master authority to a console named CONS1 (device number 031), code the following CONSOLE statement in CONSOLxx:

```
CONSOLE DEVNUM(031) NAME(CONS1)AUTH(MASTER)
```

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. 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, HMCS and SMCS consoles. However, to control operator logon, limit the use of specific commands to specific MCS, HMCS and SMCS consoles, or control command use for extended MCS consoles, your 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 a 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, HMCS 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.

An operator typically logs on to a single console. However, if you want to allow an operator to log on to multiple consoles concurrently within a system or sysplex, your security administrator can enable this. When the security profile MVS.MULTIPLE.LOGON.CHECK is defined in the OPERCMDS class, an operator may log on to multiple consoles. Defining this profile allows all operators to log on multiple times. There is no limit to the number of consoles to which an operator may log on. Operators are still required to provide a password while logging on to each console.

Consoles password phrase support becomes enabled on a system when the security profile is defined. There is no authority access checking from a user ID perspective.

The consoles function checks for the existence of a security profile in the OPERCMDS class to cover the MVS.CONSOLE.PASSWORDPHRASE.CHECK resource.

For example, the following RACF command can be used to define the profile:

```
REDEFINE OPERCMDS (MVS.CONSOLE.PASSWORDPHRASE.CHECK)
```

If the profile exists, the new LOGON panel display is revealed which will allow for either the new password phrase input or the standard eight (8) character passwords.

After enabling password phrases, active consoles need to be recycled to pick up the setting. If the console is not recycled, the 8-character password processing remains in effect for that console. There are several ways to recycle the console so the new password state is used:

- Place the console in standby mode (VARY CN(*),STANDBY) and then take the console out of standby mode by pressing the enter key on the console.
- Vary the console offline (VARY CN(*cnname*),OFFLINE) and then back online (VARY CN(*cnname*),ONLINE). Note that the online request must be made from another active console.
- Re-IPL the system.
- Note that SMCS consoles do not support standby, so they must be logged off and then reconnected to z/OS.

Note that during the process of an operator logging on, z/OS may issue messages referring to passwords. In these messages, passwords mean either passwords (8-byte variety) or password phrases.

If your installation uses extended MCS consoles, you need to plan for their security. Your TSO or 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 159.

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), security profiles are used. The security administrator can define a security 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 that 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 security 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 security group. For example, if you have several operators or TSO/E users with identical access requirements, you can have the security administrator create a security group and define the access for the individual operators or TSO/E users through the group. For more information using RACF, see "Defining users with RACF" on page 58.

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 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 62.

Resources, such as commands, MCS or SMCS consoles, and TSO terminals, also require security 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 using RACF, see "Defining commands with RACF" on page 59 and "Defining consoles with RACF" on page 60. For an example of defining a TSO/E terminal as a resource, see "Controlling extended MCS consoles using RACF" on page 159.

You need to work with the security administrator to set up the security 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, HMCS, SMCS and extended MCS consoles using RACF commands. (See "Controlling extended MCS consoles using RACF" on page 159.)

Defining users with RACF

Your installation's security policy determines how you define the operators, MCS consoles, HMCS 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 must be defined as individual users. If your installation uses the LOGON(AUTO) option in CONSOLxx to automatically log on MCS, HMCS 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, HMCS 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 159.

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

MVS commands, RACF access authorities, and resource names in *z/OS MVS System Commands* 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 MVS commands, RACF access authorities, and resource names, 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 (CON1 is defined in CONSOL*xx*), 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, HMCS 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 62.
- Require each operator to log on to the system before issuing commands. See "Required LOGON" on page 62.
- Allow MCS console command authorization to control access to commands. See "Optional LOGON" on page 63.

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.

  Note: If an operator has not logged on to the console, commands are passed to the security product indicating an operator id of *BYPASS*.

The LOGON keyword affects only full-capability display consoles. It does not prevent the operator from receiving synchronous messages. However, the LOGON keyword setting may prevent the operator from receiving synchronous write-to-operator-with-reply (WTOR) messages. For details, see individual topics about LOGON values: "Automatic LOGON" on page 62, "Required LOGON" on page 62, and "Optional LOGON" on page 63.

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, HMCS or SMCS consoles" on page 61.

Setting LOGON requirements for individual MCS, HMCS or SMCS consoles

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

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

LOGON
: Controls the logon for operators of MCS, HMCS 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.

  Note: If an operator has not logged on to the console, commands are passed to the security product indicating an operator id of *BYPASS*.

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.

Note: For an SMCS console, see "Defining SMCS consoles" on page 33.

Automatic Timeout Processing

The CONSOLE statement in CONSOLxx supports a TIMEOUT keyword that causes the system to automatically issue a LOGOFF command as a result of console input inactivity. Examples of console input are the pressing of attention generating keys such as enter, PFK, PA1, or PA2. Consider using this function if you are concerned about unattended consoles.

Note: The TIMEOUT function is ignored for consoles in LOGON(AUTO) mode where the user ID is the same as the console name. It is possible, on a LOGON(AUTO) console, for an operator to specify a user ID that is different from the console name. In such cases, the TIMEOUT function is active for the console.

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, HMCS 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.

Synchronous WTORs can be displayed on LOGON(AUTO) consoles only after the consoles have been logged on.

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 update a CONSOLxx parmlib member and use the **`SET CON=`** command to activate it.. You can use the VARY CN command to change LOGON(REQUIRED) on the CONSOLE statement.

During system initialization, the system accepts commands only from a master authority console (or the system console) until RACF is fully initialized and able to process LOGON requests. Allowing commands from a master authority 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.

Note: If the TIMEOUT value is specified in CONSOLxx, the console will automatically log off after the specified number of minutes of inactivity is reached.

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.

Synchronous WTORs can be displayed on LOGON(REQUIRED) consoles only after the consoles have been logged on.

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.

Note: If an operator has not logged on to the console, commands are passed to the security product indicating an operator id of *BYPASS*.

Synchronous WTORs can be displayed on LOGON(OPTIONAL) consoles that are active, even if the consoles are not logged on.

MVS commands, RACF access authorities, and resource names

For lists of all MVS commands, the RACF access authority associated with them, and the RACF resource name for the profile, see the table MVS commands, RACF access authorities, and resource names in Defining and changing console characteristics of *z/OS MVS System Commands*.

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, HMCS or SMCS console authority, you can authorize specific commands issued from an MCS, HMCS or SMCS console through the command installation exit. You can specify command installation exits in MPFLSTxx. See "Command installation exits" on page 107.

Planning console functions for operators

CONSOLxx allows you to plan MCS, HMCS 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
- The presence of 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. To accept commands and receive messages, your console must be a full-capability console. 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, HMCS or SMCS console. An out-of-line display area is a specified part of the screen that can receive status displays separate from the other messages that appear. Output from certain commands like 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, HMCS or SMCS consoles. Establishing PFKs for your MCS, HMCS 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, HMCS 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 on page 66 shows screens of a full-capability, status display, and message-stream console:

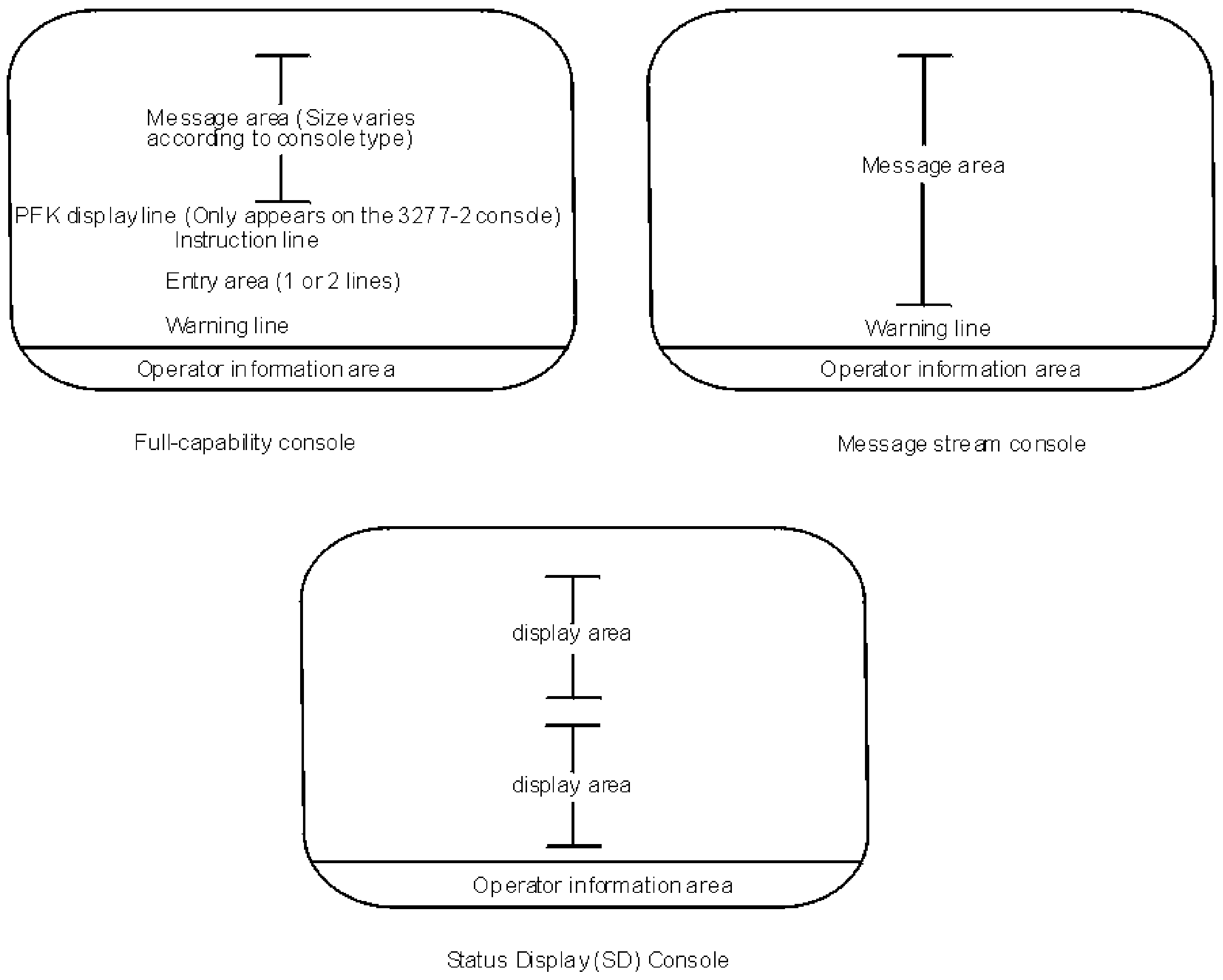

Message area (Size varies according to console type)
PFK display line (Only appears on the 3277-2 console)
Instruction line
Entry area (1 or 2 lines)
Warning line
Operator information area
Full-capability console
Message area
Warning line
Operator information area
Message stream console
display area
display area
Operator information area
Status Display (SD) 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, HMCS 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, HMCS or SMCS consoles" on page 67 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, HMCS 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 70 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 71 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 102 describes message retention of action messages.

Specifying automatic message deletion for MCS, HMCS 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.

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 75.

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 69.) 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=C3E0SS1   DEVNUM=0FE  SYS=SYSTEM1  CMDSYS=SYSTEM1  USERID=JIM

IEE163I MODE=W
```

Figure 9. Example of a Full Wrap Mode Screen

Figure 10 on page 69 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=C3E0SS1   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. An RTME value of 1/4 or 1/2 second is good for a console in wrap mode. You should verify that your console can handle a rapid update rate before setting the RTME value to 1/4 or 1/2 second. 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

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, HMCS 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 12 on page 71 shows a comparison of roll mode, roll-deletable mode, and wrap mode, and options you can specify including HOLDMODE.

Table 12. 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) and either DEL=NO or DEL=R/RD is not specified	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, perform the cancel operation (PA2).	1. If DEL(N), enter CONTROL D,N to display message line numbers 2. Enter CONTROL E,*line number* 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 *z/OS MVS System Commands*.) 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)	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.	1. Enter CONTROL D,N to display message line numbers (if DEL(N)) 2. Enter CONTROL E,*line number* 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 *z/OS MVS System Commands*.) 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.

Note:

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, HMCS 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 HMCS and SMCS consoles, which must be in full capability mode, in response to certain subsystem commands or the following MVS commands:

- DISPLAY
- CONFIG
- DEVSERV

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" on page 74 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.

You can route the output of the DISPLAY, CONFIG, or DEVSERV 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.

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

"Where to route status displays" on page 75 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 11 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, HMCS console or SMCS console. The HMCS console and 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 on page 74 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. After IPL, operators can route status displays using the location operand of the DISPLAY and CONFIG commands to area A or B, or to the general message area.

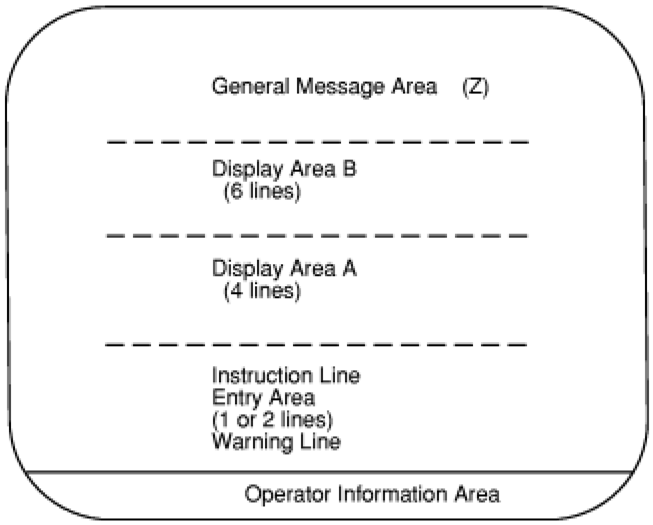


Figure 11. Sample Screen Showing Two Out-of-Line Display Areas on a Full-Capability Console

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

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

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

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, HMCS or SMCS console display

Defining the X option for an MCS, HMCS 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, HMCS 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 HMCS console, devnum must be HMCS. 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).

Use of MONITOR can affect the volume of messages that are produced and you should carefully consider which of the MONITOR messages you need and which you do not.

You can also use the SETCON MONITOR command to enable the production of MONITOR related messages for automation or logging purposes. See "Enabling message monitoring" on page 109 and the SETCON MONITOR command of *z/OS MVS System Commands*.

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, HMCS 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, HMCS 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 the PFKTABxx member of 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=BATT_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.

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

```
IEA180I  USING IBM DEFAULT PFK DEFINITIONS. NO PFK TABLES REQUESTED.
```

IBM supplies defaults for PFKs 1 through 8 in IEESPFK in SAMPLIB.

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 8. 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 on page 79 shows the PFKTAB01 Parmlib member. It contains three tables: MVSCMDS, JES2CMDS, and TLCMDS.

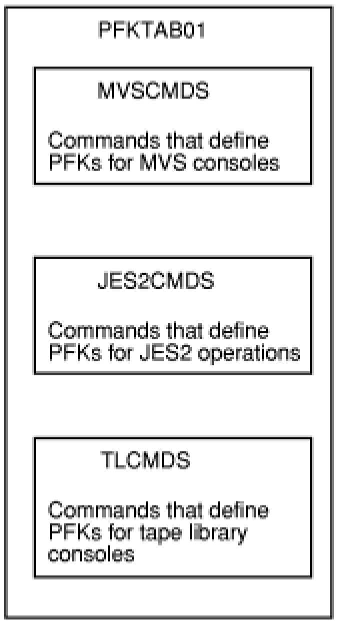

PFKTAB01
MVSCMDS
Commands that define PFKs for MVS consoles
JES2CMDS
Commands that define PFKs for JES2 operations
TLCMDS
Commands that define PFKs for tape library consoles

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:

CMDDELIM

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) or the operations log (OPERLOG). The group of messages and commands that is recorded is called the hardcopy message set. The system log or operations log is called the hardcopy medium.

Hardcopy processing is required in a sysplex.

Note: The term “hardcopy log” can refer to:

- The system log (SYSLOG)
- The operations log (OPERLOG)

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. The hardcopy message set is usually sent to the current active log, either the system log or the operations log, or both. The hardcopy message set is defined at system initialization and may subsequently be changed by the VARY command.

Characteristics of 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. The hardcopy message set includes messages with one or more of the following characteristics. Messages in the hardcopy message set:

- Are command responses that match the setting of CMDLEVEL on the HARDCOPY statement in the CONSOLxx member of Parmlib (see *z/OS MVS Initialization and Tuning Reference*). The CMDLEVEL setting can be modified by using the operator command V <hardcopy>,HARDCPY,CMDLEVEL=command (see *z/OS MVS System Commands*).
- Unsolicited system messages that have one (or more) of the routing codes specified on the ROUTCODE option of the HARDCOPY statement in the CONSOLxx member of Parmlib. The hardcopy routing code set can be modified by using the operator command V <hardcopy>,HARDCPY,ROUT=command.

- 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 a message type specified.

Messages for which “no hardcopy” is requested are not included in the hardcopy message set, regardless of their other characteristics.

Extended MCS Consoles (EMCS) have the ability to request messages that match the hardcopy message set be sent to that EMCS console.

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 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 the system log (SYSLOG) or the operations log (OPERLOG) at system initialization using the DEVNUM keyword 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 only enter the VARY HARDCPY command to change a hardcopy device from MCS, HMCS , 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 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 (SYSLOG records, or OPERLOG records that have been converted to the SYSLOG format), use the IHAHCLOG mapping macro. 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 topic 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 the sample program 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.

Duplexing the OPERLOG logstream to coupling datasets (STG_DUPLEX = YES) increases the recoverability of OPERLOG data in the event of a coupling facility failure. However, it also increases overhead, slowing the servicing of OPERLOG synchronous write requests to the logstream.

During a message flood, the slower synchronous write performance associated with STG_DUPLEX = YES increases the likelihood of reaching the OPERLOG extended console QLIMIT, resulting in the disabling of the OPERLOG EMCS console and OPERLOG data loss.

While a coupling facility failure without staging datasets (STG_DUPLEX = NO) makes recovering the OPERLOG logstream more difficult, messages lost from the OPERLOG logstream will alternatively be available in SYSLOG, provided it is active as a hardcopy medium.

Installations running with both SYSLOG and OPERLOG as hardcopy media should consider the performance benefit of running without OPERLOG staging datasets (STG_DUPLEX = NO).

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 80 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 your SYSLOG analysis programs to handle MDBs. 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. Information contained in the MDB but not in a SYSLOG record, such as descriptor codes, are lost during this conversion.

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.

SYSLOG MPF flags

The MPF Request Flag field in the SYSLOG is an 8-character hexadecimal field located at offset 46 in each SYSLOG record (a zero-relative offset is assumed).

The following information is from the IHAHCLOG member of SYS1.MODGEN, which maps a SYSLOG record.

The following hexadecimal values in the 8-character field are pertinent:

```
        xxxxxxxx
        8....... - message text changed
        4....... - route codes changed
        2....... - descriptor codes changed
        1....... - message queued to specific active console

        .8...... - unused
        .4...... - message queued by route codes only
        .2...... - console ID was changed
        .1...... - minor lines of MLWTO processed by exit

        ..8..... - the message was deleted (this never appears!)
        ..4..... - suppression was overridden (DISPLAY was forced)
        ..2..... - forced to hardcopy (LOG forced)
        ..1..... - bypass hardcopy (this never appears!)

        ...8.... - forced to hardcopy only (LOG and NODISPLAY)
        ...4.... - broadcast
        ...2.... - don't broadcast
        ...1.... - don't retain in AMRF (NORETAIN)

        ....8... - retain in AMRF (RETAIN)
        ....4... - retrieval key changed
        ....2... - 4-byte console ID changed
        ....1... - message type changed

        .....8.. - do not automate (set by MPFLSTxx and NOAUTO)
        .....4.. - automate (set by MPFLSTxx)
        .....2.. - message issued hardcopy-only
        .....1.. - unused

        ......8. - message not serviced by any exit
        ......4. - ESTAE error in IEAVX600
        ......2. - message not serviced, incompatible request
        ......1. - automation requested (AUTO)

        .......8 - Message Flood Automation processed this message
        .......4 - suppressed by subsystem
        .......2 - suppressed by exit (NODISPLAY)
        .......1 - suppressed by MPF (set by MPFLSTxx and NODISPLAY)or Message
                   Flood Automation
```

Note that in any column, these hexadecimal values are additive. For example, in the case of message flood automation, if you request LOG, NODISPLAY, NOAUTO, you get a field containing '00080A09' which is interpreted as:

- message forced to hardcopy-only

- don't automate message
- message issued to hardcopy-only
- Message Flood Automation processed this message
- message suppressed by MPF

Hardcopy failure

Table 13 on page 84 demonstrates what happens when either SYSLOG or OPERLOG are active/inactive and one or the other fails. Use this table in configuring your desired backup for hardcopy.

Table 13. Hardcopy failure backup configurations.

SYSLOG	OPERLOG	If SYSLOG fails	If OPERLOG fails
Active	Active	No SYSLOG. Hardcopy processing available on OPERLOG	No OPERLOG. Hardcopy processing available on SYSLOG
Active	Inactive	Hardcopy processing is suspended	N/A
Inactive	Active	N/A	Hardcopy processing is suspended
Inactive	Inactive	N/A	N/A

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 are held until there is no storage available for them. 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.

A parameter is available to turn off hardcopy. The parameter, UNCOND, is coded on the VARY command. For example:

```
VARY SYSLOG,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 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.

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. The SETCON MONITOR command lets you produce monitored messages for automation purposes or for logging purposes without requiring that the messages be queued to a console. The message flooding automation as described in Chapter 4, “Message flooding,” on page 121 can help address the message flooding problem on z/OS.

This topic 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 topic 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. Routing code functions include:

- Indicating messages that require a specific operator action, such as routing code 1 (primary operator action).
- Indicating messages that convey information about a specific system function or operator area. For example, messages with routing code 5 convey information about the tape library.
- Indicating 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.
- Indicating a more specific meaning. For complete information, see any volume of *z/OS MVS System Messages*.

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 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.

