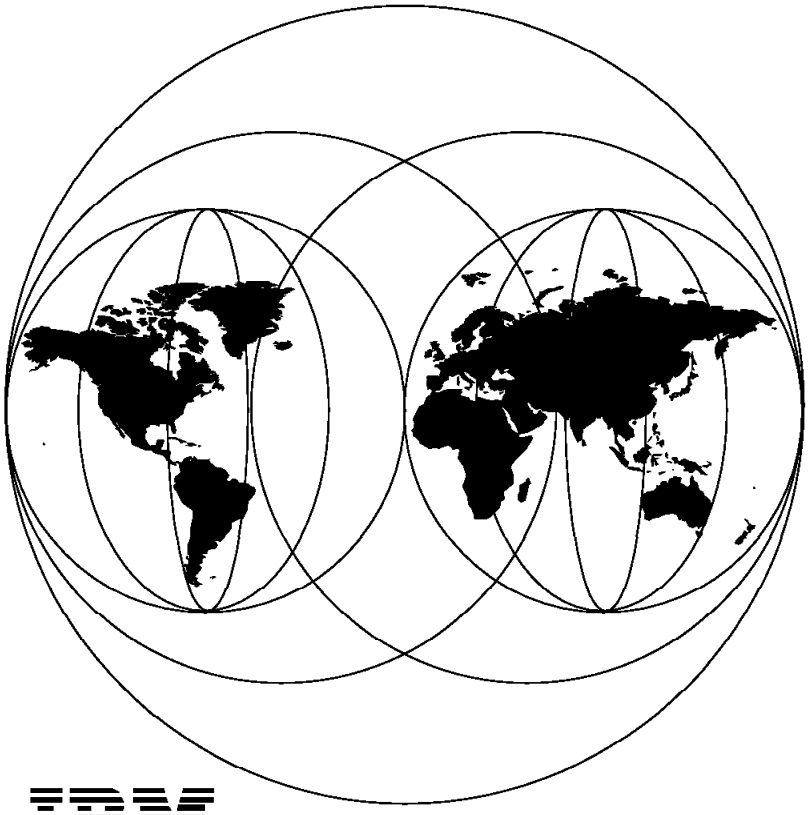
SG24-4626-00

OS/390 MVS Multisystem Consoles Implementing MVS Sysplex Operations

May 1996



International Technical Support Organization
Poughkeepsie Center

International Technical Support Organization

SG24-4626-00

OS/390 MVS Multisystem Consoles Implementing MVS Sysplex Operations

May 1996

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix L, “Special Notices” on page 223.

First Edition (May 1996)

This edition applies to MVS/ESA Version 5 Release 2 of MVS/ESA, Program Numbers 5655-068 and 5655-069 and MVS/ESA JES3 Version 5 Release 2.2, Program Number 5655-069. This edition also applies to Release 1 of OS/390 (5645-001) and to all subsequent releases and modifications until otherwise indicated in new editions.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

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

Contents

Figures

Tables

Preface

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

The introduction of a sysplex into the MVS environment provides a simpler and more flexible way to operate consoles in a multisystem environment. Many changes were introduced into multiple console support (MCS) to support the sysplex environment. These changes began with MVS/ESA Version 4 and have continued with each new MVS release. This document describes all of these changes through the MVS/ESA Version 5.2 release. This publication assumes that you have the potential or desire to run more than one system and you are going to implement a sysplex, where the ultimate goal is to run multiple systems as though they were one system.

This document was written for customers and IBM technical personnel working in support of MVS/ESA sysplex and console environment. Some knowledge of the MVS sysplex environment is assumed.

How This Redbook Is Organized

The document is organized as follows:

- Chapter 1, “Multisystem Consoles in a Sysplex” introduces the concepts of using multisystem consoles in a sysplex.
- Chapter 2, “Operator Communication in MVS” introduces the base elements of operator communication: messages and commands. It describes message attributes, operator command processing, command authorization checking and hardcopy logging.
- Chapter 3, “MCS Sysplex Consoles” contains information on how to set up the various consoles in a sysplex environment. It also includes considerations, examples, and suggestions for alternative sysplex console configurations.
- Chapter 4, “Extended MCS Consoles” contains information on extended MCS consoles (EMCS). An EMCS is a set of programmable interfaces introduced in MVS/ESA SP Version 4 that makes it possible for a program to enter commands and receive messages as though it were an MCS console.
- Chapter 5, “MVS System Logger” highlights the MVS system logger features, describes how to set up the system logger to receive operations log hardcopy message set, and how to active the operations log.
- Chapter 6, “Program to Operator/System Communication” describes console related assembler programming services and also includes examples on how to use these services.
- Chapter 7, “JES2 Sysplex Operations” The JES2 console environment changes with the introduction of multisystem consoles and the other operations features of MVS/ESA. This chapter summarizes the JES2 sysplex operations features and enhancements.

- Chapter 8, “JES3 5.2.1 Sysplex Operations” MVS/ESA JES3 Version 5 Release 2.2 improves systems management and automated operations by enabling the sysplex operations features of MVS/ESA. This chapter summarizes the JES3 sysplex operations features and enhancements.
- Appendix A, “IHAHCLOG Macro” details the IHAHCLOG macro that maps the hardcopy log message prefix.
- Appendix B, “Defining Command Resource Names to RACF” details steps required to set up operator command protection through the security authorization facility.
- Appendix C, “Controlling MCS Console LOGONs through RACF” describes steps involved in console protection through the security authorization facility.
- Appendix D, “Display System Groups Command Exit” contains a sample MCS command exit that displays system groups defined through the IEEGSYS sample program. It also demonstrates how to use the CMDAUTH service to verify an operator′s authority to enter a command.
- Appendix F, “MPF Exit - Add Routing Codes” shows a message processing facility exit that may be used as a model when customizing message routing in a sysplex.
- Appendix E, “CONSOL00 Parmlib Member” shows a sample CONSOLxx parmlib member for the configurations discussed in this redbook.
- Appendix G, “REXX EXEC for Extended MCS Console” shows a sample REXX program for a TSO/E extended MCS console.
- Appendix H, “Extended MCS Console Sample” is an example of an authorized assembler program that activates an extended MCS console to receive the hardcopy message set.
- Appendix I, “Release Migration IDs” contains sample REXX and assembler programs that list the assigned migration IDs for extended MCS consoles and releases the migration IDs on request.
- Appendix J, “Authorizing TSO Users for the CONSOLE Command” lists the steps required to set up RACF profiles that allow TSO/E users to issue the TSO/E CONSOLE and CONSPROF commands.
- Appendix K, “Define, Redefine, and Delete CPF Entry” shows a sample MPF command installation exit that lets you define, redefine, and delete command prefix facility entries.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Poughkeepsie Center.

Paul Rogers IBM ITSO

Juha Vainikainen IBM Finland

Thanks to the following people for their invaluable contributions to this project:

David Share S/390 Software Development

Mary Crisman S/390 Software Development

Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us a note at the following address:

redbook@vnet.ibm.com

Your comments are important to us!