The system sometimes issues synchronous messages that bypass normal message delivery. 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 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 process the message instead of 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, CNZ_WTOMDBEXIT exit receives control from the system for each single-line WTO, multi-line WTO, or WTOR (see “CNZ_WTOMDBEXIT exit” on page 109 for more information). Then 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 program 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. "CNZ_MSGTOSYSLOG exit" on page 108 is invoked during the hardcopy log processing.
5. "CNZ_WTOMDBEXIT exit" on page 109 receives the message. See *z/OS MVS Installation Exits* for more information.
6. 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 in too short a time), command flooding occurs.

Attached commands that run in the *MASTER* or CONSOLE address space are divided into six "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 and command flooding, see *z/OS MVS System Commands*
- For information about the IEECMDS macro, see *z/OS MVS Programming: Authorized Assembler Services Reference EDT-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 6.) 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. A program 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. "CNZ_MSGTOSYSLOG exit" on page 108 is invoked during the hardcopy log processing.
5. "CNZ_WTOMDBEXIT exit" on page 109 receives the message. See *z/OS MVS Installation Exits* for more information.
6. Sysplex services directs the message to the other systems in the sysplex.
7. The subsystems on each receiving system can receive the message.
8. The MCS, SMCS and extended MCS consoles on the appropriate system can select 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 such that critical console function is available on more than one system.

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.

Commands in a sysplex

For commands in a sysplex, you need to consider:

- Command scope in the sysplex. Some commands only affect the system that they are processed on; other commands may affect all of the systems in the sysplex regardless of which system they are processed on. See *z/OS MVS System Commands* for a discussion of which commands have a sysplex-wide scope and which have a system scope. You can route commands with a system scope to the system on which that you want them to be processed.
- Type of command routing. You can route commands to one or more systems in the sysplex for processing.

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 96.) 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 6.) 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 97.) CPF allows you to define prefixes for commands so that the operator with sufficient authority 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 96 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 97.) 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 90.
- 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 on the system that receives the command processes it.
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 level. 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.

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 48 describes how you can define console groups for synchronous messages.

Altering some console attributes might cause messages to no longer be displayed on consoles. Messages that are not displayed on a console will still be logged in SYSLOG and/or OPERLOG, and are viewable using facilities such as SDSF.

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

VARY CN
VARY CONSOLE
CONTROL V,LEVEL

Note: In the case of an instream or internal command, consoles receiving the INTIDS routing attribute receive the command response.

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 the messages a console receives 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 94.

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) means that the system assigns no routing codes 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 any volume of *z/OS MVS System Messages*

For every routing code (except routing code 11), you should ensure that there is a receiving console. (Operator consoles should 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 CONS2, code the following CONSOLE statement in the CONSOLxx member:

```
CONSOLE DEVNUM(81D) NAME(CONS2) 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 159.

Operators can use the VARY CN command to change routing codes for active MCS, SMCS, 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.

IBM recommends that you do not specify ROUTCODE(ALL) or include routing code 11. Code ROUTCODE with a small number of routing codes so that you do not send these messages to all of your consoles.

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.

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 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.

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 from which this console can receive messages.

The default is MSCOPE(*ALL) (except for the system console, for which the default is MSCOPE(*)), which indicates that messages from the local system as well as all the 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 unsolicited 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 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 automation attribute through the MCSOPER macro or RACF using the OPERPARM parameter list or OPERPARM segment. In a sysplex, you can also define which systems are to direct the messages to the extended MCS console by specifying the MSCOPE attribute in the OPERPARM parameter list or OPERPARM segment.

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 5. 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*.

Receiving messages that are directed to console ID zero

You can use the INTIDS keyword on the CONSOLE statement to control whether the console is to receive messages that are directed to console id zero. Those messages are often the responses to internally issued commands. To enable a console to receive such messages, specify INTIDS(Y) for this console in the CONSOLE statement. You can also use the VARY CN command to change this console attribute. If you do not specify this attribute, the default is N.

Note:

1. Be aware that WTORs directed to console ID zero will also be delivered to consoles that specify INTIDS(Y). This means that the operator at that console will be able to reply to the WTOR.
2. Multiline messages directed at an Out-Of-Line (OOL) area on a console defined to receive INTIDS will be forced "inline".

Receiving messages that are directed to unknown console IDs

You can use the UNKNIDS keyword on the CONSOLE statement to control whether the console is to receive messages that are directed to unknown console IDs, such as one-byte console IDs which are no longer supported. To enable a console to receive such messages, specify UNKNIDS(Y) for this console on the CONSOLE statement. You can also use the VARY CN command to change this console attribute. If you do not specify this attribute, the default is N.

The UNKNIDS queuing attribute allows messages issued with 1-byte console IDs to be queued to some console. Messages marked with the UNKNIDS queuing attribute will be queued to any console that requests UNKNIDS messages. Note that this may or may not be the console that would have corresponded to the 1-byte console ID.

Note: Multiline messages directed at an Out-Of-Line (OOL) area on a console defined to receive UNKNIDS will be forced "inline".

Routing commands

In a sysplex, you can route commands to other systems for processing in the following ways:

- "Using CMDSYS on the CONSOLE statement" on page 96
- "Using the ROUTE command" on page 96
- "Using the command prefix facility" on page 97
- "Using the L=Operand on certain commands" on page 97

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, HMCS 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)
```

Commands that ignore the CMDSYS specification include:

- All CONTROL commands except for CONTROL M
- LOGON/LOGOFF
- ROUTE

For examples of how these commands operate in a sysplex, see "Commands in a sysplex" on page 89.

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, to all systems but the system, 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, ROUTE *OTHER 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, ROUTE *OTHER 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 for IEECMDPF 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 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 topic 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 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 z/OS 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 96 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 96 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 96 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=name-a** 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=name-a** 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, HMCS or SMCS console
- The suppression of messages
- The retention of messages for the action message retention facility
- Which subsystems, if any, are to receive foreign messages.
- 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.
- All subsystems receive foreign messages and DOMs.

Using MPF to handle foreign messages

Using MPFLSTxx you can specify whether you want subsystems to receive foreign messages and DOMs (messages and DOMs from another system). By reducing the number of messages through the SSI, the installation can reduce the amount of CPU utilized for these foreign messages.

Use the following statement and one of its option in MPFLSTxx to specify whether you want subsystems to receive foreign messages and DOMs:

.FORNSSI
Specifies that you want to define whether subsystems should receive foreign messages and DOMs.

The options that you can specify for .FORNSSI are as follows:

- ***ALL** - All subsystems will receive foreign messages and DOMs.
- ***NONE** - No subsystems will receive foreign messages and DOMs.
- **NOCHANGE** - The system should retain the previous FORNSSI statement.
- (list of subsystems) - Names of one or more subsystems that are to receive foreign messages and DOMs. Subsystems not in this list do not receive foreign messages and DOMs.

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" on page 101
- For message suppression options, see "Suppressing messages" on page 102
- For message retention options, see "Retaining messages" on page 102
- For message automation options, see "Selecting messages for automation" on page 105
- For message and command processing exits, see "Installation exits for messages and commands" on page 106.

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 message 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 processing. 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(YES) will not suppress the message if it is a command response. SUP(NO) indicates that you do not want to suppress the message(s) for display. You can use SUP(ALL) 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 (see "Displaying information about messages awaiting action" on page 103).

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

CON1
MCS master authority console attached to SY1

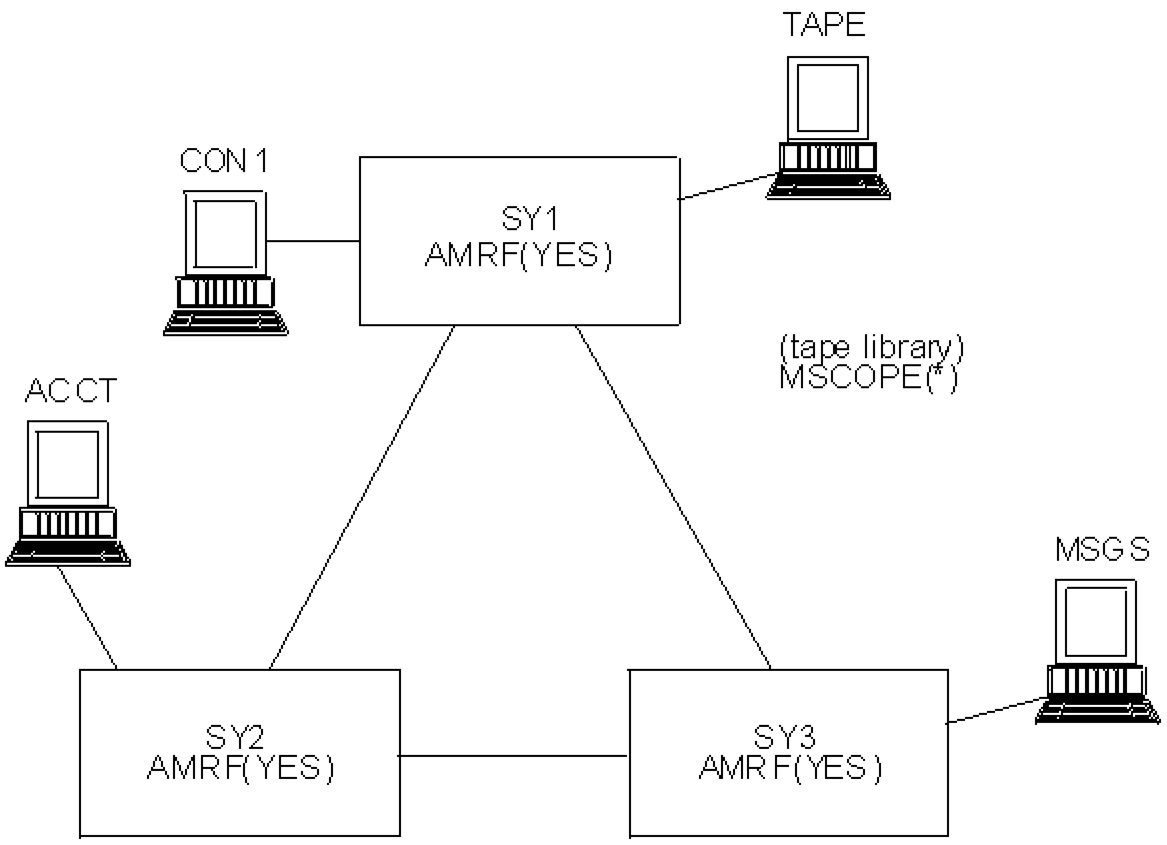
ACCT
MCS console attached to SY2

MSGS
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 CON1, enter the following command at CON1:

```
DISPLAY R,M
```

- To learn the number of outstanding action messages whose routing codes match those assigned to CON1, enter the following command at CON1:

```
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 CON1.

- To see all outstanding action messages in the sysplex, enter the following command at CON1, 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 95.

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 43.)

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 5.)

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 87.

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 16 and "MPF and MVS operations planning" on page 100.)

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 87.)

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" on page 106 and "Message and command flow in a sysplex" on page 87.)

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 specify your own message processing exit through MPFLSTxx, IEAVMXIT will get control if it is available and active when any WTO or WTOR message is issued. See "Message processing exits other than IEAVMXIT" on page 107.

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 the CONTROL M command 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 89.

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).

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 topic on sharing system commands in *z/OS MVS System Commands* for more information about using system symbols in commands.

CNZ_MSGTOSYSLOG exit

The Message to SYSLOG installation exit CNZ_MSGTOSYSLOG gets control for every message sent to the SYSLOG. Every message line that is sent to SYSLOG will be passed to the exit routines active at the exit point. Multi-line messages will be presented as a major line first, then major and each minor (one at a time). For example, a multi-line message with one major and three minor lines will result in the exit routines receiving control four times:

1. First time - For the major line
2. Second time - For the first minor line
3. Third time - For the second minor line
4. Fourth time - For the last minor line.

Code a CNZ_MSGTOSYSLOG exit routine when you want to view all messages being sent to the SYSLOG. IBM has defined the CNZ_MSGTOSYSLOG exit to the dynamic exits facility. You can refer to the exit by the name CNZ_MSGTOSYSLOG. You can use the EXIT statement of the PROGxx parmlib member, the SETPROG EXIT operator command, or the CSVDYNEX macro to control this exit and its exit routines.

Note: This exit is not able to change messages.

See *z/OS MVS Installation Exits* for a detailed description of the CNZ_MSGTOSYSLOG installation exit and a sample exit.

CNZ_WTOMDBEXIT exit

CNZ_WTOMDBEXIT receives control from the system for each single-line WTO, multi-line WTO, or WTOR. Every single-line message that is sent by WTO or WTOR will be passed to the exit routines active at the exit point. Multi-line messages will be presented only when all lines have been completed. For example, a multi-line message with 1 major and 3 minor lines will result in the exit routine receiving control one time.

Code a CNZ_WTOMDBEXIT exit routine when you want to view all messages being sent by WTO or WTOR. Information in the message is read-only; you cannot modify the message contents.

See *z/OS MVS Installation Exits* for a detailed description of the CNZ_WTOMDBEXIT installation exit and a sample exit.

Monitoring messages

Monitor messages display information about jobnames, data set names, and other status information. Use the MONITOR command to generate these messages, which appear on the console. If you want the system to generate monitor message for automation or logging purposes, but do not want the messages to appear on the console or to be written to SYSLOG and/or OPERLOG, use the SETCON MONITOR command. The DISPLAY OPDATA,MONITOR command provides information about the monitoring enablement status of the message types supported.

Enabling message monitoring

To enable monitor message production for a particular message type without sending the messages to a specific console, do one of the following:

- Issue the SETCON MONITOR command
- At IPL time, specify the SETCON MONITOR command in COMMNDxx
- For SPACE and DSNAME, specify the MONITOR keyword on the INIT statement of CONSOLxx

To enable monitor message production and identify the console to receive the monitored messages, do one of the following:

- Issue the MONITOR command
- For JOBNAMES, SESS, and STATUS, specify the MONITOR keyword on the CONSOLE statement of CONSOLxx
- For an EMCS console, use the OPERPARM parameter of the MSCOPER service when activating the EMCS console.

The MONITOR command enables monitor message production if it has not already been enabled with the SETCON MONITOR command.

Disabling message monitoring

If the SETCON MONITOR command was used to enable monitor message production, issue the SETCON MONITOR command to disable it for a particular message type. Note that the monitor message production is disabled only if there are no consoles currently receiving that message type.

If the MONITOR command was used to enable monitor message production, issue the STOP MONITOR command to disable the function. The STOP MONITOR command stops monitor message production only if it was enabled with the MONITOR command and there are no other consoles listening for that message type. If monitor message production was enabled with the SETCON MONITOR command, message production continues even if the messages are not being delivered to the console.

Controlling WTO and WTOR message buffers

MVS places WTORs in buffers in virtual storage. MVS also places WTOs queued to MCS, HMCS or SMCS consoles in 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. It is suggested that you use a minimum value of 99 for RLIM. 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. It is suggested the you use a value of 9999 for RMAX.

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 system will hold the message in a staging area until an endline is received. Unended multiline messages will not be delivered to a console until an endline is received. The system monitors unended multi-line WTOs and will detect when a time interval threshold has been exceeded. This threshold specifies the number of seconds that the system will wait before truncating an unended multiline WTO that has not received a connecting WTO. When this time interval threshold is reached, the system ends 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 input statements:

```
NAME IEANUC01 IEEUCMC
VER 01B8 001E               /* verify currently 30 seconds */
REP 01B8 002D               /* alter setting to 45 seconds */
```

IBM recommends altering the value using a formal SMP/E ++USERMOD so that SMP/E can track the modification. Use the following statements to make the alteration using SMP/E:

```
++USERMOD(TIMEOUT)          /* USERMOD name of your choice */
++VER(Z038) FMID(HBB7750)   /* FMID of your system */
++ZAP(IEECVUCM),
NAME IEANUC01 IEEUCMC
VER 01B8 001E               /* verify currently 30 seconds */
REP 01B8 002D               /* alter setting to 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 ROUTTIME parameter on the INIT statement in CONSOLxx. This affects the entire sysplex.
- Change the current ROUTTIME 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.

See *z/OS MVS Initialization and Tuning Reference* for more details about CONSOLxx, and *z/OS MVS System Commands* about the CONTROL and ROUTE command.

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 ROUTTIME 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=00.26.27 DATE=2006.060  UTC: TIME=04.26.27
      DATE=2006.060
     - SYS1  ro sys2,d t
   B SYS2  IEE136I LOCAL: TIME=00.26.33 DATE=2006.060  UTC: TIME=04.26.33
      DATE=2006.060
     - SYS1  ro t=0,*all,d t
   C SYS1  IEE136I LOCAL: TIME=00.26.45 DATE=2006.060  UTC: TIME=04.26.45
      DATE=2006.060
     SYS2  IEE136I LOCAL: TIME=00.26.45 DATE=2006.060  UTC: TIME=04.26.45
      DATE=2006.060
     SYS3  IEE136I LOCAL: TIME=00.26.45 DATE=2006.060  UTC: TIME=04.26.45
      DATE=2006.060















IEE612I CN=C3E0SS1  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:

```
     - SYS1  d t
       SYS1  IEE136I LOCAL: TIME=00.26.27 DATE=2006.060  UTC: TIME=04.26.27
        DATE=2006.060
     - SYS1  ro sys2,d t
       SYS2  IEE136I LOCAL: TIME=00.26.33 DATE=2006.060  UTC: TIME=04.26.33
        DATE=2006.060
     - SYS1  ro t=0,*all,d t
    A  SYS1  IEE136I LOCAL: TIME=00.26.45 DATE=2006.060  UTC: TIME=04.26.45
        DATE=2006.060
       SYS2  IEE136I LOCAL: TIME=00.26.45 DATE=2006.060  UTC: TIME=04.26.45
        DATE=2006.060
       SYS3  IEE136I LOCAL: TIME=00.26.45 DATE=2006.060  UTC: TIME=04.26.45
        DATE=2006.060
     - 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=00.29.41 DATE=2006.060  UTC:
         TIME=04.29.41 DATE=2006.060
 SYS2     IEE136I LOCAL: TIME=00.29.41 DATE=2006.060  UTC:
         TIME=04.29.41 DATE=2006.060
 SYS3     IEE136I LOCAL: TIME=00.29.41 DATE=2006.060  UTC:
         TIME=04.29.41 DATE=2006.060


IEE612I CN=C3E0SS1  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=C3E0SS1  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=C3E0SS1  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 SYSLOG. 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.

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 116.
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 116.
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 118 and "Using parmlib to control message translation" on page 118.
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 system

   See "Using parmlib to control message translation" on page 118.
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 119.

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 on page 116 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 on page 119 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	• 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.

Table 14. Summary of Message and Command Processing that MVS Provides (continued)

Service	**Features**	**Where specified**	**Scope**
MPF	• Suppress messages • Retain messages • Select messages for automation • Specify installation exits to process messages and commands	MPFLSTxx	System
Message Flood Automation	• Recognize message floods • Take action against flood messages	MSGFLDxx	System
AMRF	• Retain action messages	CONSOLxx INIT	Sysplex
IEAVMXIT	• Process messages • Control color, highlighting, and intensity of messages	CONSOLxx INIT	System
Installation-defined message processing exits	• Process messages selected through MPFLSTxx • Control color, highlighting, and intensity of messages	MPFLSTxx	System
Command Installation exits	• Process commands	MPFLSTxx	System
MMS	• Process messages for translation	CONSOLxx INIT	System

Chapter 4. Message flooding

Many z/OS systems are troubled by cases of message flooding, where a user program or a z/OS process itself issues a large number of messages to the z/OS consoles in a short time. For example, a user program might enter an unintentional loop that includes a WTO call, with the result that a potentially infinite number of messages are issued in a short time. Cases of hundreds (or even thousands) of messages a second are not uncommon.

The messages in a message flood are often similar or identical, but are not necessarily so. Techniques to identify similar messages can be difficult and time consuming.

z/OS Message Flood Automation

z/OS Message Flood Automation addresses the problems of message flooding on z/OS. z/OS Message Flood Automation does not claim to identify all cases of erroneous behavior, nor to take the 'correct' action in all cases. Its intention is to identify runaway WTO conditions that can cause severe disruptions to z/OS operation and to take installation-specified actions in these cases.

Message flooding causes disruptions as follows:

- Large numbers of messages to the z/OS consoles can obscure important messages and delay them from being acted on.
- Large numbers of messages to the automation system (for example, NetView) can delay the processing of normal messages.
- Messages can use excessive CPU and storage resources. Buffering excessive message traffic can use large amounts of virtual and real storage and it can cause SQA to overflow into CSA. This can cause jobs, subsystems and complete systems to be delayed or even to fail.

Message Flood Automation can react to potential message flooding situations in a matter of tens or hundreds of messages (specifiable by the installation), before buffers begin to fill, console queues begin to build, and console message rates begin to skyrocket. Furthermore, its actions do not result in residual buffers or queues of messages that must be worked down to return to normal processing. Because its processing is targeted to the messages causing the problem, very few uninvolved messages are acted upon. By contrast, the act of flushing console queues (with the K Q command) can result in throwing away innocent and often important messages.

Message Flood Automation cannot handle DOM (Delete Operator Message) floods.

Operation

Message Flood Automation is part of z/OS WTO processing. Message Flood Automation examines each message in the z/OS system, and attempts to identify when too many WTOs are being issued and by whom. It then takes appropriate actions: usually to suppress the message from display at a z/OS console, and to indicate that automation processing is not required. It can also issue commands, for example, to cancel the user or process.

Three separate classes of messages are handled. These classes are:

- SPECIFIC messages: a set of messages identified by the installation that are to be handled separately.
- ACTION messages: messages that have one or more of the following descriptor codes sets:

  1

  System failure messages (typically messages with a W message ID suffix)

  2

  Messages requiring immediate action (typically messages with an A or D message ID suffix)

3
: Messages requiring eventual action (typically messages with an E message ID suffix)

11
: Messages requiring critical eventual action (typically messages with an E message ID suffix)

- REGULAR messages: messages that do not fall into any of the above categories.

Each class of messages is handled separately. Each class has its own set of controls (MSGTHRESH, INTVLTIME, and so on). Each set of controls operate independently; for example, the system can be in intensive mode for regular messages but not for action messages. The effect is that z/OS still processes action messages in the normal way.

Message Flood Automation can take action against *privileged* messages that are queued to consoles even in storage shortage situations.

Message Flood Automation runs in two modes: *normal* and *intensive*.

- In *normal* mode, messages are counted. When a threshold number (MSGTHRESH) of messages have been counted, the *time* taken to count those messages is determined. If the time is less than a limit value (INTVLTIME), the system is placed into *intensive* mode. This determination is likely to be done infrequently, for example, every 50-100 messages or more. The INTVLTIME value should be set to identify high message rates, for example, a value of 5 seconds for INTVLTIME indicates an average rate of 20 messages per second if MSGTHRESH is set to 100.

  The processing overhead in *normal* mode is therefore very small. Only a small number of instructions are executed in Message Flood Automation for each message.
- In *intensive* mode, each message is subject to extra processing. Messages are counted for each address space (up to a maximum of 128) issuing messages and compared to a further limit value (JOBTHRESH). If any one address space issues JOBTHRESH messages within INTVLTIME, it is subject to action from that time on. This action can be installation-specified, but is typically defaulted to be no-display and no-automation.

  At the end of each interval of MSGTHRESH messages a check is made to see if *intensive* mode should be maintained, and whether address spaces in act-upon mode should remain so.

  Message bursts can end suddenly. The address space that issues them might suddenly exit a tight-loop condition and resume normal processing. In this circumstance, it is likely that subsequent messages are important and should be processed normally. To allow this to happen, there are two further controls: system inter-message time (SYSIMTIME) and job (or message) inter-message time (JOBIMTIME or MSGIMTIME).

  In intensive mode, if the time since the last message is greater than SYSIMTIME, then intensive mode is discontinued. This ensures that the first message after a break is not acted upon.

  Similarly, if an address space is in act-upon mode, and the time since its last message exceeds the JOBIMTIME, then it is removed from act-upon mode.

  For specific messages, if a message is in act-upon mode, and the time since the last message exceeds the MSGIMTIME, then the message is removed from act-upon mode.

The control algorithms for regular and action messages are identical as described previously. For specific messages, the control algorithm is similar although it is applied to individual messages and not to jobs or address spaces. The MSGLIMIT parameter performs the same function in specific message processing that the JOBTHRESH parameter performs in regular and action message processing. The MSGIMTIME parameter performs the same function in specific message processing that the JOBIMTIME parameter performs in regular and action message processing, although it is applied against specific messages rather than address spaces.

Message flood detection behavior

A message flood can begin when a message counter is anywhere between zero to MSGTHRESH; for example, zero, one, or equal to MSGTHRESH.

- If the message counter is zero, the first message of the flood will cause the timestamp to be stored and MSGTHRESH messages later (assuming this occurs in less than INTVLTIME), cause the threshold exceeded message to be issued and intensive mode to be entered. For this case, only MSGTHRESH messages are required to enter intensive mode.
- If the message counter is one, the timestamp marking the beginning of the interval *has already been stored*, and after MSGTHRESH-1 messages have been counted and the ending timestamp acquired, *the difference between the timestamps may not cause intensive mode to be entered*. If a flood is underway, the next MSGTHRESH number of flood messages will cause intensive mode to be entered. In this case, it will take 2 x MSGTHRESH - 1 flood messages to cause intensive mode to be entered.
- If the message counter is already at MSGTHRESH, the first flood message will cause the ending timestamp to be stored, and the difference in timestamps will probably cause intensive mode to not be entered. However, the next MSGTHRESH flood messages will cause intensive mode to be entered. So in this case, it will take MSGTHRESH + 1 messages to cause intensive mode to be entered.
- If the message counter is between one and MSGTHRESH, it will take 2 x MSGTHRESH - n messages to cause intensive mode to be entered, where "n" is the number of messages already counted.

The bottom line is that the triggering of intensive mode may not occur precisely after MSGTHRESH flood messages have occurred.

Message Flood Automation and CONSOLxx parameters

CONSOLxx members of PARMLIB contain statements that can affect the routing and hardcopying of messages.

During WTO processing, the route codes defined by the DEFAULT statement are applied to *unsolicited* messages which have been issued without route codes, without descriptor codes, or without a console ID or console name.

If the ROUTCODE parameter is not supplied on the DEFAULT statement, route codes 1-16 are applied.

Message Flood Automation can affect the processing of messages whose route codes are defined by DEFAULT processing just like messages that have been issued with route codes, descriptor codes or console routing information.

Specifying the MSGFLD parameter on the CONSOLxx INIT statement allows a MSGFLDxx parmlib member to be automatically loaded and optionally enabled during system initialization. This eliminates the need for an operator or automation to enter a SET MSGFLD=xx command, followed by a SETMF ON command to initialize Message Flood Automation. For more details, see *z/OS MVS Initialization and Tuning Reference*.

The HARDCOPY statement defines the route codes of the messages that are subject to being hardcopied. Messages with route codes 1, 2, 3, 4, 7, 8, 10 and 42 are always hardcopied whether the ROUTCODE parameter is supplied on the HARDCOPY statement or not.

Message Flood Automation can affect the hardcopying of messages that have been forced to hardcopy by HARDCOPY processing just like messages that were explicitly issued to hardcopy.

Message Flood Automation and MPFLSTxx parameters

MPFLSTxx members of PARMLIB contain statements that can affect the display, automation and retention of messages.

During MPF processing, the RETAIN, AUTO and SUP parameters on an MPFLSTxx entry are processed first. Then *either* a user exit (specified by the USEREXIT parameter) is invoked *or* IEAVMXIT is invoked — *but not both*.

Note that if an MPFLSTxx entry does not exist for a message, the settings from the NO_ENTRY specification are applied.

Message Flood Automation message processing runs before MPF exit processing. Therefore, MPF exits can override the RETAIN, AUTO and SUP specifications set by Message Flood Automation.

Message Flood Automation and the Subsystem Interface (SSI)

Message Flood Automation processing occurs before a message is placed onto the Subsystem Interface (SSI). Automation products such as NetView, which can obtain messages from the subsystem interface, see messages after Message Flood Automation have seen (and potentially modified) them. All requests to delete, log or queue messages are processed *after* return from the subsystem interface. Therefore, NetView and other automation products that sit on the subsystem interface can see the message and potentially copy or modify the message (possibly overriding Message Flood Automation decisions) *before* z/OS deletes, logs or queues the message.

The NetView 5.2 Message Revision Table (MRT) performs all of its message processing on the Subsystem Interface, after Message Flood Automation has processed the message. Message Revision Table logic can see and override message specification changes that Message Flood Automation requested. Changes to the message's specifications that the Message Revision Table requested can affect the logging, display, retention and automation of the message by z/OS.

NetView can obtain messages for automation through either the Subsystem Interface or an EMCS console interface or both. (If the NetView MSGIFAC parameter is set to SSIEXT, USESSI, QUESSI or QSSIAT, NetView will obtain messages for processing from the Subsystem Interface. If the MSGIFAC parameter is set to SSIEXT, only unsolicited messages are obtained from the SSI; command response messages are obtained through the EMCS console interfaces). When NetView obtains messages from the Subsystem Interface, it obtains a *copy* of the original message, before z/OS has an opportunity to delete, log or queue the message to a console. The original message is processed for deletion, logging and queuing *after* NetView has made its copy. Traditional (non-Message Revision Table) NetView automation can see but cannot alter changes to the message that Message Flood Automation made.

Message Flood Automation and EMCS consoles

Message Flood Automation processing occurs before a message is queued to Extended MCS (EMCS) consoles. Because EMCS console interfaces can be used by automation products such as NetView, there are special considerations:

- If a message flood occurs, and Message Flood Automation has been requested to suppress the message from display, the message is not queued to any EMCS console unless automation of the message has been requested (typically by specifying AUTO on the message's MPFLSTxx entry).
- If a message flood occurs, and Message Flood Automation has been requested to suppress the message from display and NOT automate it, the message is not queued to any EMCS console.

Note that the decisions to log, display or automate a message are independent decisions. It is possible (and might be desirable) to obtain messages at EMCS consoles for automation purposes without logging or displaying them.

If the NetView MSGIFAC parameter is set to SYSTEM, NetView obtains all messages for automation processing from the EMCS console interfaces. (If the MSGIFAC parameter is set to SSIEXT, NetView only obtains command response messages from the EMCS console interfaces; unsolicited messages are obtained from the SSI). The messages that NetView "sees" at the EMCS console interface have already been processed by both Message Flood Automation and z/OS deletion, logging and console queuing processing.

Limitations

Message Flood Automation has the following limitations:

- It only counts the first (major) line of multi-line messages.
- It does not handle branch-entry messages until they are re-issued as normal messages. It cannot affect them while they are being queued for re-issue.
- It specifically ignores IEF196I and IEF170I messages.

IEF170I
A write-to-programmer message operation failed. The IEF170I message includes the reason for the failure and 53 characters of the failing message's text.

IEF196I

A message from a task started under the Master Subsystem (MSTR) is being written to the system log because it could not be written to the system message data set or joblog data set. The IEF196I message includes the message ID and text of the failing message.

- It does not receive control for WTOR messages.

If Message Flood Automation does not take action against a message, it is often because of these restrictions.

Operator commands

The following operator commands are available to control Message Flood Automation. For more details, see *z/OS MVS System Commands*.

Table 15. Operator commands to control Message Flood Automation

Function	Command
Enable message flood checking.	SETMF ON
Disable message flood checking.	SETMF OFF
Enable the collection of message rate information.	SETMF MONITORON
Disable the collection of message rate information.	SETMF MONITOROFF
Re-initialize the counts, indicators and actions, and read the specified MSGFLDxx PARMLIB member.	SET MSGFLD=*xx*
Modify the counts and parameters used by Message Flood Automation.	SETMF MSGTYPE=msgtype, keyword=value[,keyword=value]
Display the counts and parameters used by Message Flood Automation.	D MSGFLD,PARAMETERS
Display the status of the Message Flood Automation function.	D MSGFLD,STATUS
Display whether intensive mode is active for the different classes of messages.	D MSGFLD,MODE
Display message rate information collected by the message rate monitoring function.	D MSGFLD,MSGRATE[,n][,m]

PARMLIB specifications

Installation policy for controlling message flooding situations is specified through the MSGFLDxx member of PARMLIB.

To provide a MSGFLDxx member in PARMLIB:

1. See the sample MSGFLDxx member as shown in the MSGFLDxx (message flood automation parameters) chapter of *z/OS MVS Initialization and Tuning Reference*. You should provide a MSGFLDxx member similar to it and place the member into a data set in the PARMLIB concatenation.
2. You might have as many MSGFLDxx PARMLIB members as you like but Message Flood Automation only supports one member to be active at a time. Message flood automation processing requires that the MSGFLDxx suffix *xx* be alphabetic, numeric or national characters. Other special characters are not supported.

The following statement types are provided. For more details about the statement and parameters, see *z/OS MVS Initialization and Tuning Reference*.

- comment statements
- msgtype statements

- DEFAULT statements
- DEFAULTCMD statements
- JOB statements
- MSG statements

SYSLOG records

The SYSLOG will contain information about the messages that Message Flood Automation processed. It will contain Message Flood Automation messages about the decisions it made and the actions it took. It will also contain information about the actions taken on individual messages (unless NOLOG was specified).

Each SYSLOG record is prefaced by a two-character record type field.

Valid first characters are:

- N - single-line message
- W - single-line message with reply
  - WTOR messages are not processed by Message Flood Automation.
- M - first line of a multi-line message
  - Message Flood Automation can only react to the first line of a multiline message, not to any of the label, data or end lines
- L - multi-line message label line
- D - multi-line message data line
- E - multi-line message data/end line
- S - continuation of previous line
- O - LOG command input
- X - non-hardcopy or LOG command source

Valid second characters are:

- C - command issued by operator
- R - command response message
- I - internally issued command
- U - command from unknown console ID (z/OS R8 and above)

SYSLOG message ordering

Message flood automation processing is driven by the issuance of a message. As soon as that message is created, it obtains a timestamp, and this occurs before the message is seen by Message Flood Automation. If Message Flood Automation makes a decision based on that message, it interrupts the processing of that message until it has taken whatever action it needs to take. Once that has occurred, processing of the original message by the operating system is then allowed to resume.

In the following example, the *second* IOS100I message caused Message Flood Automation to exceed the REGULAR message threshold. Further processing of the second IOS100I message was then suspended while:

- Message Flood Automation issued its CNZZ002E message
- Message Flood Automation took action against the second IOS100I message (as seen in its MPF flags)

The second IOS100I message was then allowed to continue, causing it to be written to the SYSLOG *after* the CNZZ002E message had been written to it.

```
11:14:58.79 ... 00000010  IOS100I DEVICE 891B BOXED, LAST PATH 75 LOST
11:14:58.91 ... 00000010  CNZZ002E MESSAGE THRESHOLD REACHED FOR JOB NONAME
11:14:58.79 ... 00080A09  IOS100I DEVICE 891C BOXED, LAST PATH 75 LOST
```

Recovery

If a failure occurs during message flood automation processing, a dump will be taken. Some failures might cause Message Flood Automation to be turned off and the policy to be reset to the default policy. It might be possible to reactivate Message Flood Automation using the SETMF ON command.

Other information

The Message Flood Automation software has not been tested with ISV software, including message automation products. Before activating the Message Flood Automation function, you need to assess whether there are possible interactions between Message Flood Automation and any ISV software you run. Use of Message Flood Automation with ISV software might require adjustments to Message Flood Automation policy, ISV policy or both and it is possible that Message Flood Automation cannot be used in conjunction with particular ISV software. Service is provided by IBM Support, Console Services, Level 2.

Migration

This section describes how to install, initialize and shut down z/OS Message Flood Automation.

Migrating from one level to another

As of z/OS Version 1 Release 12 Message Flood Automation is part of the z/OS operating system; therefore, you no longer require the IEAVMXIT ++USERMOD for Message Flood Automation and need to remove it. If you have already used the PTF for OA25602 on z/OS Version 1 Release 11 to implement the ++USERMOD for Message Flood Automation, do not reapply the ++USERMOD through the SMP/E RESTORE command on Version 1 Release 12. Review all Message Flood Automation user exits to verify the appropriate actions to take in order to ensure successful migration to the new version of Message Flood Automation for Release 12.

If you have been using Message Flood Automation and want to install a new level, take the following actions:

1. Remove Message Flood Automation processing from the MPF installation exit IEAVMXIT. If Message Flood Automation is the only user of IEAVMXIT, do the following:
   - Replace the UEXIT(Y) parameter with UEXIT(N) on the INIT statement with the CONSOLxx PARMLIB member.
   - Remove IEAVMXIT from the LINKLIST concatenation.

   If Message Flood Automation is not the only user of IEAVMXIT, do one of the following:
   - Fall back to the earlier version of the exit without Message Flood Automation invocations.
   - Remove the invocations of Message Flood Automation from the exit. Reassemble and rebind the exit to verify that the new exit has replaced the old exit in the LINKLIST concatentation.
   - Remove the invocation of Message Flood Automation from the CNZZVXT2 sample program. Reassemble and rebind the exit to verify that the new exit replaced the old exit in the LINKLIST concatenation.
   - Remove IEAVMXIT from the LINKLIST concatenation.
2. If Message Flood Automation is the only user of the command processing exit specified for USEREXIT on the MPFLSTxx .CMD statement, before you load the initial program of z/OS Version 1 Release 12, remove the .CMD statement from all MPFLSTxx members in which the exit has been specified. If Message Flood Automation is not the only user of the exit, remove the CNZZCMXT entry from all .CMD statements.
3. Remove all invocations of the SETMF FREE command from the system, including COMMNDxx and automation.
4. You are not required to make changes to the MSGFLDxx PARMLIB member. Because Message Flood Automation commands are now subject to authorization checking, you can define the Message Flood Automation commands to your security product. For profiles defined to security products, you can see

the table MVS Commands, RACF Access Authorities, and Resource Names in *z/OS MVS System Commands*.

Initializing Message Flood Automation

Issue a **SET MSGFLD=00** command from a z/OS console to cause the Message Flood Automation PARMLIB member MSGFLD00 to be read.

- You should see a message indicating that PARMLIB member MSGFLD00 is being loaded and another message indicating that the loading of PARMLIB member MSGFLD00 was complete.

To activate Message Flood Automation processing, issue a **SETMF ON** command.

- You should see a message indicating that Message Flood Automation is enabled.

Interpreting message rate information

The message rate information gathered is presented in multi-line message CNZZ043I:

- The total number of messages counted
- The total elapse time from when message rate monitoring was started to the current time
- The average message rate (in messages / second) from when message rate monitoring was started to the current time
- The number of messages occurring at the most common message rate
- A message rate distribution graph

The message rate distribution graph shows the percent of time at a given message rate on the Y-axis and instantaneous message rates (in messages/second) on the X-axis. The X-axis scale is logarithmic with each character position being a factor of 2 greater than the previous position in the rightward direction. Tick marks are provided at 8X intervals.

Each vertical bar of asterisks in the graph is rightward cumulative, that is, each bar represents not only the fraction of time at its own rate, but the fraction of time with a lesser rate. (A bar's own contribution to the time at a given message rate is therefore the difference between its height and the height of its immediate leftward neighbor).

A vertical line (|) indicates the most common message rate.

The graph should have a characteristic S shape to it caused by there being relatively few messages occurring at very low message rates (the bottom left of the S curve) and very few messages occurring at very high message rates (the top right of the S curve).

```
CNZZ043I     MSGFLD Message Rates
                 Instantaneous Message Rates
            515 messages in       492 seconds     1.046 msg/sec
% of time at msg rate              112 messages w/most common rate
     100.000%|           | ***********
      96.000%|           |************
      92.000%|           |************
      88.000%|           |************
      84.000%|           *************
      80.000%|           *************
      76.000%|           *************
      72.000%|           *************
      68.000%|           *************
      64.000%|           *************
      60.000%|          **************
      56.000%|          **************
      52.000%|          **************
      48.000%|          **************
      44.000%|          **************
      40.000%|          **************
      36.000%|          **************
      32.000%|         ***************
      28.000%|         ***************
      24.000%|         ***************
      20.000%|         ***************
      16.000%|        ****************
      12.000%|        ****************
```

```
       8.000%|      ******************
       4.000%|      ******************
            0+--+--+>-+--|--+--+---+-<+--------------------
             0           1  8 64  1K 8K  messages/second

             Suggested threshold for 95% is     2
             Suggested threshold for 96% is     3
             Suggested threshold for 97% is     3
             Suggested threshold for 98% is     4
             Suggested threshold for 99% is     6
```

This example was produced using a testcase that issued messages with an exponential distribution of inter-arrival times and a mean inter-arrival time of 0.5 seconds. The vertical bar indicates that the most common (mean) message rate is 1 messages/second. The average message rate is only slightly more than 1 message/second, a rate that has been determined by IBM human factor studies to be the maximum rate that messages should be presented on any one console.

On the X-axis, the minimum and maximum message rates recorded are indicated (by the > and < symbols respectively) on either side of the mean message rate. The percentage of messages occurring at the maximum message rate is usually quite small and may not be visible unless the resolution of the graph is improved by increasing the number of message lines in the graph.

The graph presents instantaneous message rates that are determined from the inter-arrival times of the messages. Small inter-arrival times result in high instantaneous message rates; large inter-arrival times result in low instantaneous message rates. A high message rate on the graph does not necessarily imply that multiple, consecutive messages were issued at that rate. It is quite possible (as in the example) for a high message rate to be indicated without Message Flood Automation being triggered. (It is multiple, consecutive, high message rate messages that trigger Message Flood Automation).

The suggested threshold values represent the message rates that are not exceeded some fraction of the time. In the example, a message rate of 4 messages/second is not exceeded 98% of the time; a message rate of 6 messages/second is not exceeded 99% of the time. You can use the suggested threshold values to set an appropriate REGULAR MSGTHRESH value.

Look at a more interesting graph:

```
CNZZ043I     MSGFLD Message Rates
                 Instantaneous Message Rates
          34299 messages in     78111 seconds     0.439 msg/sec
% of time at msg rate             5993 messages w/most common rate
     100.000%|         **********************
      96.000%|      *************************
      92.000%|     **************************
      88.000%|     **************************
      84.000%|    ***************************
      80.000%|    ***************************
      76.000%|    ***************************
      72.000%|   ****************************
      68.000%|   ****************************
      64.000%|   ****************************
      60.000%|   ****************************
      56.000%|   ****************************
      52.000%|   ****************************
      48.000%|  *****************************
      44.000%|  *****************************
      40.000%|  *****************************
      36.000%|  *****************************
      32.000%|  *****************************
      28.000%|  *****************************
      24.000%|  *****************************
      20.000%| ******************************
      16.000%| ******************************
      12.000%| ******************************
       8.000%| ******************************
       4.000%| ******************************
            0+->+--+--+--+--+--+--|+--+-----<-------------
             0           1  8 64  1K 8K  messages/second