Chapter 1. Multisystem Consoles in a Sysplex

The introduction of a sysplex into the MVS environment provides a simpler and more flexible way to operate consoles in a multisystem environment. Many changes were introduced into multiple console support (MCS) to support the sysplex environment. These changes began with MVS/ESA Version 4 and have continued with each new MVS release. This document describes all of these changes through the MVS/ESA Version 5.2 release. This publication assumes that you have the potential or desire to run more than one system and you are going to implement a sysplex, where the ultimate goal is to run multiple systems as though they were one system.

In a sysplex, MCS consoles can:

- Be attached to any system
- Receive messages from any system in the sysplex
- Route commands to any system in the sysplex

Therefore, new considerations need to be made when defining MCS consoles in this environment, such as:

- There is no requirement that each system have consoles attached.
- The 99 console limit for the sysplex can be extended with the use of extended MCS consoles, see 1.2, “Extended MCS Consoles in a Sysplex.” This adds greater flexibility when attaching consoles.
- A sysplex, which can be up to 32 systems, can be operated from a single console.
- Multiple consoles can have master command authority.

1.1 MCS Consoles in a Sysplex

With MCS consoles in a sysplex, no matter where they are attached, it is possible to control any system in the sysplex. The ability to assign each console a unique name and unique characteristics greatly eases the task of system management.

Currently, MCS consoles are attached to non-SNA control units. MCS allows an installation to define up to 99 MCS and subsystem consoles in a sysplex, each possessing a number of attributes, such as authority to issue different types of commands, routing codes and message level to determine the types of message traffic displayed at the console, message roll time, and so on.

1.2 Extended MCS Consoles in a Sysplex

MCS support provides a programming interface that allows authorized users to issue commands and have the responses returned to them for processing. The commands include any MVS, JES2, JES3, or other subsystem commands that the operator has the authority to issue.

The extended MCS console (EMCS) is a set of programmable interfaces introduced in MVS/ESA SP Version 4 that makes it possible for a program to enter commands and receive messages as though it were an MCS console. See

Chapter 4, “Extended MCS Consoles” on page 97 for a description of how to use these consoles.

This facility can be used by a TSO/E user, a batch job, or a started task address space. With MVS/ESA SP Version 4, TSO/E Version 2 Release 2 and higher, this facility is exploited by the following commands:

- TSO/E CONSOLE
- TSO/E CONSPROF

To implement this new facility in a batch job or in a started task address space environment, three new authorized macros are available; namely:

- MCSOPER
- MCSOPMSG
- MGCRE

Extended MCS consoles are not subject to the 99 console limit. It is possible to assign alternate consoles for extended MCS consoles and to have extended MCS consoles serve as alternates for other (extended and non-extended) MCS consoles.

Note: MCS sysplex support, together with EMCS consoles and hardware consoles, means that a non-SNA control unit and display are not required for every system in the sysplex.

1.3 Console Enhancements for Sysplex

Many enhancements have been added to MCS console support for the sysplex environment. The major functional changes are briefly discussed as follows:

- **Console usability enhancements**

  Three usability enhancements were introduced in MVS/ESA SP Version 4 to improve the readability of console messages and ease operator command entry. These enhancements are available only on MCS consoles:

  – WRAP mode display
  – HOLD mode display
  – Multiple command recall

  In addition to the ROLL mode, MCS provides a WRAP mode option. WRAP mode causes new messages to overlay old messages (wrap around) rather than to appear on the screen at the bottom and “push” older messages up the screen. See 3.2.1.4, “Usability Enhancements” on page 57.

  In HOLD mode, the operator can temporarily suspend the message display on an MCS console to view messages that would otherwise roll off the screen or be overlaid. With this facility an operator can easily “freeze” the console display to view important messages. See 3.2.2.1, “Usability Enhancements” on page 69.

  For information on multiple command recall, see 3.2.1.5, “Multiple Command Recall” on page 58.

- **Integrated system and MVS console**

  It is possible to use the system console of certain IBM ES/9000 processors or the hardware management console of S/390 Parallel Enterprise Servers as an operator console. These consoles are used by MCS as extended MCS consoles. See 3.4, “System Console” on page 87 and 3.6, “Hardware

Management Console Operation" on page 94. This integrated console facility is intended for performing initialization, recovery and diagnostic tasks.

- **Enhanced console switching**

  Prior to MVS/ESA SP Version 4, console functions allowed a console switch from a failing console to a designated alternate attached to the same MVS system. MCS sysplex support permits the installation to switch the function between consoles within a sysplex regardless of where they are attached. Thus, the alternate or backup console can be attached to any system in the sysplex. The alternate console can also be an extended MCS console. See 2.2.1.1, "Console Switching" on page 23 and 3.3.2, "Defining Console Groups" on page 82.

- **Console groups**

  A further flexibility enhancement is the ability to combine MCS and extended MCS consoles into named groups over multiple systems. You can then specify a group name as an alternate for a failing console. When a console fails, MVS transfers its functions to the first available console in the group. Console groups can be changed and displayed using the `SET CNGRP` and `D CNGRP` commands. See 3.3.2, "Defining Console Groups" on page 82. This is an alternative and recommended way of defining alternate consoles for console switching.

- **System symbols**

  MVS/ESA 5.2.0 provides support for symbolic substitution in SYS1.PARMLIB member parameters, MVS and JES commands, and started task JCL. This allows the capability to be able to use a single parmlib member, a single procedure, or a single command to handle multiple instances throughout the sysplex, resulting in simplified system management and decreased system management cost. See 2.5, "System Symbols in Commands" on page 36.

  These system symbols can be used when defining the CONSOLxx parmlib member. See 3.3.4, "Shared CONSOLxx Member with System Symbols" on page 86.

1.4 Operational Considerations in a Sysplex

The major goals of a sysplex from a systems management point of view are to provide:

- Single system image
- Single point of control
- Minimize human intervention

The hardware management console (HMC) of the S/390 9672 Parallel Enterprise Server models has features to support single system image, single point of control, and minimal human intervention for those sysplex environments with these processors. See 1.5.1, "Hardware Management Console" on page 7 for more detail.

In a sysplex, the major tasks of operating an individual MVS image do not change very much. Consoles are used to receive messages and issue commands to accomplish system tasks. With the existence of multiple MVS images and multiple subsystems, the potential exists for the operator to receive

an enormous number of messages, since there is the capability to route all messages from all MVS systems to a single console.

In the sysplex environment you may want to consider the following:

- Reducing the number of consoles by eliminating consoles providing the same function

  You may want to do this if your current environment has many consoles performing the same function and each console operates a different MVS image. Perhaps, this function is cancelling or restarting jobs on an MVS image and now this can be provided by a single console that communicates with all the MVS images.

  With the ability to route messages from all systems to a single console, this consolidation of messages reduces the number of consoles you need.

- Review automation and suppression of messages

  In the sysplex environment, it may be necessary to determine if current automation is functioning in the most efficient manner, for example:

  – The number of images may have increased.

  – Reply IDs are sysplex-wide and replies can be issued from any system. Determine if a single system should do certain tasks for all the other systems.

  – Is there a focal point for automation on a single system and can it handle the movement of a subsystem from one system to another in the sysplex.

  – With the increased number of messages in a sysplex, message suppression should be reviewed.

  – In some environments, an MVS console operator starts, operates, and stops the MVS operating system. This involves controlling the MVS system software and hardware, which includes processors, channel paths, and I/O devices. The operator has to handle messages and problems from many sources including MVS, JES, DFSMS/MVS, CICS, IMS, VTAM and possibly many other subsystems. For this type of environment, automation and the use of the HMC can be used to operate the sysplex.

In a sysplex, MCS consoles can be defined to receive messages from multiple systems and commands entered on any MCS console can be processed on any system. This capability can be used by operations personnel such as tape and print operators, network operators, and the help desk personnel to control their functions from a single console for all systems in the sysplex.

1.4.1 Single System Image

A single system image allows the operator to interact with multiple images of a product as though they were one image. The operator can use the MVS ROUTE command to direct a command to all systems in the sysplex or a subset of the systems in the sysplex instead of repeating the command for each system. When issuing a command to multiple systems, the operator receives all responses grouped together on the same console. See 2.7.3, “MVS ROUTE Command” on page 43 for the syntax description.

The WTOR reply IDs are unique in a sysplex. Each WTOR is assigned a sysplex-wide unique reply ID from a single range of reply IDs. An operator can

reply to a WTOR from any console in the sysplex, even if the console is not directly attached to the system that issued the WTOR.

Some operator commands also have a sysplex scope independent of any routing or automation because the information they retrieve is from resources that are shared throughout the sysplex. See *MVS/ESA SP V5 System Commands* for the MVS commands that have sysplex scope.

Operators can control individual images within the sysplex or they can control groups of images within the sysplex where these images include systems that:

- Have a particular type of workload
- Have the same database regions

1.4.1.1 Managing Logs in a Sysplex

In a sysplex, each system has its own SYSLOG and LOGREC data sets. With multiple systems, you may want to merge the logs from all systems to provide a single system image view of all the log data within the sysplex. Beginning with MVS/ESA SP Version 5, a new component called the system logger provides a set of services that allow you to manage log data across the systems in the sysplex. The log data is in a log stream that resides on a coupling facility. For more detail, see Chapter 5, “MVS System Logger” on page 123.

1.4.2 Single Point of Control

Having a single point of control allows an operator to interact with a product that runs on more than one system. In this way, the operator can accomplish a set of tasks from a single console and thereby reduce the number of consoles the operator has to deal with.

With a single point of control, the operator can receive messages from all systems on a single console. If you are automating or suppressing most of your messages, it is recommended that you set up most of your MCS consoles to receive messages from all systems for the following reasons:

- You might need messages from multiple systems for diagnosing problems.
- Consolidating messages reduces the number of consoles you need.

1.4.3 Minimize Human Intervention

In a sysplex, the need to minimize human intervention can be achieved through extensive automation and message suppression. The following tasks can be automated in the sysplex environment:

- Restarting failed subsystems
- Responding to system failures
- IPLing the sysplex
- Stopping and starting subsystems
- Shutting down the sysplex
- Many normal operator tasks

Many products have functions that help towards achieving minimal human intervention. Try to provide the ability to monitor and control the sysplex system resources from a single point of control as a single system image. These

resources are the sysplex hardware, software, and applications. Organize the automation and message suppression such that:

- Operators should only have to respond to actions that cannot be automated.
- Operators are alerted only to exception conditions.
- Operators are aware of the status of the sysplex.

1.4.3.1 Considerations for Designing Automation

Minimize assumptions about which systems a given application or subsystem runs on. You want to be able to move work around the sysplex without redoing your automation.

You need to understand the sysplex command and message flows. The automation and message suppression routines may see commands and messages from other systems in the sysplex.

Note: See 2.3, “Message Flow in a Sysplex” on page 25 and 2.6, “Command Flow and Routing in a Sysplex” on page 38.

1.5 Operator Consoles in a Sysplex

One way to perform operator tasks in a sysplex is by using a programmable workstation, such as a PS/2 running OS/2, where different products are integrated or accessible. Such a workstation can be referred to as an *integrated operations workstation*. With such a workstation, you need some kind of automated operations to identify the actions and events to be monitored.

The following is a partial list of IBM products that an operator could use to perform automation and tasks in a sysplex to provide a single system image:

- Hardware management console [1]
- NetView
- AOC/MVS [1]
- TSCF [1]
- Command Tree/2 [1]
- CICSPlex SM
- RMF
- ESCON Manager [1]
- TSO/E (see 4.4, “TSO/E Extended MCS Console” on page 112)
- SDSF for JES2 (see 7.3, “SDSF Enhancements for a Sysplex” on page 156)
- SystemView for MVS

Notes

[1] These IBM products are candidates for an *integrated operations workstation*.

Using these type of products in a sysplex environment reduces the need for the operator to be aware that something in the sysplex has failed. These products can monitor most console messages allowing the operator to focus on the exception conditions. Problem management tools simplify or eliminate operator tasks by automatically performing certain actions in response to a problem.

IBM provides an implementation of an integrated operations workstation with Sysplex Operations Manager.

1.5.1 Hardware Management Console

The hardware management console (HMC) is a PS/2, running the hardware management console application (HWMCA), and running under OS/2 using the standard OS/2 graphical user interface. This type of workstation can reduce the complexity of running a multiple image sysplex. The HMC can simplify the sysplex operation by providing:

1.5.1.1 Single System Image

Local operating systems and coupling facility control code running in a coupling facility logical partition can use the console integration facility of the hardware to send operator messages to be displayed by the hardware management console. The hardware management console allows you to monitor and respond to the operating system messages from any CPC Image, coupling facility, or any group of CPC Images configured to it.

All hardware and operating system messages can be routed to the HMC. Therefore, you can operate a multiple image sysplex using a single HMC console or you can have up to four HMCs managing your sysplex. The HMC is the focal point for 9672 and 9674 hardware management.

Note: The HMC should be used as a back up in the event all MCS consoles fail and it should not be considered as a replacement for a master console to operate a sysplex.

1.5.1.2 Single Point of Control

The hardware management console uses a graphical display that provides a single point of control for hardware functions. The HMC allows the operator to perform CPC management functions from one single console controlling all the CPCs in the System/390 microprocessor cluster. For example, the operator can perform an action, such as activate or IPL, against a collection of CPCs in one operation rather than having to activate each individual CPC in the System/390 microprocessor cluster in turn. The HMC also provides the ability to operate on groups instead of requiring multiple operations. The HMC reduces or eliminates repetitive tasks with the use of activation profiles to activate system images. An activation profile consists of various parameters and settings for a system image. You can activate a group of system images at the same time.