             Suggested threshold for 95% is     1
             Suggested threshold for 96% is     1
             Suggested threshold for 97% is     1
             Suggested threshold for 98% is     1
             Suggested threshold for 99% is     1
```

This graph looks very different from the previous one. The first reaction of most people is to look at the average message rate of 0.439 messages per second and the fact that the most commonly occurring message has a rate of 512 messages per second and wonder how these two statistics can be reconciled. It is important to understand what an average can tell you and what it cannot. What an average can tell you is that (in this case) 34299 messages occurred during the 78111 second interval that was monitored. **What the average message rate cannot tell you is *how* those messages were distributed within the monitoring interval.** If the messages were distributed uniformly within the monitoring interval, the time between messages would be the same -- but a quick look at the graph shows this to not be the case: there were some number of messages that occurred at an instantaneous message rate of 1 message every 1024 seconds (at the left edge of the graph) and there were some number of messages that occurred at an instantaneous message rate of 262144 messages per second (at the right edge of the graph). And there were the 5993 "most commonly occurring" messages that occurred at a rate of 512 messages per second. The answer to this riddle is that one or more message "spikes" occurred at some point in the monitoring interval, and those spikes produced at least 5993 messages at a rate of 512 messages per second. Why doesn't this very high message rate affect the overall average message rate? Because, this very high message rate only occurred for 11.7 seconds (5993/512) -- which represents only 0.015% of the time within the interval of 78111 seconds.

The very broad "top" to the graph is indicative of a very small number of messages that occurred with very high instantaneous message rates. However, these messages occur for such brief periods of time that they have almost no effect on the overall message rate. The very broad "base" of the graph is indicative of a very non-uniform distribution of messages within the monitoring interval.

The "suggested thresholds" are all one because one is the lowest value that can be specified.

Setting thresholds based on message rates

The Message Rate Monitoring function measures the message rate for all messages that are subject to control by Message Flood Automation. The suggested thresholds provided in message CNZZ043I in response to a DISPLAY MSGFLD,MSGRATE command are good values to start with.

- Set your REGULAR message threshold (MSGTHRESH) value based on the suggested thresholds in the CNZZ043I message. IBM recommends that you use the 99% threshold value, but you may want to set the MSGTHRESH value higher.

  Inter-message time is the inverse of message rate: a message rate of 2.0 messages/second means that messages arrive on average every 0.5 seconds (so the inter-message time is 0.5 second). You should set your REGULAR message inter-message time (SYSIMTIME) at or slightly above the inverse of the REGULAR MSGTHRESH value (1/MSGTHRESH). For example, if the REGULAR MSGTHRESH value is set to 25, you should set the REGULAR message inter-message time (SYSIMTIME) value to 0.04 (1/25) or slightly higher.

  The REGULAR JOB message threshold (JOBTHRESH) **must** be set to a value less than that of MSGTHRESH. A JOBTHRESH value that is 30-40% of MSGTHRESH is a good starting point. This will allow you to handle 2-3 message flooding jobs simultaneously. A general "Rule of Thumb" is to take the MSGTHRESH value and divide by the number of jobs (less than 128) that you want Message Flood Automation to be able to handle simultaneously and use the result as the JOBTHRESH value.
- The message rate for ACTION messages is typically a small fraction of REGULAR messages, so your ACTION message threshold (MSGTHRESH) can be less than your REGULAR message threshold. (Setting the ACTION threshold lower than the REGULAR threshold does not increase your overhead because the ACTION messages occur less frequently.) Because the ACTION message rate is lower, the ACTION inter-message time (SYSIMTIME) can be greater than your REGULAR message inter-message time.

  The ACTION JOB message threshold (JOBTHRESH) must be set to a value less than that of MSGTHRESH. A JOBTHRESH value that is 30-40% of MSGTHRESH is a good starting point. This will allow you to handle 2-3 message flooding jobs simultaneously. A general "Rule of Thumb" is to take the MSGTHRESH value and divide by the number of jobs (less than 128) that you want Message Flood Automation to be able to handle simultaneously and use the result as the JOBTHRESH value.
- Unless you have chosen very common messages, the message rate for SPECIFIC messages is typically a small fraction of REGULAR messages, so your SPECIFIC message threshold (MSGTHRESH) can be

less than your REGULAR message threshold. Because the SPECIFIC message rate is lower, the SPECIFIC inter-message time (SYSIMTIME) can be greater than your REGULAR message inter-message times.

The SPECIFIC MSG message threshold (MSGLIMIT) must be set to a value less than that of MSGTHRESH. A MSGLIMIT value that is 15-20% of MSGTHRESH is a good starting point. This will allow you to handle 5-6 SPECIFIC message flooding messages simultaneously. A general "Rule of Thumb" is to take the MSGTHRESH value and divide by the number of messages (less than 1024) that you want Message Flood Automation to be able to handle simultaneously and use the result as the MSGLIMIT value.

The message rate specified by a threshold is also a function of the interval over which the threshold number of messages occurs. You can specify the same message rate through different combinations of the threshold and interval values. For example, setting MSGTHRESH=50 and INTVLTIME=1 specifies a message rate of 50 messages / second. Setting MSGTHRESH=100 and INTVLTIME=2 also specifies a message rate of 50 minutes / second. You may wish to choose which way you specify the message rates to achieve other goals:

- Using the MSGTHRESH=50 and INTVLTIME=1 specification will make Message Flood Automation more responsive to detecting message flooding situations because only 50 messages will be counted between computations of the message rate; however, the overhead of the message rate computation will be incurred twice as frequently as the MSGTHRESH=100 and INTVLTIME=2 specification.
- Using the MSGTHRESH=100 and INTVLTIME=2 specification will make Message Flood Automation less responsive to detecting message flooding situations because 100 messages will be counted between computations of the message rate; however, the overhead of the message rate computation will be incurred half as frequently as the MSGTHRESH=50 and INTVLTIME=1 specification.

You can use different combinations of threshold and interval to trade-off message flood detection responsiveness and message flood detection overhead.

The general idea is to set the various thresholds high enough that they are not being triggered by normal fluctuations in message rates but are triggered when sudden, very high message rates are encountered. For REGULAR messages, using one of the suggested threshold values provided by the CNZZ043I message is a good first approximation. You should set your thresholds high enough that Message Flood Automation is not constantly oscillating into and out of intensive mode. Receiving message CNZZ001I is usually a good indication that you have set the REGULAR message threshold too low; receiving message CNZZ019I is usually a good indication that you have set the ACTION message threshold too low.

Shutting down Message Flood Automation

Issue a **SETMF OFF** command from a z/OS console to disable Message Flood Automation.

- You should see a message indicating that message flood automation was disabled.
- You can re-enable message flood automation by issuing a **SETMF ON** command from a z/OS console.

The state of message flood automation can always be queried using the **DISPLAY MSGFLD,STATUS** command.

Chapter 5. Defining auto-reply policy for WTORs

With the auto-reply policy for WTORs, you can get an automatic response from the system to WTOR messages, when there is no automation, the operator is unaware of the outstanding request, or spends a long time determining what response should be given.

Specifically, the auto-reply policy provides the following enhancements on z/OS:

- If an operator or customer-supplied automation has not provided any reply to a WTOR in a specified amount of time, and the auto-reply policy contains this WTOR, the system will use the reply from the policy to reply to the message.
- The default auto-reply policy is activated during IPL, unless you explicitly request that the policy not be activated. If you don't activate the default policy, WTORs issued during NIP cannot be automated.
- You can add to or alter the default auto-reply policy, or provide your own auto-reply policy.
- You can use an operator command to activate or deactivate the auto-reply policy on a system.
- You can use an operator command to display the auto-reply policy and the current outstanding WTORs that are being monitored by auto-reply processing.
- You can use an operator command to deactivate auto-reply processing or to stop monitoring a current outstanding WTOR.
- You can specify a system parameter AUTOR=xx in the IEASYSxx parmlib member or in response to message IEA101A SPECIFY SYSTEM PARAMETERS, to allow your installation to provide a set of parmlib members that contain the auto-reply policy, or to request that auto-reply processing not be activated.

Migration

During IPL, if the parmlib member AUTOR00 exists, auto-reply processing is activated. If the WTORs listed in AUTOR00 are automated by your existing automation product, examine the WTOR replies in the AUTOR00 parmlib member. If the replies or delay duration are not desirable, you can create a new AUTORxx parmlib member and make corresponding changes. Also compare the replies to what your automation product would reply to these WTORs. Make sure that the AUTOR00 replies are in accordance with the replies from your automation product. It's not recommend to make updates to AUTOR00, because updates to AUTOR00 might be made by the service stream or in new z/OS releases.

Note:

1. If you have created an AUTORxx parmlib member, update the IEASYSyy parmlib member that you use for IPL. Add the following statement to the IEASYSyy member:

   ```
   AUTOR=(xx,00)
   ```

   Here xx corresponds to the AUTORxx parmlib member that you created. The IEASYSyy members specifying AUTOR cannot be shared with prior z/OS releases. If you only need the default AUTOR00 settings, you can omit specifying AUTOR= in IEASYSyy, and other z/OS levels can continue to use IEASYSyy. Even if AUTOR= is not specified in IEASYSyy, AUTOR00 is used if it exists.
2. If you don't want to activate auto-reply processing, specify AUTOR=OFF in the parmlib member IEASYSxx or in response to message IEA101A SPECIFY SYSTEM PARAMETERS. It is not recommended that you remove AUTOR00 from parmlib, because service or new releases might reinstall AUTOR00. If there is no AUTOR00 member in parmlib, auto-reply is not activated and the following messages are produced:

   ```
   CNZ2600I AUTO-REPLY POLICY ATTEMPTING TO USE AUTOR=00.
   IEA301I AUTOR00 NOT FOUND IN PARMLIB
   CNZ2601I AUTO-REPLY POLICY NOT ACTIVATED.
   NO ENTRIES SPECIFIED
   ```

Operator commands

The following operator commands are available to control the auto-reply policy for WTORs. For more details, see *z/OS MVS System Commands*.

Table 16. Operator commands to control auto-reply policy for WTORs

Function	Command
Activate auto-reply processing on a system by specifying the AUTORxx parmlib member that the system is to use.	SET AUTOR=(*xx*[,*xx*]...)
Display the auto-reply policy active on the system.	D AUTOR,POLICY
Display the current outstanding WTORs that are being monitored by auto-reply processing.	D AUTOR,WTORS
Deactivate the auto-reply processing and stop monitoring all WTORs issued on the system.	SETAUTOR OFF
Request that the auto-reply processing stop monitoring an outstanding WTOR.	SETAUTOR IGNORE

PARMLIB specifications

To activate auto-reply processing on a system, you need to specify the AUTORxx member of PARMLIB.

Statements provided for the AUTORxx parmlib member are NOTIFYMSGS and MSGID(). The parameters include:

- NOTIFYMSGS(HC)
- NOTIFYMSGS(CONSOLE)
- MSGID() NOAUTORREPLY
- MSGID() DELAY() REPLY()

For details about the statements and parameters, and the wildcard rules applied when you specify MSGID(), see *z/OS MVS Initialization and Tuning Reference*.

IBM supplies a suggested auto-reply policy as AUTOR00. You can modify the member (which is not recommended), or define another AUTORxx member to customize the auto-reply policy. If you don't want a WTOR in AUTOR00 to be monitored, your AUTORxx member can override the policy by specifying the NOAUTOREPLY option. If you want to change the reply or delay value, code a new MSGID() statement for the WTOR. When you specify parmlib members, make sure that AUTORxx comes before AUTOR00.

To enable your installation to specify its own auto-reply policy during IPL, or to request that auto-reply processing not be activated, you need to specify an AUTOR= option in the parmlib member IEASYSxx or in response to message IEA101A SPECIFY SYSTEM PARAMETERS.

Displaying WTORs being monitored by auto-reply processing

Outstanding WTORs that are being monitored by auto-reply processing can be displayed using the D AUTOR,WTORS command. The response to the command is message CNZ2604I, indicating the WTOR text, the reply that will be used, and the time at which the reply will be issued. See *z/OS MVS System Messages, Vol 4 (CBD-DMO)* for details about message CNZ2604I.

You can also issue the DISPLAY R command to display outstanding WTORs that are being monitored by auto-reply processing. The reply ID of the WTORs will be prefixed with a % or & character, depending on whether the message issuer was executing in problem state or in supervisor state.

Additionally, MPF can be used to specify the presentation attributes (color, highlighting, and intensity) for WTORs that are being monitored by auto-reply processing. Attributes can be specified by including a .MSGCOLR AUTOR(c,h,i) statement in your MPFLSTxx member. The default is the same attributes as other WTORs that are specified in the IMEDACTN MPF entry. For more information about the parmlib member MPFLSTxx, see *z/OS MVS Initialization and Tuning Reference*.

Auto-reply notification messages

The purpose of auto-reply notification messages is to record in the hardcopy log the fact that auto-reply plans to take action and whether auto-reply does take action. One of the messages is for notification if the reply would be too long for the WTOR issuer. Where the notification messages are displayed depends on the setting of the NOTIFYMSGS statement in the AUTORxx parmlib member. For more information about auto-reply notification messages, see *z/OS MVS Initialization and Tuning Reference* and *z/OS MVS System Messages, Vol 4 (CBD-DMO)*.

SDSF support for auto-reply policy

SDSF provides support for auto-reply policy with columns and an action character "AutoReply Ignore" (AI) on the system requests (SR) panel. For more information about action characters, and information about providing security for the action character and adding the columns to customized field lists, see *z/OS SDSF Operation and Customization*.

Chapter 6. Planning for operation tasks

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 systems 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.

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 topic 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 topic 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.
- 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*.

Operators can activate the AutoIPL function so that the system can take predefined actions automatically when it is about to enter a disabled wait state. An automatic response can be to re-IPL z/OS, or to take a stand alone dump (SADMP), or to take a SADMP and have SADMP re-IPL z/OS when it has finished. See "Using the automatic IPL function" on page 149 for details.

Operators need to take certain software-side actions after performing the dynamic CPU addition on the hardware side. The newly-added CPUs are either not available or offline and thus need to be brought online. In addition, because the total number of the active processors changes, operators might want to adjust the trace buffer size of the system. See "Exploiting dynamic CPU addition" on page 151 for more details.

Initializing the system

During initialization of an MVS system, the operator uses the system console or hardware management console, which is connected to the 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 139. 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 142.

Using the system console

Use the system console facility of the hardware management 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 141.

Using the AUTOACT console group

If an "automatic activate group" (AUTOACT) is active for the system console, the system will automatically activate and deactivate the system console by issuing the VARY CN(*syscons*),ACTIVATE and VARY CN(*syscons*),DEACTIVATE commands.

- To find out if an AUTOACT group is in place, issue D EMCS,I,CN=*syscons name* or D C,CN=*syscons name*
- To display the consoles in the AUTOACT group, issue D CNGRP

- To add or change the AUTOACT value for the system console, issue VARY CN(*syscons name*),AUTOACT=*groupname*
- To add or change a console group, change the parmlib member CNGRP*xx* , then issue SET CNGRP=*xx*
- To delete the AUTOACT value for the system console, issue VARY CN(*syscons name*),AUTOACT=*NONE*

For more information on AUTOACT, see "Using AUTOACT with the system console" on page 140.

Specifying LOAD information

From the system console facility of the 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 HCD Planning* and *z/OS MVS Initialization and Tuning Reference*.
- 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 HCD Planning*.

Using LOADxx is a good way to automate the initialization procedure for your system and simplify the process for your operators.

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 topic 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.

Using the HMCS console for initialization

The HMCS console can be used to IPL z/OS. Use of the system console or NIP consoles defined in HCD is not required when using HMCS consoles to IPL. If the HMCS console is not available, z/OS looks in the IODF for devices to be used as NIP consoles.

Important: If automation is used to remotely IPL z/OS, the HMCS console (the "Integrated 3270 Console" located on the Hardware Management Console) ***must be disconnected*** from the z/OS partition that is to be IPLed, ***before*** the IPL is initiated. Failure to do so will prevent automation from receiving messages issued during the IPL and prevent automation from taking any appropriate actions.

The NIP console

If no NIP console is defined and ready, MVS will use the system console as the NIP-time console. The first NIP console which is defined and ready will be used during initialization. MVS will not switch from one console to another during NIP.

You can also define the same device that you use for the NIP console on a CONSOLE statement in CONSOLxx as an MCS console. An SMCS console cannot be the NIP console.

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

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 57.

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 141.

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.

Using AUTOACT with the system console

AUTOACT specifies the "automatic activate group" for the system console. It is only valid when DEVNUM(SYSCONS) is specified. If AUTOACT (*groupname*) is specified in CONSOL*xx*, *groupname* is the name of a console group, as defined in CNGRP*xx*.

While the AUTOACT group is defined and not suspended:

- The system console will automatically be placed into problem detemination (PD) mode when all of the consoles in AUTOACT are inactive.
- The system console will automatically be removed from PD mode when any console in the AUTOACT group becomes active.

To suspend AUTOACT processing, issue a VARY CN(*),ACTIVATE or DEACTIVATE command from the system console. This manual intervention will override automatic processing until the opposite command is issued.

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 45.)

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 or perform the following functions:

- Synchronous messages not displayed on another MCS console. Synchronous messages can indicate system problems that require the operator to respond through the system console directly attached to the support element.
- Messages that an operator directs to the system console by specifying the system console name.
- Issue any MVS command.

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, SMCS 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, SMCS, 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
- 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 or are unavailable, you can activate problem determination (PD) mode manually or automatically (for diagnosis purposes, PD mode expands message processing for the system console). To manually activate and deactivate PD mode, use the VARY CN,ACTIVATE and VARY CN,DEACTIVATE commands, respectively. To set up the system console to automatically activate and deactivate PD mode, use the VARY CN,AUTOACT= command (see "Using AUTOACT with the system console" on page 140.)

For details on the syntax and usage of VARY CN,ACTIVATE, VARY CN,DEACTIVATE, and VARY CN,AUTOACT=, 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), and MONITOR attribute on the CONSOLE statement for the system console.

During regular operations (when the system console in 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 141.) 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 can issue any MVS command. 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. MVS commands can be issued from the system console regardless of its operating 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" on page 142.

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.
- 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 the console attribute definition ROUTCODE for the system console SYSCON1 defined in CONSOLxx as follows:

```
CONSOLE DEVNUM(SYSCONS) NAME(SYSCON1)
        ROUTCODE(1-5)
```

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 value applies:

- ROUTCODE(NONE)

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 enters VARY CN(*),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 value for ROUTCODE defined in CONSOLxx:

   - ROUTCODE(1-5)

   Along with other messages it can receive, the system console receives messages defined by routing codes 1 through 5.

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 any console to deactivate problem determination mode.

2. **Normal operations**

   The system console again receives a minimum set of messages. The CONSOLxx default for ROUTCODE is in effect:

   - ROUTCODE(NONE)

The MCS console fails again on the system. The operator reissues VARY CN(*),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:

   - ROUTCODE(ALL)

   Along with other messages it can receive, the system console receives messages defined by all routing codes.

   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*.

Setting the TOD clock accuracy monitor service

The TOD clock accuracy monitor service allows the specification of an acceptable time deviation for the TOD clock from the current external time source (ETS). This service is disabled unless a non-zero value is specified for the ACCURACY parameter in the active CLOCK*xx* member of SYS1.PARMLIB. This service is activated only if the CEC is operating in an STP-only Coordinated Timing Network (CTN) using one of the following External Timing Sources (ETS) as the source of time:

- The dial-out service on the HMC.
- A Network Time Protocol (NTP) server.
- A NTP server with a pulse per second output option.

Note: The ACCURACY value specified is with respect to the ETS time, which may deviate from Universal Coordinated Time (UTC) by a small amount. See section 2.3 of *Server Time Protocol Planning Guide Redbook*, SG24-7280, for additional details of the ETS accuracies.

If the ACCURACY value is non-zero and the system requirements are met, message IEA034I 'THE TOD CLOCK ACCURACY MONITOR IS ACTIVE' is issued.

If the ACCURACY value is non-zero, but the system requirements are not met, message IEA036I 'THE TOD CLOCK ACCURACY MONITOR IS NOT ACTIVE' is issued at IPL.

If the TOD clock exceeds +/- the ACCURACY value, message IEA032E 'TOD CLOCK ACCURACY LIMITS MAY HAVE BEEN EXCEEDED' is issued and then re-issued every one hour until the condition is corrected. Possible corrective actions are:

- Allow the system to correct the time difference on its own. Note: This may take up to 7 hours per every second of deviation to correct.
- Follow your installation's clock synchronization process. Note: This may cause outages of the partitions on all the affected CEC(s).
- Shut down the CTN, deconfigure it, and then use the Set or Adjust the Time panel on the Hardware Management Console (HMC) to correct the time on the CEC. Next, reconfigure the CTN, and then re-IPL the partitions.

If the TOD clock exceeds +/- the ACCURACY value, but drifts back or is corrected to be within the specified tolerance range, message IEA033I 'THE TOD CLOCK IS NOW WITHIN SPECIFIED ACCURACY BOUNDS' is issued.

The CLOCKxx ACCMONINTV parameter allows for a user-defined timing interval to check the TOD clock accuracy. The ACCMONINTV value is only valid when ACCURACY is specified and has a nonzero value. If the ACCMONINTV parameter is not specified, then the ACCURACY value is checked with a default timing interval of 60 minutes.

The TOD clock accuracy monitor service will also issue message IEA049I to the hardcopy log. The message will show the time discrepancy for the TOD clock from the current external time source (ETS). It will also show the ACCURACY threshold specified in CLOCKxx.

Message IEA049I is intended for customers who are using an ETS that is a NTP server with a pulse per second (NTP with PPS) output option, which IBM strongly recommends. NTP with PPS refreshes its discrepancy data every 10 minutes, while a NTP server without PPS has a much longer refresh interval. Thus, message IEA049I will show more accurate data with NTP with PPS, while it may show stale data on a NTP server without PPS.

Stale discrepancy data can also occur on a NTP server with PPS if the user-defined time interval specified by the CLOCKxx ACCMONINTV keyword value is less than the 10 minute NTP with PPS discrepancy data refresh rate.

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.

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 148 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 147. 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

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.
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.

Note:

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.

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.

Using the automatic IPL function

As part of planning your installation's operations so that operators or the system can respond promptly and appropriately to disabled wait states, consider activating the AutoIPL function. AutoIPL can re-IPL z/OS, or take a stand alone dump (SADMP), or take a SADMP and have SADMP re-IPL z/OS when it has finished. The desired actions are represented in an AutoIPL policy, which you state in a DIAGxx parmlib member that the system checks at wait state time.

AutoIPL requires hardware support, standard on all systems beginning with System z10 Enterprise Class (z10 EC). For System z9® Enterprise Class (z9 EC), AutoIPL is provided with feature code 9904 and hardware driver 67 or later (both are required). After applying the feature, you must IPL the system to detect AutoIPL on the z9 EC.

AutoIPL is also available for a z/OS guest on z/VM® Release 5.3.0 or later.

AutoIPL works on a single-system basis (that is, a disabled wait state on a given system prompts that system to take the actions stated in the AutoIPL policy of the system).

Use the following steps to activate AutoIPL:

1. If the policy is to include the taking of a SADMP, generate SADMP on some volume, ensuring that the level of SADMP is correct for the level of z/OS, that the SADMP data set is of sufficient size and has the correct properties set.
2. Code an AUTOIPL statement in a DIAGxx parmlib member, using the syntax described in the *z/OS MVS Initialization and Tuning Reference*.
3. Prompt the system to read the DIAGxx member, either by setting up the parmlib concatenation and IPLing, or by issuing a SET DIAG=xx operator command.
4. Issue the DISPLAY DIAG command to verify that the policy established is what you intended.

To deactivate AutoIPL:

1. Code the following statement in the DIAGxx member, and prompt the system to read it:

   ```
   AUTOIPL SADMP(NONE) MVS(NONE)
   ```

2. Next, issue `DISPLAY DIAG` to verify that the system displays `AUTOIPL SADMP(NONE) MVS(NONE)`.

Part of the AutoIPL support includes a hard-coded table of wait state and reason codes, called the wait state action table (WSAT), which is part of the z/OS nucleus. Each entry has a flag to indicate whether the SADMP part of the AutoIPL policy should be honored, and a flag to indicate whether the MVS part of the AutoIPL policy should be honored. (This is necessary because a few z/OS wait states are inappropriate for a SADMP or a re-IPL.)

When the Loadwait component of z/OS is invoked to load a disabled wait state, it checks the requested wait state and reason code against the table.

For non-restartable wait states, Loadwait will fully honor the AutoIPL policy unless a matching WSAT entry is found that has one or both flags off. If a bit is found off, then the corresponding SAD or re-IPL will not be performed.

For restartable wait states, Loadwait will ignore the AutoIPL policy unless a matching WSAT entry is found that has one or both flags on. If a bit is found on, then the corresponding SAD or re-IPL will be performed. As of this writing, the WSAT contains no entries matching any restartable wait state and reason codes, so a restartable wait state request will not result in any AutoIPL action.

The contents of the wait state action table are described in "Wait state action table (WSAT)" on page 150.

Note:

1. AutoIPL is not appropriate in a GDPS® environment.
2. Do not specify a load device that is defined as a secondary device in a PPRC pair.
3. If an AutoIPL action is performed, the following message does not appear at the HMC: `Central processor (CP) x is in a nonrestartable stopped state due to a System Control Program (SCP)initiated reset of the I/O interface for partition` *n*.
4. Verify the required hardware support on the system by attempting to establish an AutoIPL policy (with SADMP setting other than NONE). If message IGV010I appears, some or all the required support is not present.
5. AutoIPL supports only ECKD devices.
6. To give sysplex failure management (SFM) some time to perform fencing isolation on the failed system, AutoIPL might delay for several minutes before actually initiating the SADMP or the re-IPL. During this time, the failed system will appear to be hung.
7. The AutoIPL functions can optionally be requested on a `VARY XCF` command issued to remove a system from a sysplex. They may not be performed as requested if the sysplex partitioning and/or sysplex resource cleanup activities initiated on the removed system cause that system to request a disabled wait state other than the one associated with the SADMP and/or REIPL keywords specified on the VARY XCF command. If a wait state is requested as a result of sysplex partitioning and/or resource cleanup activities on the removed system, the AutoIPL functions are performed based on the wait state code that was actually requested to be loaded, rather than based on the AutoIPL functions that were requested on the VARY XCF command.
8. Specialty processors such as the System z® Integrated Information Processor (zIIP) and System z Application Assist Processor (zAAP) do not support the Load function that AutoIPL initiates. If a wait state that should result in AutoIPL actions is requested on a specialty processor, the system attempts to switch to a general-purpose processor. If a switch cannot be performed, the AutoIPL action is not performed, and the system loads the requested wait state.

Wait state action table (WSAT)

Entries are of the form frrrrwww, where

f
represents flags

rrrr
represents the reason code

www
represents the wait state code

The '0010'b flag indicates that SADMP is to be IPLed.

The '0001'b flag indicates that z/OS is to be IPLed.

Both flags on ('0011'b) indicates that SADMP is to be IPLed, followed by z/OS.

The '1000'b flag indicates that any reason code (for this wait state code) should be considered a match.

The entries coded into the WSAT as of this writing are as follows:

```
X'000040A2'
X'1017C0A2'
X'201800A2'
X'301840A2'
X'200010B5'
X'200020B5'
X'A0000007'
X'A0000009'
X'A0000037'
X'A0000039'
X'A0000056'
```

Exploiting dynamic CPU addition

This section describes the software-side actions after you perform the dynamic CPU addition on the hardware side. For more information about how to update the LPAR definition, see the "Dynamic Capacity Upgrade on Demand" topic of *System z10 Enterprise Class Processor Resource/Systems Manager Planning Guide*.

After you add CPUs to an LPAR, message ISN011I will be issued for each CPU that is dynamically added.

HiperDispatch=NO environments support only the first 64 CPUs being brought online. When HiperDispatch=NO, you can dynamically add a CPU after the 64th CPU has been added, but z/OS does not allow the CPU to be brought online while the system is running HiperDispatch=NO. If a CPU is dynamically added after the 64th CPU is brought online when HiperDispatch=NO, the system issues ISN012E to indicate HiperDispatch=YES is required to bring the CPU online. For information about HiperDispatch, see "HiperDispatch Mode" in *z/OS MVS Planning: Workload Management* and "IEAOPTxx (OPT) Parameters" in *z/OS MVS Initialization and Tuning Reference*.

Then you need to check if the newly-added processors are offline or not available. Issue the DISPLAY M=CPU command to view the status of the newly-added processors, and the status shown will be either OFFLINE or NOT AVAILABLE. The actions to take are as follows:

1. Change the NOT AVAILABLE status to OFFLINE: either deconfigure one or more CPUs offline from one or more other logical partitions as needed, or perform a physical machine upgrade to add more physical processors to the machine configuration.
2. Bring offline processors online: use the CONFIG CPU command.

For more information about the CONFIG command, see *z/OS MVS System Commands*.

With the CPU addition, you might want to adjust the trace buffer size of your system. The way that you defined the buffer size for the previously existing processors will impact the buffer size set aside for the newly-added processors.

- If you used the TRACE ST,nn command to define the size of the per-processor trace buffer, the newly-added processors will get the same size you assigned to the previously existing processors. If the total size of all the trace buffers exceeds a system-defined maximum, the size per processor will be reduced.
- If you used the TRACE ST,BUFSIZ=nn command to define the overall size of the trace buffer, the total BUFSIZ value will still be honored and the per-processor buffer size will be calculated based on the new

number of processors, thus reducing the per-processor size by a value related to the number of newly-added processors compared to the number of previously existing processors.

To resize the total trace buffer size, use the TRACE ST,BUFSIZ= command. For more information about the TRACE ST command, see *z/OS MVS System Commands*.

Exploiting the z/OS IBM z Advanced Workload Analysis Reporter (IBM zAware) for OPERLOG

Steps necessary to set it up and have operlog messages sent to IBM zAware server:

1. Use OPERLOG hardcopy log. For complete information about the HARDCOPY statement of CONSOLxx, see *z/OS MVS Initialization and Tuning Reference*.
2. Specify ZAI(YES) ZAIDATA('OPERLOG') on the SYSPLEX.OPERLOG log stream definition. See Planning for system logger applications in *z/OS MVS Setting Up a Sysplex* for information on system logger requirements and log stream usage.
3. See Preparing for z/OS IBM zAware log stream client usage in *z/OS MVS Setting Up a Sysplex* for details on getting operlog data to the IBM zAware server.

Chapter 7. 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 17 on page 153 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 18 on page 157 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.

Note: The values for these keywords can also be changed by using the SET `CON=xx` command.

“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.) Values for keywords with a sysplex scope persist through single system IPLs; however, when the sysplex is reinitialized, the values set in the CONSOLxx parmlib members or the IBM defaults are in effect. *z/OS MVS System Commands* provides complete reference information and examples for using MVS commands.

Table 17. CONSOLE Statement Summary

CONSOLE statement keyword	OPERPARM equivalent for extended MCS consoles	Command to change keyword value	Scope	Meaning
CONSOLE DEVNUM	N/A	1. Use the SETCON command or the sample program IEARELCN to delete the console. 2. Update the CONSOLxx parmlib member. 3. Use the SET CON=xx command to add the new console.	See **Note 2**	Identifies SMCS, SYSCONS, subsystem, or the 3-digit or 4-digit device number for the MCS console

Table 17. CONSOLE Statement Summary (continued)

CONSOLE statement keyword	OPERPARM equivalent for extended MCS consoles	Command to change keyword value	Scope	Meaning
CONSOLE UNIT	N/A	1. Use the SETCON command or the sample program IEARELCN to delete the console. 2. Update the CONSOLxx parmlib member. 3. Use the SET CON=xx command to add the new console.	System	Defines the unit device for the MCS console
CONSOLE NAME	See **Note 1**	1. Use the SETCON command or the sample program IEARELCN to delete the console. 2. Update the CONSOLxx parmlib member. 3. Use the SET CON=xx command to add the new console.	Sysplex	Defines the console name
CONSOLE SUPSBY	N/A	VARY CN,SUPSBY	See **Note 4**	Defines whether this console supports the standby attribute
CONSOLE TIMEOUT	N/A	VARY CN,TIMEOUT	See **Note 4**	Defines the automatic LOGOFF timeout attribute for this console
CONSOLE AUTH	OPERPARM AUTH	VARY CN,AUTH	See **Note 2**	Defines command groups or authority
CONSOLE USE	N/A	CONTROL V,USE	See **Note 4**	Defines the input/output capability of the console
CONSOLE DEL	N/A	CONTROL S,DEL	See **Note 4**	Specifies automatic message deletion
CONSOLE RNUM	N/A	CONTROL S,RNUM	See **Note 4**	Defines number of messages per screen rolls

Table 17. CONSOLE Statement Summary (continued)

CONSOLE statement keyword	OPERPARM equivalent for extended MCS consoles	Command to change keyword value	Scope	Meaning
CONSOLE RTME	N/A	CONTROL S,RTME	See **Note 4**	Defines interval of time between screen rolls
CONSOLE CON	N/A	CONTROL S,CON	See **Note 4**	Defines conversational or non-conversational message deletion
CONSOLE SEG	N/A	CONTROL S,SEG	See **Note 4**	Defines the number of lines to delete using CONTROL E,SEG
CONSOLE AREA	N/A	CONTROL A	See **Note 4**	Defines status display areas for a console
CONSOLE MFORM	OPERPARM MFORM	CONTROL S,MFORM	See **Note 4**	Defines message formats for the console
CONSOLE MONITOR	OPERPARM MONITOR	MONITOR	See **Note 3**	Displays jobname, data set status, or TSO/E information
CONSOLE PFKTAB	N/A	CONTROL N,PFK	See **Note 4**	Defines the PFK table for the console
CONSOLE ROUTCODE	OPERPARM ROUTCODE	• VARY CN,ROUT • VARY CN,AROUT • VARY CN,DROUT	See **Note 3**	Defines the routing codes for the console
CONSOLE LEVEL	OPERPARM LEVEL	CONTROL V,LEVEL	See **Note 3**	Defines message levels
CONSOLE MSCOPE	OPERPARM MSCOPE	• VARY CN,AMSCOPE • VARY CN,DMSCOPE • VARY CN,MSCOPE	See **Note 3**	Defines systems that direct messages to a console
CONSOLE CMDSYS	OPERPARM CMDSYS	CONTROL V,CMDSYS	See **Note 3**	Defines systems where commands on a console can be directed for processing

Table 17. CONSOLE Statement Summary (continued)

CONSOLE statement keyword	OPERPARM equivalent for extended MCS consoles	Command to change keyword value	Scope	Meaning
CONSOLE SYSTEM	N/A	VARY CN,ONLINE,SYSTEM	Sysplex	In a sysplex, specifies which system the installation expects the console to be initialized on.
CONSOLE LOGON	N/A	VARY CN,LOGON	See **Note 4**	Defines the LOGON attribute of this console.
CONSOLE LU	N/A	VARY CN,LU	Sysplex	Defines the predefined LU for an SMCS console only.
CONSOLE INTIDS	OPERPARM INTIDS	VARY CN,INTIDS	See **Note 3**	Defines the INTIDS attribute for this console.
CONSOLE UNKNIDS	OPERPARM UNKNIDS	VARY CN,UNKNIDS	See **Note 3**	Defines the UNKNIDS attribute for this console.
CONSOLE RBUF	N/A	None	See **Note 4**	Specifies the number of previously entered commands that can be retrieved on this console by pressing the PA1 key.
CONSOLE AUTOACT	N/A	VARY CN,AUTOACT	System	Specifies the automatic activate group for the system console.

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.
2. Has sysplex scope for SMCS, SYSCONS, and subsystem console; system scope for MCS console.
3. Has sysplex scope for SMCS and SYSCONS console; system scope for MCS console.
4. Has sysplex scope for SMCS console; system scope for MCS console.

Table 18. Summary of INIT, HARDCOPY, and DEFAULT Statements

INIT, HARDCOPY, and DEFAULT statement keywords	MVS commands (in addition to SET CON=XX) 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 MONITOR	MONITOR SETCON MN	System	Displays mount message information
INIT PFK	SET PFK	System	Activates the PFKTABxx member for MCS consoles
INIT CMDDELIM	None	System	Defines the command delimiter for entering multiple messages on MCS consoles
INIT CTRACE	TRACE CT	System	Specifies the parmlib member that contains tracing options for the operation services component
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 18. Summary of INIT, HARDCOPY, and DEFAULT Statements (continued)

INIT, HARDCOPY, and DEFAULT statement keywords	MVS commands (in addition to SET CON=XX) to change keyword value	Scope	Meaning
DEFAULT SYNCHDEST	See Note "2" on page 159.	System	Specifies the console group from which the system can select a console to display synchronous messages
DEFAULT LOGON	1. Update the CONSOLxx parmlib member. 2. Use the SET CON=xx command to change the value.	System	Specifies default LOGON attributes for MCS and SMCS consoles
DEFAULT HOLDMODE	1. Update the CONSOLxx parmlib member. 2. Use the SET CON=xx command to change the value.	System	Specifies whether the operator can freeze the display on MCS console screens
DEFAULT ROUTCODE	1. Update the CONSOLxx parmlib member. 2. Use the SET CON=xx command to change the value.	System	Assigns routing codes for messages without a target console
DEFAULT RMAX	K M,RMAX See Note "1" on page 159.	Sysplex	Specifies maximum number of WTOR reply ids
HARDCOPY DEVNUM	VARY OPERLOG|SYSLOG,HARDCPY	System	Specifies the hardcopy log
HARDCOPY ROUTCODE	• VARY OPERLOG|SYSLOG, HARDCPY,AROUT • VARY OPERLOG|SYSLOG, HARDCPY,ROUT • VARY OPERLOG|SYSLOG, HARDCPY,DROUT	System	Defines route codes for the hardcopy log
HARDCOPY CMDLEVEL	• VARY OPERLOG|SYSLOG, HARDCPY,NOCMDS • VARY OPERLOG|SYSLOG, HARDCPY,INCMDS • VARY OPERLOG|SYSLOG, HARDCPY,STCMDS • VARY OPERLOG|SYSLOG, HARDCPY,CMDS	System	Defines command recording options for the hardcopy log. See Note "3" on page 159.
HARDCOPY HCFORMAT	None	System	Defines 4-digit year format for hardcopy records

Table 18. Summary of INIT, HARDCOPY, and DEFAULT Statements (continued)

INIT, HARDCOPY, and DEFAULT statement keywords	MVS commands (in addition to SET CON=XX) to change keyword value	Scope	Meaning
Note: 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) )
```

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.

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, the CONSOLE command defaults to the userID as the console name, but the user can override the default with the NAME(console-name) operand. 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(OPERCMDS)
   ```

2. Create specific MVS.MCSOPER.console-name profiles naming the intended consoles, and granting users to them only as appropriate for their intended level of authority.

   ```
   RDEFINE OPERCMDS MVS.MCSOPER.console-name UACC(NONE)
   ```

3. Grant the TSO/E user or application program access to the OPERCMDS resources:

   ```
   PERMIT MVS.MCSOPER.console-name CLASS(OPERCMDS) ID(console-name) ACCESS(READ)
   ```

   Console_name must have a RACF user profile defined. See "Defining the user profile of an extended MCS console" on page 159.

4. Issue SETROPTS RACLIST command to refresh the OPERCMDS reserve class:

   ```
   SETROPTS RACLIST(OPERCMDS)
   ```

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. Create specific MVS.MCSOPER.console-name profile naming the intended console, and granting the user access only as appropriate for their intended level of authority. The following example assumes that the OPERCMDS class is active and RACLISTed.

   ```
   RDEFINE OPERCMDS MVS.MCSOPER.TAPE1 UACC(NONE)
   ```

   ```
   PERMIT MVS.MCSOPER.TAPE1 CLASS(OPERCMDS) ID(TAPE1) ACCESS(READ)
   ```

   ```
   SETR RACLIST(OPERCMDS) REFRESH
   ```

2. Issue SETROPTS to activate the TSOAUTH resource class:

   ```
   SETROPTS CLASSACT(TSOAUTH)
   ```

3. 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.

4. 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 that is specified by terminal-id.

   If TAPE1 needs authority to the OPERPARM keyword on the CONSOLE command, issue RDEFINE to define the authority to the resource in class TSOAUTH with a universal access (UACC) of NONE:

   ```
   RDEFINE TSOAUTH CONOPER UACC(NONE)
   ```

and issue RACF PERMIT to authorize TAPE1 to use the OPERPARM keyword:

```
PERMIT CONOPER CLASS(TSOAUTH) ID(TAPE1) ACCESS(READ)
```

5. To refresh the TSOAUTH resource class by using SETROPTS RACLIST, issue the following:

```
SETROPTS RACLIST(TSOAUTH)
```

Changing console attributes using RACF

To change the console attributes, use one of the console-related MVS commands. The RACF ALTUSER command method changes the default attributes for the named console, but will not be effective until the next time the console is activated. It does not change the attributes of existing consoles. Console related MVS commands are used to change the attributes of existing consoles.

```
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: 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 56 and "Defining RACF profiles" on page 57.)

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 57.
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 57.

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 57.

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 57.

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 56 and "Defining RACF profiles" on page 57.)

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 57.

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 57.
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 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 57.

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 57.
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 57.
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 routines of a dynamic exit

To control who can call a dynamic exits 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(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 57.
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 57.

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 56 and “Defining RACF profiles” on page 57.)

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 57.
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 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 CSVDYNL.lnklstname.DEFINE UACC(NONE)
   ```

   where `lnklstname` 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 use the REQUEST=DEFINE option of the CSVDYNL macro for LNKLST set l, issue the following:

   ```
   PERMIT CSVDYNL.l.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 57.

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 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.lnklstname.ADD UACC(NONE)
   ```

   where `lnklstname` 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 l, issue the following:

   ```
   PERMIT CSVDYNL.l.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 57.

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.lnklstname.DELETE UACC(NONE)
   ```

   where `lnklstname` 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 delete a data set from LNKLST set l, issue the following:

   ```
   PERMIT CSVDYNL.l.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 57.

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 CSVDYNL.lnklstname.UNDEFINE UACC(NONE)
```

where `lnklstname` 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 remove the definition of LNKLST set l, issue the following:

   ```
   PERMIT CSVDYNL.l.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 57.

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.lnklstname.TEST UACC(NONE)
   ```

   where `lnklstname` 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 l, issue the following:

   ```
   PERMIT CSVDYNL.l.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 57.
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.l.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 57.
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.lnklstname.ACTIVATE UACC(NONE)
   ```

where `lnklstname` 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 l, issue the following:

   ```
   PERMIT CSVDYNL.l.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 57.
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 56 and “Defining RACF profiles” on page 57.)

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 57.

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 CSVDYLPA.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 57.
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 CSVDYLPA.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,

   ```
   CSVDYLPA.DELETE.M*
   ```

   If you have RACF 1.9 or higher installed, you can use the following generic to cover all module names:

   ```
   CSVDYLPA.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 CSVDYLPA.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 57.
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 design a cluster to suit the special needs of your installation. A typical cluster might consist of four consoles set up as follows:

- A console with master authority, a minimum of routing codes 1 and 2, and in full-capability mode to receive the action messages or important informational messages 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.

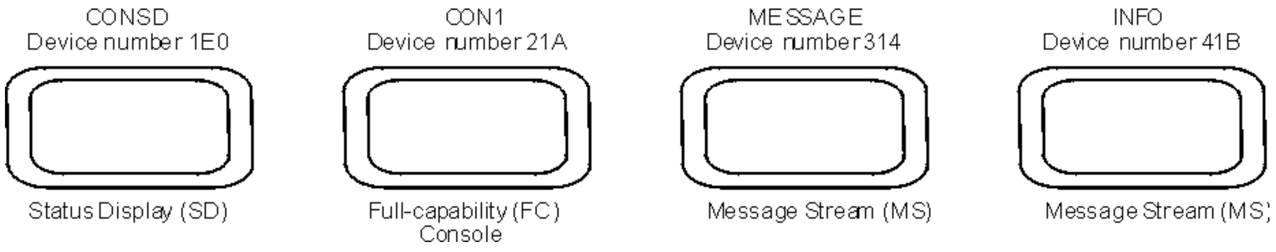
Include other consoles in the cluster if you want to divide the console message traffic even more.

Setting up and using a console cluster

If you decide to set up a 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 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.
- 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:

- CONSD, the status display console, will receive the output from the DISPLAY command.
- CON1, the full-capability console with master authority, will receive the messages that the 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 185 for a summary of the coded CONSOLE statements used in this example. See "Defining PFKs for CON1 " on page 182 for a summary of the PFK table definitions for the console CON1.

Operators can use commands to change these values; however, in this example, only the SYS1.PARMLIB definitions are shown.

Setting up a console cluster requires several steps. This example describes:

- 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 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)

Defining routing codes for the consoles

Use routing codes to set up CON1 so that it receives only messages for which the operator is responsible. Direct other messages to MESSAGE and all routing codes to INFO. In the section "Defining the Operating Modes and the Message Levels for the Consoles", 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 CON1 and the message stream consoles (MESSAGE and INFO), substituting device numbers appropriate to your installation:

Use the INTIDS and UNKNIDS attributes to set up CON1 so that it receives messages issued to console id zero and unknown console ids.