1.5.1.3 Minimal Human Intervention

The HMC provides the ability to do exception-based monitoring, and allows you to pre-define system profiles. Using an OS/2 graphical interface improves the sysplex operation tasks.

Activation of system images can be done using activation profiles. An activation profile consists of various parameters and settings for a system image. Activation profiles could be duplicated and modified for multiple system images. You can also activate a group of system images at the same time.

1.5.1.4 HMC Configuration

Figure 1 on page 8 shows a configuration consisting of IBM 9672 models R1, R2, and R3. An integrated support element (SE) is connected to its CPC and to a token ring LAN that may connect the SEs of more than one CPC with the HMC. In the CPC of the R2 and R3 models, the SE is a ThinkPad.

If all the CPCs in a sysplex are connected to the same HMC LAN domain, the control of all the system images is through the use of a common hardware

console (HMC). A single HMC can be used to operate all configured CPCs attached through their SEs. This reduces the systems management overhead usually associated with multiple systems consoles and greatly improves productivity. Up to four HMCs can be configured on the LAN for high availability through redundancy, or for separation of operational functions.

Activation Profiles: To simplify operations, frequently used sequences of hardware initialization tasks can be combined to form activation profiles. A single HMC activation request can then bring the system to an operational state.

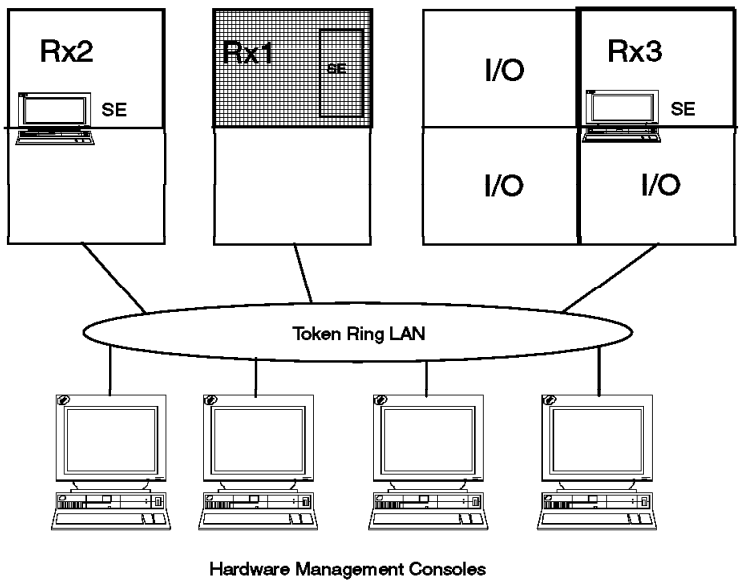


Figure 1. S/390 - 9672 and 9674 - HMC and CPC/SE Connections

There are two kinds of HMCs:

- Master HMC
- Non-master HMC

A master HMC is capable of responding to the hardware configuration definition running in an MVS/ESA system image. HCD must communicate with a master HMC to access profiles stored in the SEs. You can assign a HMC as a master HMC by using the enable hardware management console services task to enable LIC change in the LIC box. From one to four HMCs can be attached to the LAN and the sysplex can be managed from any of the HMCs. All four HMCs could be master HMCs.

Note: If you have more than one HMC in the LAN, make sure their *profile pointers* are identical. The reason is that when HCD chooses a HMC for accessing the profiles stored in the SEs, the choice is random. Any master HMC can respond to HCD. If any of the HMCs has different profile pointers, HCD might not be able to perform certain configuration tasks.

1.5.2 NetView Consoles

Operators can use NetView to monitor, identify, and fix problems in a system, the sysplex, and the network. NetView presents its information in graphical displays. With NetView you can use automated operations facilities to effectively control an unattended sysplex environment by controlling a variety of operational tasks such as operations management, problem management, and change control.

From a NetView or MVS console, you can send operations requests through NetView command lists (in REXX) to do such things as:

- Activate or deactivate the systems in the sysplex
- Request hardware service and Licensed Internal Code changes from the IBM service support system and control the activation
- Monitor the IPL progress of multiple systems without having local NIP consoles or using the MCS console integration windows for each system on the HMC

NetView optionally uses extended MCS consoles so NetView automation can interact with the MVS system as if the NetView operator were an MVS operator with MVS/ESA Version 4 Release 2.2 or later. Using extended MCS consoles enables NetView automation to interact with the MVS system without some of the restrictions imposed in other versions of the MVS system.

NetView as a subsystem receives all messages in the sysplex through the SSI (see 2.3, “Message Flow in a Sysplex” on page 25 and 2.3.3, “SSI Processing” on page 28). Some advantages of using extended MCS consoles with NetView are:

- The 99 console limit on the number of MVS operator consoles is eliminated.
- MVS consoles can be defined dynamically for NetView operators.
- Information appearing on the NetView command facility screen can be made to look more like MVS operator consoles.
- The EMCS consoles do not need to be defined in the CONSOLxx parmlib member.
- System messages can optionally be delivered directly by routing codes.
- The MVS SWITCH command can be used to switch delivery of system messages from one console to another or with the SWITCH parameter of the NetView RELCONID command.

1.5.3 AOC/MVS

AOC/MVS provides a single point for the operator to monitor the status of key resources, or objects, in the sysplex. The resources include:

- Systems or groups of systems
- Applications or groups of applications
- Subsystems
- Volumes
- Coupling facilities and couple data sets

AOC/MVS can be used to initiate automation procedures that perform operator functions to manage these resources in the sysplex. AOC/MVS only needs to know the sysplex name where the application runs to be able to monitor it.

Operators can access the HMC either on the same PS/2 workstation as AOC/MVS uses or remotely on another PS/2 workstation. With AOC/MVS, operators can access other products, such as RMF, SDSF, NetView, and TSCF, and issue commands to them to:

- Monitor hardware status by monitoring alerts from TSCF, which interacts with the processor controller, support element, or system console.
- Report RMF-based service status for any job in the system.
- Interact with RMF so that, when certain application or system groups exceed predefined thresholds, AOC/MVS changes the group′s color to show its changed status.
- Select an AOC/MVS resource from the sysplex support display and, using Command Tree/2, enter one of a selected subset of MVS commands for the resource.

AOC/MVS automation procedures can be used to assist automation by combining several commands under one procedure name. An automation procedure is a sequence of MVS, NetView, VTAM, or application commands, along with AOC/MVS generic routines and common routines packaged as a command list, REXX procedure, or NetView command processor.

AOC/MVS provides automation procedures to initialize, start, recover, shut down, or restart MVS components, subsystems, and applications that are defined using the customization dialogs. In addition to using automation procedures provided with AOC/MVS, you can write your own automation procedures or command processors. To write automation procedures, use the NetView command list language or the REXX language.

In a sysplex, where multiple MVS systems are interconnected and require consolidated operations at one focal point system, you can use focal point services to monitor and control the systems. These services allow an operator to view the status of multiple systems from a single system acting as a focal point, and to send messages, commands, and responses from a focal point system to other systems.