```
CONSOLE DEVNUM(21A) NAME(CON1) ROUTCODE(1,2,9,10) INTIDS(Y) UNKNIDS(Y)
CONSOLE DEVNUM(314) NAME(MESSAGE) ROUTCODE(3-8,12-15,42)
CONSOLE DEVNUM(41B) NAME(INFO) ROUTCODE(1-10,12-128)
```

As a result of these statements, CON1, MESSAGE, and INFO display the range of messages to be processed by this cluster. You do not need to define routing codes for CONSD because you are going to put CONSD in status display mode.

Defining the operating modes and the message levels for the consoles

Code the following statement in CONSOLxx to define the operating mode of CONSD to output-only for system status displays:

```
CONSOLE DEVNUM(1E0) NAME(CONSD) USE(SD)
```

Use statements in CONSOLxx to define the operating modes and the message levels for CON1, MESSAGE, and INFO. To further reduce the messages that appear at CON1, which is already in full-capability mode, eliminate non-action messages from CON1. Code the following statement in CONSOLxx:

```
CONSOLE DEVNUM(21A) NAME(CON1) LEVEL(R,I,CE,E,NB)
```

As a result of this statement CON1 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 CON1 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 the areas on CONSD. In this example, the screen size of CONSD is 43 lines. To define area A to be 28 lines and area B to be 15 lines, code the following statement in CONSOLxx:

```
CONSOLE DEVNUM(1E0) NAME(CONSD) AREA(28,15)
```

CON1 should also have a display area. An area of ten lines should be enough. To establish this area, code the following statement in CONSOLxx:

```
CONSOLE DEVNUM(21A) NAME(CON1) AREA(10)
```

The display areas you have established on the consoles are:

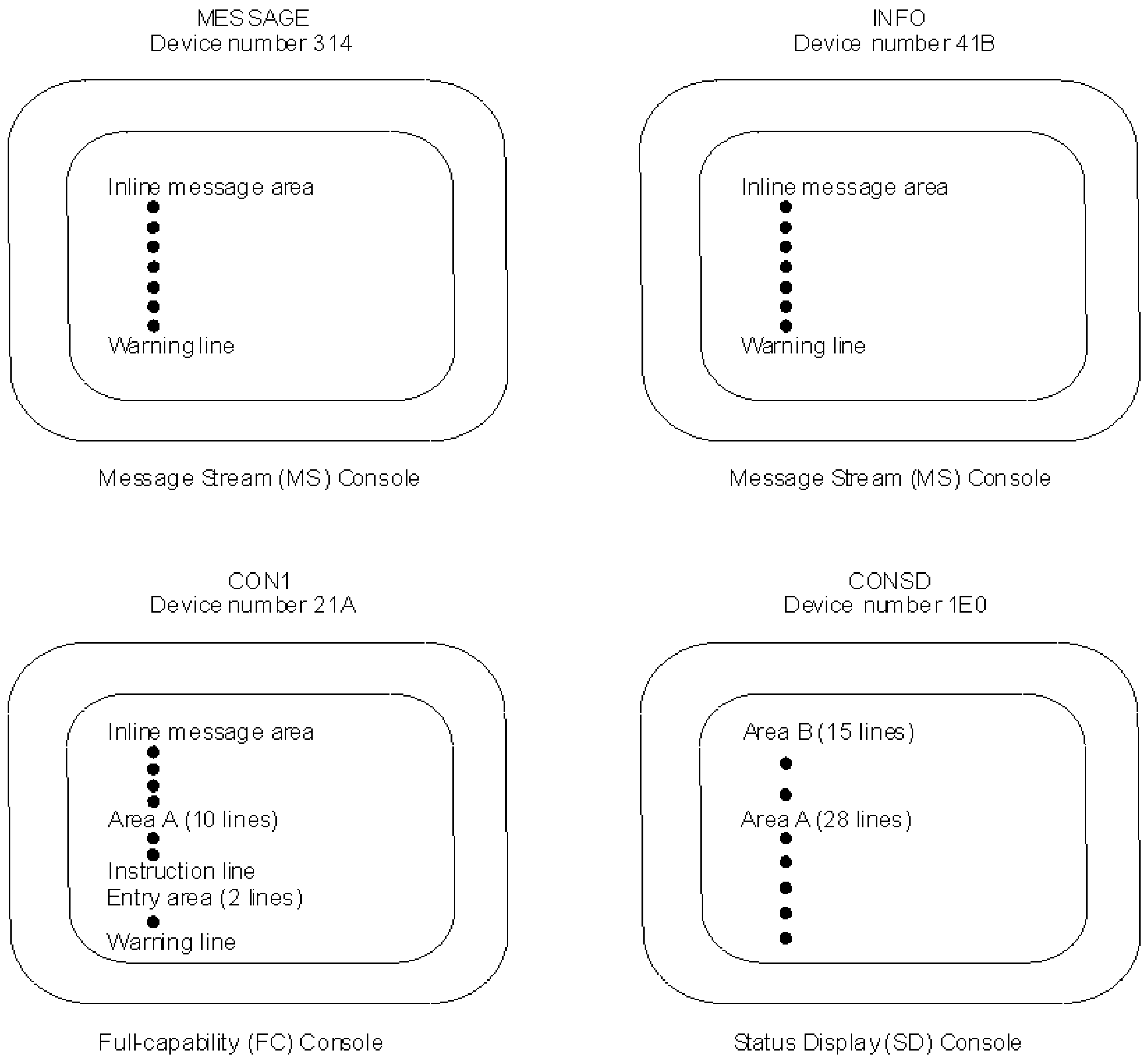


Figure 14. Display Areas on Consoles in the Console Cluster

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 CON1, MESSAGE, and INFO, code the following statements in CONSOLxx:

```
CONSOLE DEVNUM(21A) NAME(CON1) DEL(RD) SEG(39) RNUM(10) RTME(1/2)
CONSOLE DEVNUM(314) NAME(MESSAGE) DEL(W) RTME(1/4)
CONSOLE DEVNUM(41B) NAME(INFO) DEL(R) SEG(39) RNUM(10) RTME(1/2)
```

These statements put CON1 in roll-deletable mode, MESSAGE in wrap mode, and INFO in roll mode. (See "Specifying automatic message deletion for MCS, HMCS or SMCS consoles" on page 67 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 CON1 and INFO. Adjust the RTME specification for MESSAGE in wrap mode. To allow HOLDMODE 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 70.

Once you determine the appropriate values for RNUM and RTME, code the values in the RNUM and RTME parameters in CONSOLxx.

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 CONSD. Issue the following command to make this change:

```
REDIRECT(CON1),DA=CONSD-B,DF=CONSD-B,
 DF=CONSD-B,DI=CONSD-B,DJ=CONSD-B,DN=CONSD-B,
 DQ=CONSD-B,DU=CONSD-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 CONSD, issue the following command:

```
$TA,I=60,'$VS,''D R,L,L=CONSD-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.

Defining PFKs for CON1

You have to redefine some of the PFKs 1 through 8 that the system assigns at IPL and define additional PFKs for CON1 because:

- The eight PFKs are not enough to set up and use the console cluster effectively.
- PFKs 1 through 8 do not put the commands you need to operate the console cluster in the most convenient places.

You need to define PFKs on CON1 for the common operator command functions and the commands to control the console cluster because CON1 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 PFKDEF1. All the definitions in this section follow the first statement in the table:

```
PFKTAB TABLE(PFKDEF1)
```

The commands you define in this table go into effect at IPL, providing you activate the table. "Activating the PFK table" on page 185 describes how you activate PFKDEF1 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 PFKDEF1 to control erasing and displaying of messages on CON1:

```
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 CON1, code the following entry in PFKDEF1:

```
PFK(15) CMD('K E,D,L=CONSD-B')
PFK(16) CMD('K D,F,L=CONSD-B')
```

As a result of these definitions, PFK 15 erases a status display from display area B of CONSD and PFK 16 frames a status display in display area B of CONSD.

To establish the message routing instructions for JES2 messages, add the following entry to PFKDEF1:

```
PFK(3) CMD("$ADD REDIRECT(CON1),DA=CONSD-B,DF=CONSD-B,
            DF=CONSD-B,DI=CONSD-B,DJ=CONSD-B,DN=CONSD-B,
            DQ=CONSD-B,DU=CONSD-B;$TA,I=60,'$VS,
            ''D R,L,L=CONSD-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 CON1 to display area B of CONSD:

  ```
  $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 CONSD.

You should define PFKs to remove action messages quickly from the screen of CON1 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 PFKDEF1:

```
PFK(12) CMD('K S,DEL=R,L=CON1')
```

As a result of this command, pressing PFK 12 causes CON1 to roll all messages.

When PFK 12 makes CON1 roll all its messages, it reduces the number of backed-up messages, in effect, by displaying them all.

Note:

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
- Tailor the 3270 emulator to support larger screen sizes. For example, you can define the number of rows to be 90 and the number of columns to be 100
- Keep CON1 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 CON1.

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 PFKDEF1:

```
PFK(21) CMD('K V,USE=MS,L=CONSD;K V,USE=SD,L=CONSD;K A,15,18,10,
                L=CONSD;$DU,L=CONSD-A;D R,L,L=CONSD-B)
```

As a result of this command, pressing PFK 21:

- Changes CONSD to message stream mode
- Puts CONSD back in status display mode
- Defines new out-of-line display areas A (15 lines), B (18 lines), and C (10 lines) for CONSD
- Displays JES2 unit record device status in out-of-line display area A of CONSD
- Displays outstanding requests in out-of-line display area B of CONSD

Summary of the PFK definitions for the cluster

The PFK table named PFKDEF1 now contains the definitions that have been defined as in the section "Defining PFKs for CON1 " on page 182. If you issue DISPLAY PFK,TABLE=PFKDEF1, 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=PFKDEF1 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   $ADD REDIRECT(CON1),DA=CONSD-B,DF=CONSD-B
          DF=CONSD-B,DI=CONSD-B,DJ=CONSD-B,DN=CONSD-B,
          DQ=CONSD-B,DU=CONSD-B;$TA,I=60,'$VS,'D R,L,L=CONSD-B'
4    NO   K D,F       FRAME DISPLAY FORWARD IN AREA
5    NO   K S,NAME(CON1) DEL=N   HOLD IN-LINE OUTPUT
6    NO   K S,NAME(CON1) DEL=RD  RESUME IN-LINE OUTPUT
7    NO   D A,L       LIST ACTIVE JOBS AND TSO USERS
8    NO   D R,L       LIST OPERATOR REQUESTS
9         NOT DEFINED
10        NOT DEFINED
11        NOT DEFINED
12   NO   K S,NAME(CON1) DEL=R
13   NO   K E,1
14   NO   K E
15   NO   K E,D,L=CONSD-B
16   NO   K D,F,L=CONSD-B
17   NO   K S,DEL=N
18   NO   K S,DEL=RD
19        NOT DEFINED
20        NOT DEFINED
21   NO   K V,USE=MS,L=CONSD;K V,USE=SD,L=CONSD;
          K A,15,18,10,L=CONSD;$DU,L=CONSD-A;
          D R,L,L=CONSD-B
22        NOT DEFINED
23        NOT DEFINED
24        NOT DEFINED
```

Note:

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 PFKDEF1 must reside in a PFKTABxx Parmlib member. In this example, assume that the member is named PFKTAB02. The following statements in CONSOLxx activate PFKDEF1:

```
CONSOLE DEVNUM(21A) NAME(CON1) PFKTAB(PFKDEF1)
INIT PFK(02)
```

The PFK commands you defined in PFKDEF1 go in effect for CON1 at the next IPL.

Summary of contents of CONSOLxx for the cluster

The statements you place in CONSOLxx to initialize the cluster are:

```
CONSOLE DEVNUM(1E0) NAME(CONSD)
               USE(SD)
               AREA(28,15)

CONSOLE DEVNUM(21A) NAME(CON1)  AUTH(MASTER)
               ROUTCODE(1,2,9,10)
               LEVEL(R,I,CE,E,NB)
               AREA(10)
               DEL(RD) SEG(39) CON(N) RNUM(10) RTME(1/2) INTIDS(Y) UNKNIDS(Y)
               PFKTAB(PFKDEF1)

CONSOLE DEVNUM(314) NAME(MESSAGE)
               ROUTCODE(3-8,12-15,42)
               USE(MS) LEVEL(R,I,CE,E,NB)
               DEL(W) RTME(1/4)

CONSOLE DEVNUM(41B) NAME(INFO)
               ROUTCODE(ALL)
               USE(MS) LEVEL(IN)
               DEL(R) SEG(39) RNUM(10) RTME(1/2)

HARDCOPY CMDLEVEL(STCMDS)

DEFAULT HOLDMODE(YES)

INIT PFK(02) CNGRP(01)
```

Note:

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(CON1),DA=CONSD-B,DF=CONSD-B,DF=CONSD-B,DI=CONSD-B,DJ=CONSD-B,DN=CONSD-B,DQ=CONSD-B,DU=CONSD-B
   - $TA,I=60,'$VS,''D R,L''

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.

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. 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). You might want to set up a console group on each system to handle synchronous messages.

Figure 15 on page 186 illustrates one plan that you might use:

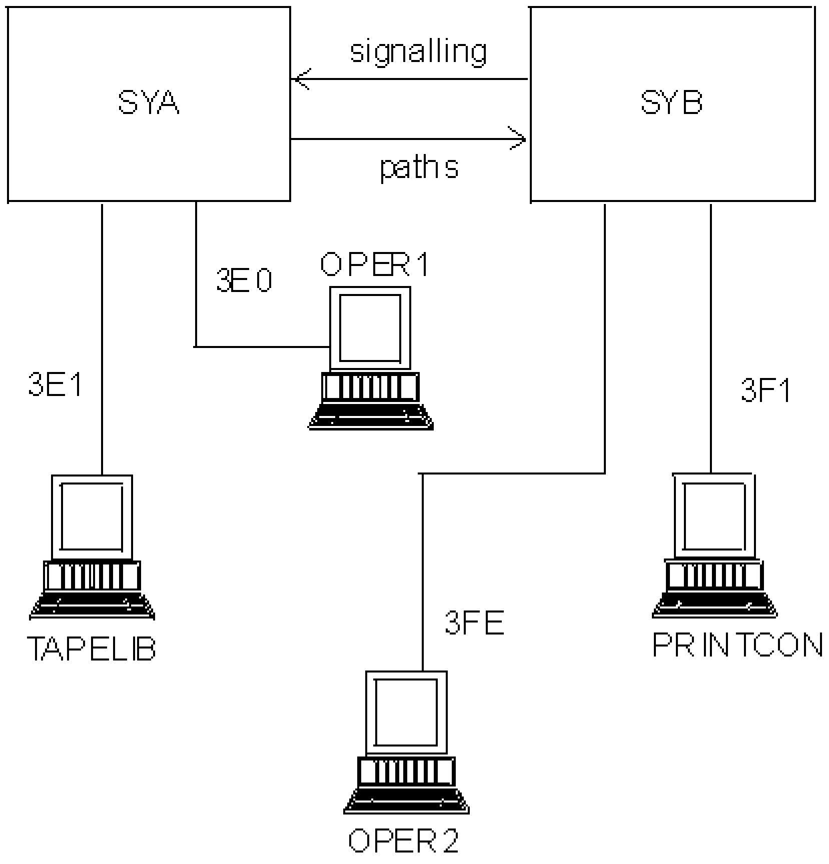


Figure 15. Console Configuration for a Two-System Sysplex

This configuration uses four consoles in the sysplex. Solid lines indicate physical attachments. OPER1 and TAPELIB are both defined to SYA. PRINTCON and OPER2 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. OPER1 and OPER2 are defined similarly to provide redundancy.

Console groups for the sysplex

The following console groups definitions are defined for the sysplex:

```
CNGRP0A  GROUP NAME(SYNCHSYA)
             MEMBERS(OPER1,TAPELIB,*SYSCON*)

        CNGRP0B  GROUP NAME(SYNCHSYB)
             MEMBERS(OPER2,PRINTCON,*SYSCON*)
```

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.

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. You define these console group names on DEFAULT SYNCHDEST for each system. (See "Planning console recovery" on page 47.)

Defining CONSOLxx for each system

SYA and SYB use unique CONSOLxx members to define the console configuration for the sysplex in Figure 15 on page 186.

The CONSOLxx definitions for SYA are as follows:

```
CONSOLE DEVNUM(3E0) NAME(OPER1) UNIT(3270-X)
                    AUTH(MASTER) ROUTCODE(1,2,4,6,8,65-96,9,10)
                    USE(FC) DEL(RD) AREA(18,12) INTIDS(Y) UNKNIDS(Y)
                    RNUM(15) RTME(1) MSCOPE(*ALL)

CONSOLE DEVNUM(3E1) NAME(TAPELIB) UNIT(3270-X) AUTH(SYS,IO,CONS)
                    ROUTCODE(3,5,42) LEVEL(R,I,IN)
                    USE(FC) DEL(R) RNUM(15) RTME(1)
                    AREA(18,12) MFORM(S) MSCOPE(*)

INIT    MPF(01,02,03) CNGRP(0A,0B)

DEFAULT HOLDMODE(YES) SYNCHDEST(SYNCHSYA)
```

You plan to IPL SYA into the sysplex first.

When SYA is IPLed into the sysplex, OPER1 and TAPELIB are active. Both OPER1 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.

OPER1 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. OPER1 also has two display areas defined of 18 lines and 12 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.

OPER1 receives primary operator 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. All messages targeted for console id 0 will go to OPER1 since it has the INTIDS routing attribute. All messages targeted for 1 byte console IDs will go to OPER1 since it has the UNKNIDS routing attribute. 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). OPER1 by default receives messages from all message levels.

SYA specifies MPFLST01, MPFLST02, and MPFLST03 on the INIT statement.

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(OPER2) UNIT(3270-X)
                    AUTH(MASTER) ROUTCODE(1,2,4,6,8,65-96,9,10)
                    USE(FC) DEL(RD) AREA(18,12) INTIDS(Y) UNKNIDS(Y)
                    RNUM(15) RTME(1) MSCOPE(*ALL)
CONSOLE DEVNUM(3E1) UNIT(3270-X) NAME(PRINTCON)
                    ROUTCODE(97-128) USE(FC) DEL(W) RTME(1/4)

INIT MPF(04) MMS(01) CNGRP(0C)

DEFAULT SYNCHDEST(SYNCHSYB)
```

When SYB is IPLed into the sysplex, OPER2 and PRINTCON are active on SYB. Both CNGRP0A and CNGRP0B are already active in the sysplex.

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 add groups, remove groups, or change the members of a group.

OPER2 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 A. AUTOR00 parmlib member

The following is the contents of the parmlib member AUTOR00.

```
/********************************************************************/
/*                                                                  */
/*               Auto-Reply Policy Specifications                   */
/*                                                                  */
/* PROPRIETARY STATEMENT=                                           */
/** Proprietary Statement *******************************************/
/*                                                                  */
/* LICENSED MATERIALS - PROPERTY OF IBM                             */
/* 5650-ZOS COPYRIGHT IBM CORP. 2010, 2013                          */
/*                                                                  */
/* STATUS= HBB7790                                                  */
/*                                                                  */
/** End of Proprietary Statement ************************************/
/*                                                                  */
/* Function: Provide auto-reply policy definitions for common WTORs */
/*                                                                  */
/* Notes   : The message descriptions are just comments. In some    */
/*           cases, the WTORs refer to other messages and those     */
/*           other messages are included here for documentation.    */
/*                                                                  */
/*           For JES2 messages, the wildcard ? is used as the first */
/*           character since the first character is installation    */
/*           dependent.                                             */
/*                                                                  */
/*           The rule delay time is a suggested value. Actual   @P3A*/
/*           values are determined by the WTOR owners.          @P3A*/
/*                                                                  */
/*           For synchronous WTORs, it is suggested that the    @L4A*/
/*           delay value be less than the XCF's failure         @L4A*/
/*           detection interval.                                @L4A*/
/*                                                                  */
/*           For synchronous WTORs, the reply should not        @L4A*/
/*           contain any symbolics since they will not be       @L4A*/
/*           resolved when the reply is issued.                 @L4A*/
/*                                                                  */
/* Rules used to determine if a WTOR should be considered for       */
/* auto-reply processing:                                           */
/*                                                                  */
/*    1   System Detected Problem                                   */
/*                                                                  */
/*        If the system detected an error during execution of an    */
/*        operator initiated action, for example if an operator     */
/*        reply could cause corruption, an auto-reply is needed.    */
/*                                                                  */
/*        Reply:     Cancel the action                              */
/*        Rationale: Resources could be held up; respond quickly.   */
/*                   Worst case is that the user has to reinitiate  */
/*                   the action.                                    */
/*        Delay time: 60 seconds (reject quickly and allow user     */
/*                   to reinitiate)                                 */
/*                                                                  */
/*                                                                  */
/*    2   System Detected Recovery Issue                            */
/*                                                                  */
/*        Tough love on sick but not dead situations.               */
/*                                                                  */
/*        Reply:     Terminate                                      */
/*        Rationale: Want to quickly address before situation       */
/*                   further deteriorates.                          */
/*        Delay time: 60 seconds                                    */
/*                                                                  */
/*                                                                  */
/*    3   System Detected Dynamic Changes                           */
/*                                                                  */
/*        For dynamic changes that were made, choose the latest     */
/*        system active configuration when there is a recognized    */
/*        discrepancy.                                              */
/*                                                                  */
/*        Reply:     Option that reflects latest configuration      */
/*        Rationale: Dynamic changes were intended, but not         */
/*                   hardened.  Return to intended state (dynamic)  */
/*        Delay Time: 30 seconds                                    */
/*                                                                  */
/*                                                                  */
/*    4   Confirmation WTORs                                        */
/*                                                                  */
/*        If a generic confirmation message, reply negative.        */
/*                                                                  */
/*        Reply:     Negative confirmation (e.g., NO, CANCEL ...)   */
/*        Rationale: If the operator has not responded immediately  */
/*                   to the message, assume there is some confusion */
/*                   and do not automatically assume the original   */
/*                   command was correctly entered.  Allow him to   */
/*                   re-enter the command.                          */
/*        Delay time:    60 seconds                                 */
/*                                                                  */
/*                                                                  */
/*    5   Continue with IPL                                         */
/*                                                                  */
/*        If there is a condition that is preventing the system     */
/*        from IPLing, reply to allow the system to continue to     */
/*        IPL.                                                      */
/*                                                                  */
/*        Reply:     Option to allow IPL to continue. (e.g., GO,    */
/*                   CONTINUE ...)                                  */
/*        Rationale: If the condition is preventing the system from */
/*                   ipling and the system needs to be up to        */
/*                   correct the condition, reply to allow the      */
/*                   system to continue the IPL.                    */
/*        Delay time: 60 seconds                                    */
/*                                                                  */
/*                                                                  */
/*    6   Component/Product Recommended Values                      */
/*                                                                  */
/*        Component level expert specified values.                  */
/*                                                                  */
/*        Reply:     Specified by component level expert            */
/*        Rationale: Component Level expert specifications may      */
/*                   over-ride auto-reply rules.                    */
/*        Delay time: Specified by component level expert           */
/*                                                                  */
/*                                                                  */
/* Change Activity:                                                 */
/*   $L0=AUTOR    HBB7770  081225  PDSS: Auto-Reply support         */
/*   $P1=ME15924  HBB7770  080421  PDED: Fix message id type        */
/*   $P2=ME16939  HBB7770  090831  PDSS: Fix typo in prolog         */
/*   $P3=ME17157  HBB7770  090917  PDSS: Update XCF delay values    */
/*   $L1=ME17489  HBB7770  091029  PDSS: Add RMM messages           */
/*   $L2=ME18784  HBB7780  100611  PDED: Add ARC0264A and BBO msgs  */
/*   $L3=ME19489  HBB7780  100708  PDED: Remove BBOT0025D           */
/*   $L4=AUTORDCF HBB7790  101225  PDSS: Auto-Reply for DCCF        */
/*   $L5=FLASHSUP HBB7790  110701  PD00A8: Auto-Reply for CONFIG    */
/*   $L6=FLASHSUP HBB7790  110801  PD00FX: New message ID           */
/*      =ME23014  HBB7790  111021  PD00FX: Cleanup                  */
/*   $01=OA38452  HBB7770  120127  PDHB: Added ending comment       */
/*                                       delimiter for EDG8011D.@01A*/
/*   $L7=ME25144  HBB7790  121113  PD00FX: Reinstate FLASHSUP       */
/*   $02=OA41019  HBB7770  130213  PDOF: IXC501A IXC560A IXC508A    */
/*   $04=OA42321  HBB7770  130628  PDHB: Remove EDG4001D.       @04A*/
/*                                                                  */
/********************************************************************/
/* $HASP070 SPECIFY RECOVERY OPTIONS ('RECOVER' OR 'TERMINATE' OR   */
/*          'SNAP' AND, OPTIONALLY, ',NODUMP')                      */
/*                                                                  */
/* Rule: 2                                                          */
/*                                                                  */
   Msgid(?HASP070)   Delay(30S) Reply(TERMINATE)
/********************************************************************/
/* $HASP294 WAITING FOR RESERVE (VOL volser). REPLY 'CANCEL' TO     */
/*          END WAIT                                                */
/*                                                                  */
/* Rule: 1                                                          */
/*                                                                  */
   Msgid(?HASP294)   Delay(30S) Reply(CANCEL)
/********************************************************************/
/* $HASP360 jobname REQUESTS ACCESS TO JESNEWS (Y OR N)             */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(?HASP360)   Delay(60S) Reply(N)
/********************************************************************/
/* $HASP457 FORWARDED DATA SET NAME FOUND. SHOULD JES2 FORWARD?     */
/*          ('Y' OR 'N')                                            */
/*                                                                  */
/* Rule: 3                                                            */
/*                                                                    */
   Msgid(?HASP457)   Delay(60S) Reply(Y)
/**********************************************************************/
/* $HASP811 REPLY Y TO CONTINUE OR N TO TERMINATE START PROCESSING  */
/*                                                                    */
/* Rule: 5                                                            */
/*                                                                    */
   Msgid(?HASP811)   Delay(30S) Reply(Y)
/**********************************************************************/
/* ANTU2220D "READY FOR FLASHCOPY. REPLY 'I' TO INITIATE, 'C' TO    */
/*           CANCEL"                                                  */
/*                                                                    */
/* Rule: 4                                                            */
/*                                                                    */
   Msgid(ANTU2220D)  Delay(60S) Reply(C)
/**********************************************************************/
/* ANTX8925A device_number TERMINATE STORAGE CONTROL SESSION        */
/*           session_number? REPLY 'Y' OR 'N'                         */
/*                                                                    */
/* Rule: 4                                                            */
/*                                                                    */
   Msgid(ANTX8925A)  Delay(60S) Reply(N)
/**********************************************************************/
/* ANTX8926A device_number RECOVER STORAGE CONTROL SESSION          */
/*           session_number? REPLY 'Y' OR 'N'                         */
/*                                                                    */
/* Rule: 4                                                            */
/*                                                                    */
   Msgid(ANTX8926A)  Delay(60S) Reply(N)
/**********************************************************************/
/* ANTX8942A REMOVE device_number FROM STORAGE CONTROL SESSION      */
/*           session_number? REPLY 'Y' OR 'N'                         */
/*                                                                    */
/* Rule: 4                                                            */
/*                                                                    */
   Msgid(ANTX8942A)  Delay(60S) Reply(N)
/**********************************************************************/
/* ANTX8943A TERMINATE ALL type SDM SESSIONS? REPLY 'Y' OR 'N'      */
/*                                                                    */
/* Rule: 4                                                            */
/*                                                                    */
   Msgid(ANTX8943A)  Delay(60S) Reply(N)
/**********************************************************************/
/* ANTX8944A TERMINATE STORAGE CONTROL SESSION session_id ON STORAGE*/
/*           CONTROL ssid? REPLY 'Y' OR 'N'                           */
/*                                                                    */
/* Rule: 4                                                            */
/*                                                                    */
   Msgid(ANTX8944A)  Delay(60S) Reply(N)
/**********************************************************************/
/* ANTX8973A device_number SUSPEND STORAGE CONTROL SESSION          */
/*           session_number? REPLY 'Y' OR 'N'                         */
/*                                                                    */
/* Rule: 4                                                            */
/*                                                                    */
   Msgid(ANTX8973A)  Delay(60S) Reply(N)
/**********************************************************************/
/* ANTX8978A EXECUTE CREFRESH FORCE? REPLY 'Y' OR 'N'               */
/*                                                                    */
/* Rule: 4                                                            */
/*                                                                    */
   Msgid(ANTX8978A)  Delay(60S) Reply(N)
/**********************************************************************/
/* ANTX8981A SUSPEND ALL XRC SESSIONS? REPLY 'Y' OR 'N'             */
/*                                                                    */
/* Rule: 4                                                            */
/*                                                                    */
   Msgid(ANTX8981A)  Delay(60S) Reply(N)
/**********************************************************************/
/* ARC0264A {MCDS|BCDS} CLUSTERS CHANGED FROM m TO d. IF NOT        */
/*          INTENDED, STARTUP WILL RESULT IN CDS CORRUPTION.         */
/*          INTENDED? (Y OR N)                                        */
/*                                                                    */
/* Rule: 1                                                            */
/*                                                                @L2A*/
   Msgid(ARC0264A)   Delay(15M) Reply(N)
/**********************************************************************/
/* ARC0310A CAN TAPE volser BE MOUNTED ON DEVICE devno? REPLY Y OR N*/
/*                                                                    */
/* Rule: 1                                                            */
/*                                                                    */
```

```
   Msgid(ARC0310A)   Delay(15M) Reply(N)
 /*******************************************************************/
 /* ARC0311A SYSTEM TIMER INOPERABLE - CAN volser BE MOUNTED? REPLY  */
 /*          Y OR N                                                  */
 /*                                                                  */
 /* Rule: 1                                                          */
 /*                                                                  */
   Msgid(ARC0311A)   Delay(15M) Reply(N)
 /*******************************************************************/
 /* ARC0314A CAN THE nvol VOLUME(S) ABOVE BE MOUNTED FOR {RECYCLE |  */
 /*          RECOVER | RESTORE}? REPLY Y OR N                        */
 /*                                                                  */
 /* Rule: 6                                                          */
 /*                                                                  */
   Msgid(ARC0314A)   Delay(15M) Reply(Y)
 /*******************************************************************/
 /* ARC0346A OPEN HAS NOT COMPLETED FOR TAPE volser MOUNTED IN       */
 /*          DEVICE ddd. REPLY Y TO START ADDITIONAL minutes MINUTES */
 /*                                                                  */
 /* Rule: 1                                                          */
 /*                                                                  */
   Msgid(ARC0346A)   Delay(30S) Reply(Y)
 /*******************************************************************/
 /* ARC0380A RECALL WAITING FOR VOLUME volser IN USE BY HOST procid, */
 /*          FUNCTION function. REPLY WAIT, CANCEL, OR MOUNT         */
 /*                                                                  */
 /* Rule: 1                                                          */
 /*                                                                  */
   Msgid(ARC0380A)   Delay(60S) Reply(CANCEL)
 /*******************************************************************/
 /* ARC0387A RECOVER OF DATA SET dsname TIMED OUT WAITING FOR TAPE   */
 /*          VOLUME volser TO BECOME AVAILABLE. SHOULD THE DATA SET  */
 /*          RECOVER REQUEST CONTINUE TO WAIT? REPLY Y OR N          */
 /*                                                                  */
 /* Rule: 2                                                          */
 /*                                                                  */
   Msgid(ARC0387A)   Delay(60S) Reply(N)
 /*******************************************************************/
 /* ARC0505D {PRIMARY SPACE MANAGEMENT | SECONDARY SPACE MANAGEMENT |*/
 /*          INTERVAL MIGRATION | AUTOMATIC BACKUP | AUTOMATIC DUMP} */
 /*          ABOUT TO START, REPLY 'Y' TO START OR 'N' TO SKIP IT    */
 /*                                                                  */
 /* Rule: 6                                                          */
 /*                                                                  */
   Msgid(ARC0505D)   Delay(60S) Reply(Y)
 /*******************************************************************/
 /* ARC0803A WARNING: AUDIT OF CATALOG MAY DEGRADE PERFORMANCE,      */
 /*          REPLY 'Y' TO START AUDIT OR 'N' TO CANCEL AUDIT COMMAND */
 /*                                                                  */
 /* Rule: 6                                                          */
 /*                                                                  */
   Msgid(ARC0803A)   Delay(60S) Reply(Y)
 /*******************************************************************/
 /* ARC0825D RECYCLE TAPE LIST CREATED, DSN=dsname. DO YOU WISH TO   */
 /*          CONTINUE? REPLY 'N' TO STOP RECYLE OR 'Y' WHEN READY    */
 /*          TO MOUNT TAPES.                                         */
 /*                                                                  */
 /* Rule: 6                                                          */
 /*                                                                  */
   Msgid(ARC0825D)   Delay(60S) Reply(Y)
 /*******************************************************************/
 /* ARC0962A All VOLUMES NOT CONTAINED IN THE SAME TAPE LIBRARY OR   */
 /*          STORAGE GROUP. ENTER 'C' TO CANCEL OR MAKE CORRECTION   */
 /*          AND ENTER 'R' TO RETRY                                  */
 /*                                                                  */
 /* Rule: 1                                                          */
 /*                                                                  */
   Msgid(ARC0962A)   Delay(30S) Reply(C)
 /*******************************************************************/
 /* ARC6254A ABACKUP CANNOT ALLOCATE TAPE VOLUME volser BECAUSE      */
 /*          ANOTHER DFSMSHSM FUNCTION HAS IT IN USE. RETRY?         */
 /*          REPLY Y OR N                                            */
 /*                                                                  */
 /* Rule: 1                                                          */
 /*                                                                  */
   Msgid(ARC6254A)   Delay(30S) Reply(N)
 /*******************************************************************/
 /* BBOO0286A BACKWARDS INCOMPATIBLE POST INSTALL ACTION(S) PENDING. */
 /*           NOTE FOR UNINSTALL. REPLY 'CONTINUE' OR 'CANCEL'       */
 /*                                                                  */
 /* Rule: 1                                                          */
 /*                                                              @L2A*/
```

```
   Msgid(BBOO0286A)  Delay(60S) Reply(CANCEL)
/*******************************************************************/
/* BBOO0287A SERVER IS STARTING OUT OF PLACE AT MIXED PTF LEVELS.   */
/*           REPLY 'CONTINUE' OR 'CANCEL'                           */
/*                                                                  */
/* Rule: 3                                                          */
/*                                                              @L2A*/
   Msgid(BBOO0287A)  Delay(60S) Reply(CONTINUE)
/*******************************************************************/
/* BBOT0015D OTS UNABLE TO RESOLVE ALL INCOMPLETE TRANSACTIONS FOR  */
/*           SERVER string. REPLY CONTINUE OR TERMINATE.            */
/*                                                                  */
/* Rule: 5                                                          */
/*                                                              @L2A*/
   Msgid(BBOT0015D)  Delay(30S) Reply(CONTINUE)
/*******************************************************************/
/* BLW004A RESTART INTERRUPT DURING jobname stepname ASID=asid      */
/*         MODE=mode PSW=pppppppp                                   */
/*         SYSTEM NON-DISPATCHABILITY INDICATOR IS {ON|OFF}         */
/*         REPLY ABEND TO ABEND INTERRUPTED PROGRAM,                */
/*         RESUME TO RESUME INTERRUPTED PROGRAM,                    */
/*         REPAIR TO PERFORM REPAIR ACTIONS.                    @L4A*/
/*                                                                  */
/* Rule: 6                                                      @L4A*/
/*       Synchronous Message                                    @L4A*/
/*                                                                  */
   Msgid(BLW004A)    Delay(60S) Reply(REPAIR)
/*******************************************************************/
/* BPXI078D STOP OF NLSname_type REQUESTED, REPLY 'Y' TO PROCEED.   */
/*          ANY OTHER REPLY WILL CANCEL THIS STOP.                  */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(BPXI078D)   Delay(60S) Reply(N)
/*******************************************************************/
/* BPXI083D RESPAWNABLE PROCESS job_name ENDED. REPLY R TO RESTART  */
/*          THE PROCESS. ANYTHING ELSE TO END THE PROCESS.          */
/*                                                                  */
/* Rule: 1                                                          */
/*                                                                  */
   Msgid(BPXI083D)   Delay(30S) Reply(N)
/*******************************************************************/
/* BPXM055D THIS SYSTEM WILL BE DISABLED AS A FILESYSTEM OWNER.     */
/*          REPLY 'Y' TO CONTINUE OR ANY OTHER KEY TO EXIT.         */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(BPXM055D)   Delay(60S) Reply(N)
/*******************************************************************/
/* BPXM061D REPLY "Y" TO PROCEED WITH ACTIVATION. ANY OTHER REPLY   */
/*          ENDS THE COMMAND.                                       */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(BPXM061D)   Delay(60S) Reply(N)
/*******************************************************************/
/* BPXM063D REPLY "Y" TO PROCEED WITH DEACTIVATION. ANY OTHER REPLY */
/*          ENDS THE COMMAND.                                       */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(BPXM063D)   Delay(60S) Reply(N)
/*******************************************************************/
/* BPXM120D F BPXOINIT,FILESYS=funcname SHOULD BE USED WITH CAUTION.*/
/*          REPLY 'Y' TO CONTINUE. ANY OTHER REPLY TERMINATES.      */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(BPXM120D)   Delay(60S) Reply(N)
/*******************************************************************/
/* CBR9810D Reply 'QUIT' to terminate or 'GO' to proceed with       */
/*          recovery.                                               */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(CBR9810D)   Delay(60S) Reply(QUIT)
/*******************************************************************/
/* CNZ9009D CONTINUE WITH MIGRATION? REPLY N TO ABORT OR Y TO       */
/*          CONTINUE                                                */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
```

```
   Msgid(CNZ9009D)   Delay(60S) Reply(N)
/*******************************************************************/
/* CPO4205I CPC name: Enter '1' to keep waiting for pending         */
/*          activation or '2' to accept current capacity setting    */
/*                                                                  */
/* Rule: 3                                                          */
/*                                                                  */
   Msgid(CPO4205I)   Delay(60S) Reply(2)
/*******************************************************************/
/* CPO4206I CPC name: Enter '1' to keep waiting for pending         */
/*          deactivation or '2' to accept current capacity setting  */
/*                                                                  */
/* Rule: 3                                                          */
/*                                                                  */
   Msgid(CPO4206I)   Delay(60S) Reply(2)
/*******************************************************************/
/* EDG0103D DFSMSrmm SUBSYSTEM INTERFACE IS INACTIVE - ENTER        */
/*          "IGNORE", "CANCEL" OR "RETRY"                           */
/*                                                                  */
/* Notes:                                                           */
/* This message is normal for starting RMM after IPL.               */
/*                                                                  */
/* Rule: 5                                                          */
/*                                                                  */
   Msgid(EDG0103D)   Delay(60s) Reply(RETRY)
/*******************************************************************/
/* EDG1107D REQUESTS WAIT TO BE PROCESSED - REPLY "STOP",           */
/* "QUIESCE", "RESTART", OR "M=xx"                                  */
/*                                                                  */
/* Notes:                                                           */
/* You stopped DFRMM while QUIESCED. Reply "RESTART".               */
/*                                                                  */
/* Rule: 6                                                          */
/*                                                                  */
   Msgid(EDG1107D)   Delay(2m)  Reply(RESTART)
/*******************************************************************/
/* EDG1200D I/O ERROR ON CONTROL DATA SET WHEN PROCESSING MESSAGE   */
/* msg_number, REPLY EITHER "RETRY" OR "CANCEL"                     */
/*                                                                  */
/* Notes:                                                           */
/* Low on scratch processing cannot wait during a WTO.              */
/*                                                                  */
/* Rule: 6                                                          */
/*                                                                  */
   Msgid(EDG1200D)   Delay(30s) Reply(CANCEL)
/*******************************************************************/
/* EDG1203D INVENTORY MANAGEMENT PREVENTED PROCESSING OF MESSAGE    */
/* msg_number, REPLY EITHER "RETRY" OR "CANCEL"                     */
/*                                                                  */
/* Notes:                                                           */
/* Low on scratch processing cannot wait during a WTO.              */
/*                                                                  */
/* Rule: 6                                                          */
/*                                                                  */
   Msgid(EDG1203D)   Delay(30s) Reply(CANCEL)
/*******************************************************************/
/* EDG2103D PERMANENT JOURNAL ERROR - REPLY "R" TO RETRY, "I" TO    */
/* IGNORE, "D" TO DISABLE OR "L" TO LOCK                            */
/*                                                                  */
/* Notes:                                                           */
/* Reply "L" and notify Tech Support to run EDGHSKP BACKUP          */
/* to re-enable the journal.                                        */
/*                                                                  */
/* Rule: 6                                                          */
/*                                                                  */
   Msgid(EDG2103D)   Delay(30s) Reply(L)
/*******************************************************************/
/* EDG2106D JOURNAL AND CONTROL DATASET DO NOT MATCH - REPLY "C" TO */
/* CANCEL, "D" TO DISABLE OR "L" TO LOCK                            */
/*                                                                  */
/* Notes:                                                           */
/* Reply "L" and notify Tech Support to run EDGHSKP BACKUP          */
/* to re-enable the journal.                                        */
/*                                                                  */
/* Rule: 6                                                          */
/*                                                                  */
   Msgid(EDG2106D)   Delay(30s) Reply(L)
/*******************************************************************/
/* EDG3213D ANOTHER GETVOLUME CURRENTLY IN PROGRESS - ENTER         */
/* "RETRY", "CANCEL", OR "IGNORE"                                   */
/*                                                                  */
/* Notes:                                                           */
/* Reply "RETRY" until GETVOLUME satisfied.                         */
/*                                                                  */
/* Rule: 6                                                          */
/*                                                                  */
   Msgid(EDG3213D)   Delay(30s) Reply(RETRY)
/********************************************************************/
/* EDG4000D JOURNAL FILE IS LOCKED DURING action FOR volser BY      */
/* jobname, stepname, ddname; ENTER "RETRY" OR "CANCEL"             */
/*                                                                  */
/* Notes:                                                           */
/* Reply "RETRY" so production tape process is good.                */
/* Run EDGHSKP BACKUP to re-enable the journal ASAP.                */
/*                                                                  */
/* Rule: 6                                                          */
/*                                                                  */
   Msgid(EDG4000D)   Delay(2m)  Reply(RETRY)
/********************************************************************/
/* EDG4010D BACKUP IN PROGRESS DURING action FOR volser BY jobname, */
/* stepname, ddname; ENTER "RETRY" OR "CANCEL"                      */
/*                                                                  */
/* Notes:                                                           */
/* Reply "RETRY" so production tape process is good.                */
/* Switch to use DSS Concurrent Copy for EDGHSKP BACKUP(DSS)        */
/*                                                                  */
/* Rule: 6                                                          */
/*                                                                  */
   Msgid(EDG4010D)   Delay(60s) Reply(RETRY)
/********************************************************************/
/* EDG8008D DFSMSrmm I/O ERROR DURING task function REQUEST FOR     */
/* volser - ENTER "RETRY" OR "CANCEL"                               */
/*                                                                  */
/* Notes:                                                           */
/* EDGTVEXT - DFRMM is probably QUIESCEd. Automation should         */
/* start DFRMM again, meanwhile we RETRY.                           */
/*                                                                  */
/* Rule: 6                                                          */
/*                                                                  */
   Msgid(EDG8008D)   Delay(60s) Reply(RETRY)
/********************************************************************/
/* EDG8010D BACKUP IN PROGRESS DURING task function REQUEST FOR     */
/* volser - ENTER "RETRY" OR "CANCEL"                               */
/*                                                                  */
/* Notes:                                                           */
/* EDGTVEXT - Reply "RETRY" so production tape is good.             */
/* Ensure EDGHSKP uses BACKUP(DSS) with concurrent copy.            */
/*                                                                  */
/* Rule: 6                                                          */
/*                                                                  */
   Msgid(EDG8010D)   Delay(60s) Reply(RETRY)
/********************************************************************/
/* EDG8011D DFSMSrmm SUBSYSTEM IS NOT ACTIVE DURING task function   */
/* FOR volser - ENTER "RETRY" OR "CANCEL"                           */
/*                                                                  */
/* Notes:                                                           */
/* EDGTVEXT - Reply "RETRY" so production tape is good.             */
/* Ensure DFRMM is always active.                                   */
/*                                                                  */
/* Rule: 6                                                          */
/*                                                                  */
   Msgid(EDG8011D)   Delay(60s) Reply(RETRY)
/********************************************************************/
/* EDG8013D DFSMSrmm JOURNAL FILE IS LOCKED DURING task function    */
/* REQUEST FOR volser - ENTER "RETRY" OR "CANCEL"                   */
/*                                                                  */
/* Notes:                                                           */
/* EDGTVEXT - Reply "RETRY" so production tape is good.             */
/* Run EDGHSKP BACKUP(DSS) to empty and enable the journal.         */
/*                                                                  */
/* Rule: 6                                                          */
/*                                                                  */
   Msgid(EDG8013D)   Delay(60s) Reply(RETRY)
/********************************************************************/
/* EDG8102D DFSMSrmm SUBSYSTEM NOT ACTIVE DURING function PROCESSING*/
/* FOR volser - ENTER "RETRY", "IGNORE", OR "CANCEL"                */
/*                                                                  */
/* Notes:                                                           */
/* EDGLCSUX - Reply "RETRY" so production tape is good.             */
/* Occurs when SMS tape is in use and volume not in library.        */
/*                                                                  */
/* Rule: 6                                                          */
/*                                                                  */
   Msgid(EDG8102D)   Delay(60s) Reply(RETRY)
```

```
/*******************************************************************/
/* EDG8108D DFSMSrmm I/O ERROR DURING function PROCESSING FOR      */
/* volser - ENTER "RETRY" OR "CANCEL"                              */
/*                                                                 */
/* Notes:                                                          */
/* EDGLCSUX - Reply "RETRY" so production tape is good.            */
/*                                                                 */
/* Rule: 6                                                         */
/*                                                                 */
   Msgid(EDG8108D)   Delay(60s) Reply(RETRY)
/*******************************************************************/
/* EDG8110D BACKUP IN PROGRESS DURING function PROCESSING FOR      */
/* VOLUME volser - ENTER "RETRY" OR "CANCEL"                       */
/*                                                                 */
/* Notes:                                                          */
/* EDGLCSUX - Reply "RETRY" so production tape is good.            */
/* Ensure EDGHSKP uses BACKUP(DSS) with concurrent copy.           */
/*                                                                 */
/* Rule: 6                                                         */
/*                                                                 */
   Msgid(EDG8110D)   Delay(60s) Reply(RETRY)
/*******************************************************************/
/* EDG8113D DFSMSrmm JOURNAL FILE IS LOCKED DURING                 */
/* function PROCESSING FOR VOLUME volser -                         */
/* ENTER "RETRY" OR "CANCEL"                                       */
/*                                                                 */
/* Notes:                                                          */
/* EDGLCSUX - Reply "RETRY" so production tape is good.            */
/* Run EDGHSKP BACKUP(DSS) to empty and enable the journal.        */
/*                                                                 */
/* Rule: 6                                                         */
/*                                                                 */
   Msgid(EDG8113D)   Delay(60s) Reply(RETRY)
/*******************************************************************/
/* EDG8121D ENTER volume req_volser INTO LIBRARY lib_name AND REPLY */
/* "RETRY", OTHERWISE REPLY "CANCEL" OR "CONTINUE"                 */
/*                                                                 */
/* Notes:                                                          */
/* EDGLCSUX - Reply "RETRY" and let operator enter the tape.       */
/*                                                                 */
/* Rule: 6                                                         */
/*                                                                 */
   Msgid(EDG8121D)   Delay(60s) Reply(RETRY)
/*******************************************************************/
/* EDG8122D ENTER volume req_volser INTO LIBRARY lib_name AND REPLY */
/* "RETRY", OTHERWISE REPLY "CANCEL"                               */
/*                                                                 */
/* Notes:                                                          */
/* EDGLCSUX - Reply "RETRY" and let operator enter the tape.       */
/*                                                                 */
/* Rule: 6                                                         */
/*                                                                 */
   Msgid(EDG8122D)   Delay(60s) Reply(RETRY)
/*******************************************************************/
/* EDG8123D ENTER volume req_volser EXPORTED IN STACKED VOLUME     */
/* stack_volser LOCATION loc_name SHELF shelf_number HOME LOCATION  */
/* home - IMPORT VOLUME TO LIBRARY lib_name  AND REPLY             */
/* "RETRY", OTHERWISE REPLY "CANCEL"                               */
/*                                                                 */
/* Notes:                                                          */
/* EDGLCSUX - Reply "RETRY" and let operator enter the tape.       */
/*                                                                 */
/* Rule: 6                                                         */
/*                                                                 */
   Msgid(EDG8123D)   Delay(60s) Reply(RETRY)
/*******************************************************************/
/* ERB306D sid : REPLY WITH OPTIONS OR GO                          */
/*                                                                 */
/* Rule: 5                                                         */
/*                                                                 */
   Msgid(ERB306D)    Delay(60S) Reply(GO)
/*******************************************************************/
/* IAT2855 JES3/VTAM OPEN ACB FAILURE, SPECIFY "RETRY" TO ATTEMPT   */
/*         OPEN AGAIN OR "TERM" TO TERMINATE SNARJP                */
/*                                                                 */
/* Rule: 5                                                         */
/*                                                                 */
   Msgid(IAT2855)    Delay(60S) Reply(RETRY)
/*******************************************************************/
/* IAT3155 SPOOL DATA INTEGRITY CHECKING IS ACTIVE;                */
/*         DO YOU WANT TO TURN IT OFF? (OFF OR CONTINUE)           */
/*                                                                 */
/* Rule: 5                                                          */
/*                                                                  */
   Msgid(IAT3155)    Delay(60S) Reply(CONTINUE)
/********************************************************************/
/* ICH15033A IF ANY SYSTEM IS USING THE DATABASE WITH MASTER DATASET*/
/*           dsname IN DATA SHARING MODE, AND ANY OTHER SYSTEM      */
/*           CONCURRENTLY USES IT IN NON-DATA SHARING MODE, DATABASE*/
/*           CORRUPTION WILL RESULT. PROFILE profile-name IN CLASS  */
/*           class-name INDICATES THAT THIS DATABASE WAS LAST USED  */
/*           IN DATA SHARING MODE, BUT IT IS NOW TO BE USED IN      */
/*           NON-DATA SHARING MODE. IF THE DATABASE IS NOT BEING    */
/*           USED BY ANOTHER SYSTEM IN DATA SHARING MODE, SPECIFY   */
/*           'CONTINUE'.  OTHERWISE SPECIFY 'CANCEL'.               */
/*    or                                                            */
/*                                                                  */
/* ICH15034A IF SYSTEMS FROM MULTIPLE SYSPLEXES USE THE DATABASE    */
/*           WITH MASTER DATASET DSNAME IN DATA SHARING MODE        */
/*           DATABASE CORRUPTION WILL RESULT. YOU ARE RVARYING INTO */
/*           A DATA SHARING MODE ENVIRONMENT. OTHER IRRPLEX_        */
/*           PROFILES EXIST, SUCH AS IRRPLEX_sysplex-name IN CLASS  */
/*           class-name. IF THE DATABASE IS NOT BEING USED BY       */
/*           ANOTHER SYSPLEX, THEN SPECIFY 'CONTINUE'. OTHERWISE    */
/*           SPECIFY 'CANCEL'.                                      */
/*                                                                  */
/*    or                                                            */
/*                                                                  */
/* ICH15042A IF ANY SYSTEM IS USING THE DATABASE WITH MASTER DATA   */
/*           SET DSNAME IN DATA SHARING MODE, AND ANY OTHER SYSTEM  */
/*           CONCURRENTLY USES IT IN NON-DATA SHARING MODE, DATABASE*/
/*           CORRUPTION WILL RESULT. YOU ARE RVARYING INTO A DATA   */
/*           SHARING MODE ENVIRONMENT. PROFILE profile-name IN      */
/*           CLASS class-name INDICATES THAT THIS DATABASE WAS LAST */
/*           USED IN NON-DATA SHARING MODE, BUT IT IS NOW TO BE USED*/
/*           IN DATA SHARING MODE. IF THE DATABASE IS BEING USED BY */
/*           ANOTHER SYSTEM NOT ENABLED FOR RACF SYSPLEX            */
/*           COMMUNICATION SPECIFY 'CANCEL'. OTHERWISE SPECIFY      */
/*           'CONTINUE'.                                            */
/*                                                                  */
/* ICH15041A VALID RESPONSES ARE 'CONTINUE' OR 'CANCEL'             */
/*                                                                  */
/* Rule: 1                                                          */
/*                                                                  */
   Msgid(ICH15041A)  Delay(30S) Reply(CANCEL)
/********************************************************************/
/* IEA015A THIS SYSTEM HAS LOST ALL CONNECTION TO THE SYSPLEX TIMER.*/
/* IF THIS EVENT OCCURRED ON SOME, BUT NOT ALL SYSPLEX MEMBERS, THE */
/* LIKELY CAUSE IS A LINK FAILURE. TO FIX, ENSURE THAT EACH AFFECTED*/
/* SYSTEM HAS AT LEAST ONE CORRECTLY CONNECTED AND FUNCTIONAL LINK. */
/*                                                                  */
/* IF THIS EVENT OCCURRED ON ALL SYSPLEX MEMBERS, THEN THE LIKELY   */
/* CAUSE IS A SYSPLEX TIMER FAILURE. TO FIX, REFER TO THE MESSAGE   */
/* IEA015A DESCRIPTION IN MVS SYSTEM MESSAGES.                      */
/*                                                                  */
/* AFTER FIXING THE PROBLEM, REPLY "RETRY" FROM THE SERVICE CONSOLE */
/* (HMC). IF THE PROBLEM WAS NOT CORRECTED, THIS MESSAGE WILL BE    */
/* REISSUED AND YOU MAY TRY AGAIN. REPLY "ABORT" TO EXIT MESSAGE    */
/* LOOP.  PROBABLE RESULT: 0A2-114 WAITSTATE                    @L4A*/
/*                                                                  */
/* Rule: 6                                                      @L4A*/
/*       Synchronous Message                                    @L4A*/
/*                                                                  */
   Msgid(IEA015A)    Delay(2M)  Reply(RETRY)
/********************************************************************/
/* IEA029D {IEASVC|ALLOC|SCHED} PARMLIB MEMBER HAS AN UNBALANCED    */
/*         COMMENT. REPLY YES TO CONTINUE IPL OR NO TO RESPECIFY    */
/*         {SVC|ALLOC|SCHED} PARM                                   */
/*                                                                  */
/* Rule: 5                                                          */
/*                                                                  */
   Msgid(IEA029D)    Delay(60S) Reply(YES)
/********************************************************************/
/* IEA367A MULTIPLE CONSOLE SUPPORT INOPERATIVE ERROR CODE = xxxx   */
/* REPLY WITH ANY CHARACTER TO CONTINUE WITHOUT MULTIPLE CONSOLE    */
/* SUPPORT                                                      @L4A*/
/*                                                                  */
/* Rule: 2                                                      @L4A*/
/*       Synchronous Message                                    @L4A*/
/*                                                                  */
   Msgid(IEA367A)    Delay(15S) Reply(U)
/********************************************************************/
/* IEA394A THIS SERVER HAS LOST CONNECTION TO ITS SOURCE OF TIME.   */
/* IF THIS EVENT OCCURRED ON SOME, BUT NOT ALL NETWORK SERVERS, THE */
/* LIKELY CAUSE IS A LINK FAILURE.                                  */
/* TO FIX, ENSURE THAT EACH AFFECTED SERVER HAS AT LEAST ONE        */
/* CORRECTLY CONNECTED AND FUNCTIONAL LINK.                         */
/* IF THIS EVENT OCCURRED ON ALL NETWORK SERVERS, THEN THE LIKELY   */
/* CAUSE IA A TIMING NETWORK FAILURE.                               */
/* TO FIX, REFER TO THE MESSAGE IEA394A DESCRIPTION IN MVS SYSTEM   */
/* MESSAGES.                                                        */
/* AFTER FIXING THE PROBLEM, REPLY "RETRY" FROM THE SERVICE CONSOLE */
/* (HMC).                                                           */
/* IF THE PROBLEM WAS NOT CORRECTED, THIS MESSAGE WILL BE REISSUED  */
/* AND YOU MAY TRY AGAIN.                                           */
/* REPLY "ABORT" TO EXIT THE MESSAGE LOOP. PROBABLE RESULT: 0A2-158 */
/* WAITSTATE.                                                   @L4A*/
/*                                                                  */
/* Rule: 6                                                      @L4A*/
/*       Synchronous Message                                    @L4A*/
/*                                                                  */
   Msgid(IEA394A)    Delay(2M)  Reply(RETRY)
/********************************************************************/
/* IEA893A NOT READY. REPLY U WHEN DEVICES ARE READY, OR NO IF NOT  */
/*         MOUNTING. dev,dev,...                                    */
/*                                                                  */
/* Rule: 1                                                          */
/*                                                                  */
   Msgid(IEA893A)    Delay(30S) Reply(NO)
/********************************************************************/
/* IEA500A RESTART INTERRUPT DURING jobname stepname ASID=aaaa      */
/*         MODE=mmmm PSW=psw                                        */
/*         REPLY RESUME TO RESUME INTERRUPTED PROGRAM               */
/*         REPLY ABEND TO ABEND INTERRUPTED PROGRAM             @L4A*/
/*                                                                  */
/* Rule: 2                                                      @L4A*/
/*       Synchronous Message                                    @L4A*/
/*                                                                  */
   Msgid(IEA500A)    Delay(60S) Reply(ABEND)
/********************************************************************/
/* IEA502A RESTART REASON COULD NOT BE OBTAINED. REPLY WITH RESTART */
/* REASON CODE:                                                 @L4A*/
/*                                                                  */
/* Rule: 2                                                      @L4A*/
/*       Synchronous Message                                    @L4A*/
/*                                                                  */
   Msgid(IEA502A)    Delay(60S) Reply(1)
/********************************************************************/
/* IEE599A CONFIG SCM WAITING TO COMPLETE - REPLY C TO CANCEL   @L7A*/
/*                                                                  */
/* Rule: 4                                                      @L7A*/
/*                                                                  */
   Msgid(IEE599A)    Delay(30S) Reply(C)
/********************************************************************/
/* IEE799D VARY CONSOLE DELAYED - REPLY RETRY OR CANCEL             */
/*                                                                  */
/* Rule: 1                                                          */
/*                                                                  */
   Msgid(IEE799D)    Delay(30S) Reply(CANCEL)
/********************************************************************/
/* IEE800D CONFIRM VARY FORCE FOR {nnnnnnnn|dev],(dev,...)(} -      */
/*         REPLY NO OR YES                                          */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(IEE800D)    Delay(60S) Reply(NO)
/********************************************************************/
/* IEF739D CONFIGURATION CHANGE DELAYED DUE TO EXCESSIVE WAIT ON    */
/*         PREVIOUS EDT - REPLY 'WAIT' OR 'TERM'.                   */
/*                                                                  */
/* Rule: 1                                                          */
/*                                                                  */
   Msgid(IEF739D)    Delay(30S) Reply(TERM)
/********************************************************************/
/* ISG017D CONFIRM PURGE REQUEST FOR SYSTEM sysname - REPLY NO OR   */
/*         YES                                                      */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(ISG017D)    Delay(60S) Reply(NO)
/********************************************************************/
/* ISG027D CONFIRM RESTART-RING FOR SYSTEM sysname - REPLY NO OR YES*/
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(ISG027D)    Delay(60S) Reply(NO)
```

```
/********************************************************************/
/* ISG082D CONFIRM REBUILD-RING FOR SYSTEM sysname - REPLY NO OR YES*/
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(ISG082D)    Delay(60S) Reply(NO)
/********************************************************************/
/* ISG101D CONFIRM PURGE FOR ACTIVE SYSTEM sysname - REPLY NO OR YES*/
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(ISG101D)    Delay(60S) Reply(NO)
/********************************************************************/
/* ISG117D CONFIRM REACTIVATE SHOULD BE COMPLETED - REPLY NO OR YES */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(ISG117D)    Delay(60S) Reply(NO)
/********************************************************************/
/* ISG186D GRS CTC dev WAS TARGET OF VARY OFFLINE,FORCE. REPLY KEEP */
/*         TO HAVE GRS RETAIN THE CTC OR FREE TO REMOVE THE CTC     */
/*         FROM GRS.                                                */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(ISG186D)    Delay(60S) Reply(KEEP)
/********************************************************************/
/* ISG220D REPLY C TO CANCEL RNL CHANGE COMMAND, OR S FOR SUMMARY OF*/
/*         RNL CHANGE PROGRESS.                                     */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(ISG220D)    Delay(60S) Reply(C)
/********************************************************************/
/* ISG366D CONFIRM SETGRS REQUEST ON SYSTEM system-name. REPLY      */
/*         {ENQMAXA|ENQMAXU}=value TO CONFIRM OR C TO CANCEL.       */
/*                                                                  */
/*   or                                                             */
/*                                                                  */
/* ISG366D CONFIRM REQUEST TO MIGRATE THE CNS TO system-name. REPLY */
/*         CNS=system-name TO CONFIRM OR C TO CANCEL.               */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(ISG366D)    Delay(60S) Reply(C)
/********************************************************************/
/* ISG880D WARNING GRSRNL=EXCLUDE IS IN USE. REPLYING FORCE WILL    */
/*         RESULT IN THE USE OF SPECIFIED RNSL. REPLY C TO CANCEL.  */
/*                                                                  */
/* Rule: 4                                                          */
/*                                                                  */
   Msgid(ISG880D)    Delay(60S) Reply(C)
/********************************************************************/
/* IXC222D REPLY U TO USE RESOLVED DATA SETS OR R TO RESPECIFY      */
/*         COUPLEXX                                                 */
/*                                                                  */
/* Rule: 3                                                          */
/*                                                                  */
   Msgid(IXC222D)    Delay(60S) Reply(U)
/********************************************************************/
/* IXC289D REPLY U TO USE THE DATA SETS LAST USED FOR typename OR C */
/*         TO USE THE COUPLE DATA SETS SPECIFIED IN COUPLExx        */
/*                                                                  */
/* Rule: 3                                                          */
/*                                                                  */
   Msgid(IXC289D)    Delay(60S) Reply(U)
/********************************************************************/
/* IXC394A ARM ELEMENT IN USE. REPLY Y TO CONFIRM THAT elementname  */
/*         SHOULD BE DEREGISTERED OR N TO CANCEL                    */
/*                                                                  */
/* Rule: 1                                                          */
/*                                                                  */
   Msgid(IXC394A)    Delay(60S) Reply(N)                    /* @P3C*/
/********************************************************************/
/* IXC403D sysname STARTED INITIALIZATION AT hh:mm:ss. REPLY W TO   */
/*         WAIT FOR sysname OR I TO COMPLETE INITIALIZATION.        */
/*                                                                  */
/* Rule: 1                                                          */
/*                                                                  */
   Msgid(IXC403D)    Delay(60S) Reply(W)                    /* @P3C*/
/********************************************************************/
/* IXC501A REPLY Y TO USE COUPLING FACILITY NAMED cfname OR N TO    */
/*         NOT USE COUPLING FACILITY                                 */
/*                                                                   */
/* Rule: 1                                                           */
/*                                                                   */
   Msgid(IXC501A)    Delay(60S) Reply(N)                      /* @02A*/
/*********************************************************************/
/* IXC560A REPLY Y TO CONFIRM THAT COUPLING FACILITY NAMED cfname   */
/*         SHOULD BE USED BY plexname, OR N TO DENY THE USE.        */
/*                                                                   */
/* Rule: 4                                                           */
/*                                                                   */
   Msgid(IXC560A)    Delay(60S) Reply(N)                      /* @02A*/
/*********************************************************************/
/* IXC508A REPLY K TO KEEP, D TO DELETE STRUCTURES FROM SYSPLEX     */
/*         plexname                                                  */
/*                                                                   */
/* Rule: 4                                                           */
/*                                                                   */
   Msgid(IXC508A)    Delay(60S) Reply(K)                      /* @02A*/
/*********************************************************************/
```