1.5.4 TSCF

The Target system control facility (TSCF) extends the automation capabilities of the NetView program to provide for the operation and monitoring of target systems. TSCF provides automation routines that run on the NetView platform, so you do not have to write those routines yourself. TSCF simulates the actions that a human operator would take at either the hardware or software consoles on various sysplex capable processors. TSCF runs on a focal-point system with NetView, from which it provides centralized control and does not have to reside on each target system.

TSCF interacts with MCS consoles by attaching to the physical console connections on the processors and simulating the activities of human operators at display consoles. TSCF inserts text into fields displayed on a console, simulates an operator pressing Enter and other signalling keys, and extracts messages and other information the processor, operating system, or other software displays on a console.

The S/390 9672 Enterprise Server support element provides an operations command facility (OCF) that accepts and processes a set of operations

management commands. These operations management commands provide facilities comparable to those on the system console of a screen-oriented processor, plus additional facilities such as powering the system on or off and designating that a command is to be executed at a specified future time.

TSCF interacts with the operations command facility by issuing operations management commands to the System/390 microprocessor cluster support element and by receiving command response messages returned by the operations command facility.

This interaction to the operations command facility enables an OCF-based processor to be connected to the TSCF focal-point system by a single physical connection. The operations command facility exchanges operator console data with the MVS operating system using MVS console integration.

With TSCF you can perform console functions for a target system, such as performing an IPL, using commands to simulate keys, and shutting down the system. In addition to the commands provided by TSCF, you can implement the levels of automation and operator control you require. In the sysplex environment, you have the ability to define groups of systems to TSCF, and then be able to IPL all the members of a group with a single TSCF command. TSCF allows the operator to control the startup, initialization, and shutdown of processors, operating systems, and coupling facilities.

1.5.5 Command Tree/2

Command Tree/2 is an OS/2 application that provides a graphical way to issue commands without having to know command syntax. Command Tree/2 uses a tree structure that provides complete command syntax, help, and dialogue panels for prompting the command parameters.

You can tailor the tree structure by adding custom nodes, and can add your own favorite commands to an existing tree. You can also add task headings to organize frequently issued commands in one place. Use Command Tree/2 to issue TSCF, NetView, and VTAM commands. In addition, through AOC/MVS, you can use Command Tree/2 to issue a subset of MVS sysplex commands.

1.5.6 CISCPlex SM

CICSPlex System Manager/ESA (CICSPlex SM) provides a single system image for managing a CICS complex. Operators can use CICSPlex SM to monitor and control CICS resources across the sysplex, without the operator having to know where the resources are. An operator can enter one command to all the systems in the CICS complex.

1.5.7 RMF

RMF can be used to evaluate system performance and identify the reasons for performance problems in the sysplex. In a sysplex, RMF provides sysplex-wide performance monitoring and capacity planning information. RMF combines performance data for individual systems into sysplex-wide overview reports. You can select different levels of detail about systems, resources, and logical entities. In a sysplex, RMF collects software and hardware measurement for the sysplex related resources such as the coupling facility. Access to RMF′s sysplex data is through the RMF sysplex data server.

1.5.8 ESCON Manager

Operators use ESCON Manager to manage the active I/O configuration. ESCON Manager shows views of the I/O configuration from a multisystem perspective, so that status and connectivity information is shown for all systems, not one system at a time. ESCON Manager also coordinates changes made from one location that affect multiple systems, such as varying a device online or offline to every system in the sysplex.

ESCON Manager increases the effectiveness of managing and controlling I/O resources in a sysplex environment, which should improve the ability to manage resources and enhance the effort of doing problem determination.

1.5.9 SystemView for MVS

SystemView for MVS offers a set of systems management functions to assist in effectively managing a MVS system. The SystemView for MVS functions have their server part on the MVS platform, and many have their user interface and possibly other client parts on OS/2 workstations.

SystemView for MVS provides a single point of control from which to graphically manage an MVS system. You install the SystemView for MVS functions you want, and these functions are made available to users through the SystemView for MVS launch window.

SystemView for MVS provides a single OS/2-based graphical interface via an integrated launch window. This window provides easy access to each of the systems management functions. For each user, a profile is maintained by SystemView for MVS that identifies which functions and on which target system that user should have authority to access these functions. SystemView for MVS requests the user to log on to the SystemView for MVS launch window. The user ID and password provided by the user to SystemView for MVS can be authenticated using RACF.

Command Tree is supported and reduces the need to be familiar with the syntax of commands because it helps build commands and route them to a designated system. MVS, JES2, and JES3 commands are now available using Command Tree.

SystemView for MVS offers optional sets of related SystemView for MVS functions that are packaged together, such as:

- System Automation for MVS, which consists of the Automated Operations Control/MVS Version 1 Release 4 (AOC/MVS). The AOC/MVS product includes the features for IMS, CICS, and OPC.
- ESCON Manager Version 1 Release 3.
- Target System Control Facility Version 1 Release 2 (TSCF) functions.
- Network Management for MVS, which consists of the Enterprise System Option of NetView Version 3 for MVS/ESA.
- NetView Multisystem Manager for MVS/ESA Version 2 Release 2 functions.

Chapter 2. Operator Communication in MVS

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

The console operations or how operators interact with MVS to monitor or control the hardware and software include message and command processing that forms the basis of operator interaction with MVS and also the basis for automation.

This chapter describes message and command flow in a sysplex.

2.1 Message Processing in a Sysplex

When the MVS system and any program running under the MVS system requires communication with operators, they issue messages. The MVS write to operator (WTO) and the write to operator with reply (WTOR) macro services cause messages to be routed to the operators and hardcopy log. Normally messages consist of:

- A message text to provide information, describe an error, or request an operator action.
- An identifier, which is usually a three-letter prefix to identify the system component that produced the message and a message serial number to identify the individual message. The identifier may contain other information, for example, a message type code.

Messages can be:

Unsolicited A message routed by a routing code; that is, the message is issued by the system and it is not a response to a command.

Solicited These messages are responses to commands issued by an operator and normally routed back to the console where the command was issued. Some operator commands support the L= operand, which specifies the display area, console or console name, or both, of the console where the command response is to be routed.

Synchronous There are situations in which system operations cannot continue until the system operator takes some external action. An example might be an authorized application detecting a critical problem that warrants stopping the entire system to correct. Using the LOADWAIT macro and the WTO macro with the WSPARM and SYNCH=YES parameters stops the system so the operator can correct a problem, if possible. By using LOADWAIT and WTO, you issue a synchronous message to the operator and place the system into a restartable or nonrestartable wait state. See 6.1.4, “WTO and LOADWAIT Macro Services” on page 145 for use of these macros.

A multiple line WTOR can be used only when SYNCH=YES is also specified. SYNCH=YES indicates the request is to be processed synchronously. This type of WTOR is used in error and recovery environments, when normal message processing cannot be used. The message is sent to the console, and the reply is obtained immediately, before control is returned to the caller.

The essential difference between solicited and unsolicited messages is that solicited messages are (normally) routed by the console ID that issued the command; while unsolicited messages go to consoles that are receiving the routing codes or other general routing attributes used for the message.

2.1.1 Message Type and Descriptor Codes

The IBM message type codes, specified by a letter following the message serial number, are associated with message descriptor codes as follows:

Table 1. Message Type and Descriptor Codes

Type Code	**Descriptor Code**
W (wait)	1
A (action) or D (decision)	2
E (eventual action)	3
I (information)	4 through 10
E (critical eventual action)	11
I (information)	12 and 13

2.1.1.1 Descriptor Codes

The descriptor code identifies the significance of the message. Descriptor codes are specified in the DESC parameter of the WTO or WTOR macro. The descriptor codes have the following meaning:

Action messages: Messages with descriptor codes 1, 2, 3, or 11 are collectively called action messages. Action messages are retained by the action message retention facility (AMRF) for re-display. You can tailor ARMF to retain only a subset of action messages through the message processing facility (MPF) specifications.

Table 2. Action Message Descriptor Codes		
Code	**Message type**	**Message Description**
1	System Failure	The message indicates an error that disrupts system operations. To continue, the operator must reIPL the system or restart a major subsystem.
2	Immediate Action Required	The message indicates that the operator must perform an action immediately. The message issuer could be in a wait state until the action is performed or the system requires the action as soon as possible to improve performance. The task waits for the operator to complete the action.
3	Eventual Action Required	The message indicates that the operator must perform an action eventually. The task does not wait for the operator to complete the action. If the task can determine when the operator has performed the action, the task should issue a DOM macro to delete the message when the action is complete.
11	Critical Eventual Action Required	The message indicates that the operator must perform an action eventually, and the action is important enough for the message to remain on the display screen until the action is completed. The task does not wait for the operator to complete the action. Avoid using this descriptor code for non-critical messages because the display screen could become filled (specially in a sysplex when display consoles are set to *roll messages* except for messages awaiting actions (RD) or *no automatic message deletion* (N) mode). If the task can determine when the operator has performed the action, the task should issue a DOM macro to delete the message when the action is complete.

Most messages are issued with a type code that indicates the message is either informational or an action type message. Action messages may be maintained by the action message retention facility (AMRF).

2.1.1.2 Action Message Retention Facility

The action message retention facility retains all action messages except for those specified on the MPFLSTxx member not to be retained. The messages are retained until either the message is deleted (DOM) or an operator has specifically deleted the message with the CONTROL C command. AMRF has a sysplex-wide scope.

Note: AMRF only applies to action messages issued by the WTO macro. Messages issued by the WTOR macro are not retained by AMRF, but are always available for retrieval using the DISPLAY REQUESTS command. For JES3 installations, see 8.5.3, “JES3 Action Message Retention Facility” on page 170.

The action message retention facility is activated as part of communications task initialization. The initial message retention status is specified on the INIT statement in the CONSOLxx parmlib member (see 3.2.3, “INIT Statement” on page 73). The CONTROL M,AMRF command can be used to dynamically change or display the AMRF status.

If AMRF is active, operators can retrieve outstanding action messages in their entirety by using the DISPLAY R operator command.

If AMRF is deactivated, those messages that have been retained are not deleted until the issuer requests deletion or an operator deletes them with a CONTROL C command. No new messages will be retained.

It is recommended to keep AMRF active when running a sysplex. Consider using MPF definitions to not retain some messages based on their descriptor code or message ID.

Information messages: Messages with descriptor codes 4 through 10, 12 and 13 are information type.

Table 3. Information Descriptor Codes

Code	Message type	Message Description
4	System Status	The message indicates the status of a system task or of a hardware unit.
5	Immediate Command Response	The message is issued as an immediate response to a system command. The response does not depend on another system action or task.
6	Job Status	The message indicates the status of a job or job step.
7	Task-Related	The message is issued by an application or system program. Messages with this descriptor code are deleted when the job step that issued them ends. These messages appear on the system hard-copy log only if they appear on at least one console before the job step ends.
8	Out-of-Line	The message, which is one line of a group of one or more lines, is to be displayed out-of-line. If a message cannot be printed out-of-line because of the device being used, descriptor code 8 is ignored, and the message is printed in-line with the other messages.
9	Operator's Request	The message is written in response to an operator's request for information by a DEVSERV, DISPLAY, TRACK, or MONITOR command.
10	TRACK Command Response	The message is issued in response to a TRACK command.
12	Important Information	The message contains important information that must be displayed at a console, but does not require any action in response.
13	Automation Information	Indicates that this message was previously automated.

The five volumes of system messages describe messages issued by most components of MVS/ESA SP Version 5. The books help you interpret and respond to the informational, error, and diagnostic messages that the system issues during operation. The books and their messages, identified by the prefixes, are: *MVS/ESA System Messages, Volume 1 (ABA-ASA)*, *MVS/ESA System Messages, Volume 2 (ASB-ERB)*, *MVS/ESA System Messages, Volume 3 (GFSA-IEB)*, *MVS/ESA System Messages, Volume 4 (IEC-IFD)*, and *MVS/ESA System Messages, Volume 5 (IGD-IZP)*.

2.1.1.3 Routing Codes

Messages can be routed to a specific console through the CONSID or CONSNAME parameter of the WTO or WTOR macro. Messages can also be routed to a group of consoles by assigning routing codes through the ROUTCDE parameter of the WTO or WTOR macro. The routing code identifies where a message is displayed. A console's definition includes the set of routing codes it should receive. More than one routing code can be assigned to a message. Routing codes have the following meanings:

Table 4 (Page 1 of 2). Routing Code Descriptions

Num	Routing Code Type	Routing Code Description
1	Operator Action	The message indicates a change in the system status. It demands action by the operator at the console with master authority.
2	Operator Information	The message indicates a change in system status. It does not demand action; rather, it alerts the operator at the console with master authority to a condition that might require action. This routing code is used for any message that indicates job status when the status is not requested specifically by an operator inquiry. It is also used to route processor and problem program messages to the system operator.
3	Tape Pool	The message gives information about tape devices, such as the status of a tape unit or cartridge, the disposition of a tape cartridge, or a request to mount a tape.
4	Direct Access Pool	The message gives information about direct access storage devices (DASD), such as the status of a direct access unit or volume, the disposition of a volume, or a request to mount a volume.
5	Tape Library	The message gives tape library information, such as a request by volume serial numbers for tapes for system or problem program use.
6	Disk Library	The message gives disk library information, such as a request by volume serial numbers for volumes for system or problem program use.
7	Unit Record Pool	The message gives information about unit record equipment, such as a request to mount a printer train.
8	Teleprocessing Control	The message gives the status or disposition of teleprocessing equipment, such as a message that describes line errors.
9	System Security	The message gives information about security checking, such as a request for a password.
10	System/Error Maintenance	The message gives problem information for the system programmer, such as a system error, a permanent I/O error, or information about system maintenance.
11	Programmer Information	The message is intended for the problem programmer. This routing code is used when the program issuing the message cannot route the message to the programmer through a system output (SYSOUT) data set. The message appears in the job log.