Appendix B. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the Contact the z/OS team web page (www.ibm.com/systems/campaignmail/z/zos/contact_z) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area.

If you hear the lines `3* , 3 HOST, 3 STATE`, you know that you can include `HOST, STATE`, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write `HOST STATE`, but you cannot write `HOST HOST`.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line `6.1+ data area`, you must include at least one data area. If you hear the lines `2+, 2 HOST, and 2 STATE`, you know that you must include `HOST, STATE,` or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

Terms and conditions for product documentation

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

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.

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.

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 a console to display synchronous messages or provide auto-activation facilities for the system console.

CONSOLxx
: The Parmlib member used to define message handling, command processing, and MCS, HMCS 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.

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, HMCS 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

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 (EMCS)
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.

full-capability console
An MCS, HMCS or SMCS console that can receive messages and send commands. See ***message-stream console*** and ***status-display console***.

hardcopy log
In systems with multiple console support or a graphic console, a permanent record of system activity. See SYSLOG or OPERLOG.

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.

HMCS
HMC multiple console support.

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.

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

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).

line number
A number associated with a line in a console screen display.

MAS
Multi-access spool.

master authority console
In a system or sysplex, a console defined with AUTH(MASTER)

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 flooding automation
An automation that reacts to the message flooding situation.

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.

NIP
Nucleus initialization program.

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.

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 (OPERLOG)
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.

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
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.

PSW
Program status word.

remote operations
Operation of remote sites from a host system.

roll mode
The MCS, HMCS 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.

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, HMCS, 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.

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.

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.

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

Product Number: 5650-ZOS

SA23-1390-40