Table 4 (Page 2 of 2). Routing Code Descriptions		
Num	**Routing Code Type**	**Routing Code Description**
12	Emulation	The message gives information about emulation.
13-20	For customer use only	
21-28	For subsystem use only	
29-40	For IBM use only	
41	For JES use only	The message gives information about JES3 job status
42	For JES use only	The message gives general information about JES2 or JES3
43-64	For JES use only	
65-96	Messages associated with particular processors	65 is the routing code equivalent of JES3 destination class M1.
97-128	Messages associated with particular devices	97 is the routing code equivalent of JES3 destination class S1.

Some level of functional message routing can be implemented by specifying consoles in a functional operational area, for example a tape pool, to receive only the subset of messages that have the function related routing codes.

The *MVS/ESA Routing and Descriptor Codes* book lists the routing and descriptor codes that IBM assigns to the messages that MVS/ESA components, subsystems, and products issue.

The MSGTYP parameter of the WTO(R) macro may also specify how a message is to be routed to consoles on which the MONITOR command is active. If you specify anything other than MSGYTP=N, which is the default, your message will be routed according to your specification on MSGTYP, and the ROUTCDE parameter is ignored.

2.1.2 Console Message Levels

You can also screen what messages are routed to a console by setting the console's message level on the CONSOLE statement in the CONSOLxx parmlib member. The levels you can specify are:

Table 5. Console Message Levels	
Level	**Level Description**
ALL	Indicates that the console is to receive all messages
NB	The console is not to receive any broadcast messages
R	The console is to receive the messages that require an operator reply
I	The console is to receive immediate action messages
CE	The console is to receive critical eventual action messages
E	The console is to receive eventual action messages
IN	The console is to receive informational messages

2.1.3 Hardcopy Processing

Hardcopy processing provides a permanent record of the system messages and, optionally, commands and command responses. To permanently record these messages, it is necessary to define the hardcopy medium. See 3.2.4, “HARDCOPY Statement” on page 78. The hardcopy medium can be a:

- Printer console
- SYSLOG
- OPERLOG

The hardcopy medium is also referred to as the hardcopy log.

The group of messages that is recorded is called the hardcopy message set. The hardcopy message set includes all messages, except for those that are explicitly omitted through the WTO macro or installation exits. You can request that the hardcopy message set not include messages with certain routing codes, but routing codes 1, 2, 3, 4, 7, 8, 10, and 42 cannot be eliminated.

Note: Extended MCS consoles can receive the hardcopy message set but they cannot be the hardcopy medium.

2.1.3.1 Hardcopy Log Messages

Through the MCSFLAG parameter of the WTO(R) macro, you can request a broadcast of the message to all active consoles (BRDCST), the message to be queued for hardcopy only (HRDCPY), or the message not to be queued for hardcopy only (NOCPY).

The hardcopy log provides a permanent record of system activity. You can request recording of system messages and, optionally, commands and command responses, by using either the system log (SYSLOG), the operations log (OPERLOG), or an MCS printer console. The system programmer can also allow extended MCS consoles to receive the hardcopy messages set, and can specify a group of console devices to serve as backup devices for the hardcopy medium.

You can determine a message’s routing information from the hardcopy log (non-JES3 format). The hardcopy log message prefix (mapped by the IHAHCLOG macro) includes the routing information:

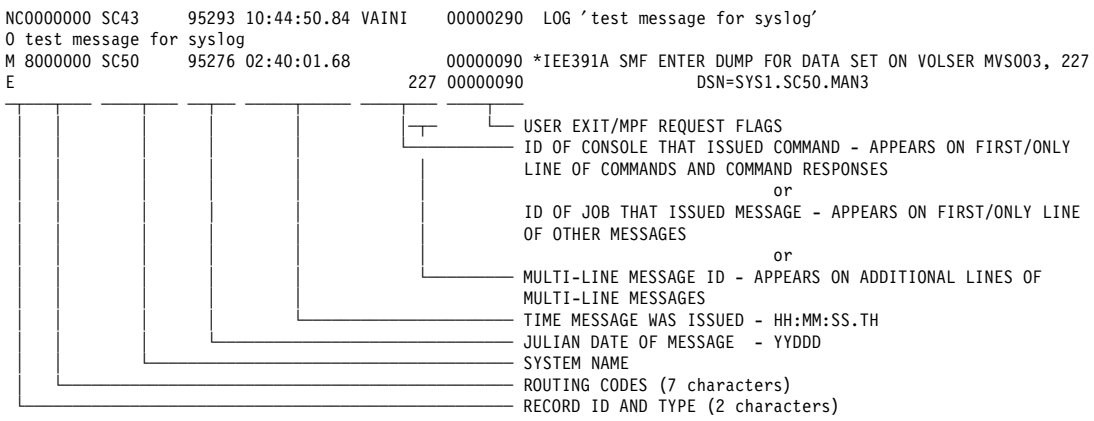


Figure 2. Hardcopy Log Messages

The IHAHCLOG macro can be found in Appendix A, "IHAHCLOG Macro" on page 179.

Notes:

The record ID "O" messages (LOG messages), shown in the second message line in Figure 2, do not have the standard message prefix.

1. The Julian date of the message may change when your system is upgraded via PTFs for the year 2000 support. An optional 4-digit year may replace the current 2-digit year.
2. The ID of the console is displayed only if the console does not have a name assigned on the CONSOLE statement in the CONSOLxx parmlib member.

2.1.3.2 Hardcopy Log Message Description

Some fields in the hardcopy log message prefix have more than one possibility for text, such as the ID of the console or the ID of a job. This prefix field is used as follows:

Console ID This field contains the console name that issued a command. For some extended MCS consoles, this name can be a user ID.

Job ID This field contains the JES2 job ID assigned to jobs, started tasks, or TSO logon users for messages related to that job.

For JES3 systems, this field is the job name that the message is issued for.

Console ID Example: In this example, HOLLICA is the console name that issued the `D U` command. The command response is the next three lines. The first line contains the console name for the command response and the next two lines contain the multi-line message ID 208, associated with the command response.

```
HOLLICA  00000290  D U,VOL=OS3RS1
HOLLICA  00000090  IEE457I 13.06.08 UNIT STATUS 208
     208 00000090  UNIT TYPE STATUS        VOLSER     VOLSTATE
     208 00000090  0CF0 3390 O             OS3RS1     PRIV/RSDNT
```

JES3 Job ID Example: In this example, ABCDE is the job name for which the messages were issued.

```
ABCDE    00000090  IEF403I ABCDE - STARTED - TIME=18.28.01
ABCDE    00000090  IEF404I ABCDE - ENDED - TIME=18.28.01
```

JES2 Job ID Example: In this example, JOB10845 is the job ID for which the messages were issued.

```
JOB10845 00000090  $HASP373 AYRESRX  STARTED - INIT    A - CLASS A - SYS
JOB10845 00000090  IEF403I AYRESRX - STARTED - TIME=15.19.26
```

2.2 Communications Task Processing

The MVS communications task is an interrupt driven system task in the CONSOLE address space. The communications task's wait service routine is a never ending task. It gets control after the communications task has been posted for work and routes control to the requested service routine. The service requests include external interrupt, attention, I/O completion, console output, and action message retention.

With few exceptions, the three macro services, WTO, WTOR, and DOM, are used to call communications task services:

WTO The Write to Operator (WTO) macro services allows you to write a message or multiple-line message to the consoles and hardcopy log. A console may be a display device, a printer, or a program that has activated an extended MCS (EMCS) console or receives messages through the subsystem interface.

For information on the use of this macro, see 6.1.1, "WTO and WTOR Macro Services" on page 135.

WTOR The Write to Operator with Reply (WTOR) macro causes one message requiring a reply to be written to one or more operator consoles and the hardcopy log. The macro also provides the information required by the system to return the operator reply to the issuing program.

When a WTO(R) macro is issued, the system assigns an identification number to the message and returns this number to the issuing program. When the program no longer requires this message displayed, it issues the DOM macro using the identification number that was returned from the WTO(R) request.

For information on the use of this macro, see 6.1.1, "WTO and WTOR Macro Services" on page 135.

DOM The delete operator message (DOM) macro service deletes an operator message or group of messages from the display screen of the operator's console and from the list of messages retained by MVS. When a program no longer requires a message to be displayed or retained, it issues the DOM macro to delete the message. The message or messages to be deleted are identified by the identification number that was returned when the WTO macro was issued.

An alternate method for identifying messages that are to be deleted is to specify the TOKEN parameter on the WTO(R) macro and to issue a DOM using the same TOKEN. The TOKEN parameter specifies a 4-byte identifier that is associated with messages. When a DOM using a TOKEN is issued, the message ID returned by WTO(R) is ignored, and the specified token value is used instead. Specifying TOKEN with DOM will delete all messages with that particular TOKEN.

MVS automatically invokes DOM for a WTOR when it receives the WTOR reply. You need only invoke DOM for a WTOR if the WTOR becomes obsolete before the system receives the reply. When the

system receives the reply, the message is no longer highlighted on the MCS console screen. The message is eventually removed from the screen of an MCS console.

For information on the use of this macro, see 6.1.3, “DOM Macro Service” on page 144.

Figure 3 on page 22 shows a high-level overview of the communications task processing.

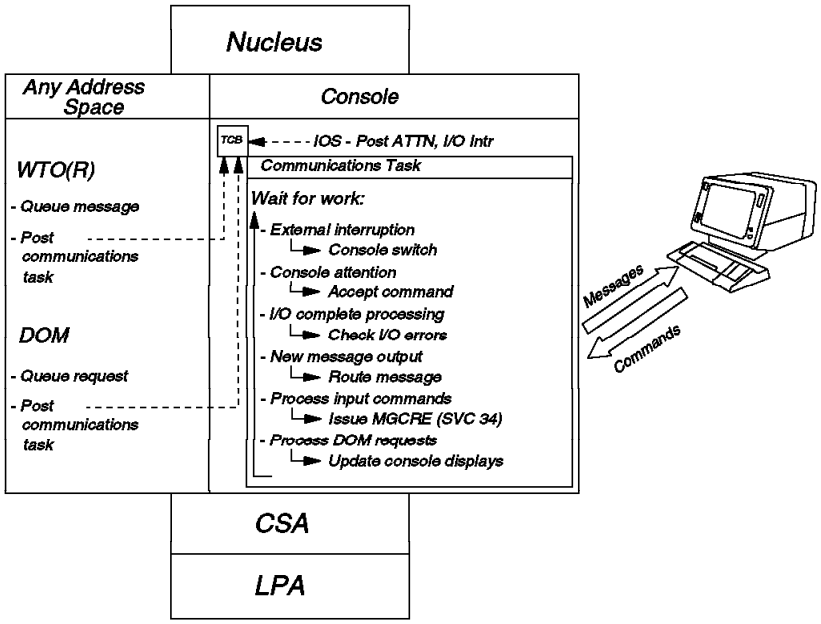


Figure 3. High-Level Overview of Communications Task Processing

In addition to the communication between user programs or the system routines and the various consoles, the communications task uses internal services for:

- Console device management
- Console attention processing
- I/O complete processing
- Console output processing
- External interrupts

2.2.1 Console Device Management

A device is defined as a multiple console support (MCS) console through the CONSOLE statement in the CONSOLxx parmlib member. Consoles should be defined as full-capability to be able to enter commands and receive status displays and messages by specifying USE=FC on the CONSOLE statement. The same device must also be specified in the IODF. A device, defined as a console, can be in three logical states:

Online The system can allocate the device to any user.

Offline The device is generally unavailable for the system to use. The device can be activated as a console.

Console The communications task can send messages to the device, and accept system commands from the device (if the device has input capability).

Note: If a device is defined as a console in the CONSOLxx parmlib member, that device cannot be dynamically reconfigured. The UCBs for the defined console devices are pinned through UCBPIN service during the system initialization and remain pinned even if the device is deactivated as a console. Pinning a UCB ensures that the UCB cannot be deleted or changed.

2.2.1.1 Console Switching

If MCS switches a console to an alternate (or when an operator switches a console as a result of the SWITCH command), the following console attributes are merged with those of the alternate:

ROUTCODE Routing codes

LEVEL Message levels

AUTH Command authority of MCS consoles

MSCOPE Message scope in a sysplex

UD Ability for the console to receive undelivered messages

Console switching occurs for both MCS and extended MCS consoles.

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

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, an operator can issue the following commands to attempt to recover the inactive console:

```
RESET CN(consname)
VARY CN(consname),ONLINE
```

2.2.1.2 No-Consoles Condition

If no consoles capable of command entry are active in a system or sysplex, a *no-consoles condition exists*. If you want to limit the selection of consoles that can become the master, use the NOCCGRP parameter on the INIT statement of the CONSOLxx parmlib member. See 3.2.3, “INIT Statement” on page 73.

Note: A no-consoles condition is only a bad situation if you want to have active MCS consoles. It is possible, though unusual, to run without any MCS consoles.

For a no-consoles condition, the operator can do any one of the following:

- Issue the following command (from an extended MCS or subsystem console) to activate a full-capability console that is offline and make it the master.

  ```
  VARY CN(consname),ONLINE
  ```

- Issue the following command (from an extended MCS or subsystem console) to change an active message stream or status display console to a full-capability console and make it the master.

  ```
  CONTROL V,USE=FC
  ```

- Press the attention interrupt key on the console device that is to become the master console, then press the external interrupt key on the system console to activate the device as the master console.
- In a sysplex, an operator can use the system console to IPL a system with a full-capability console (defined with AUTH=MASTER) into the sysplex.

See *MVS/ESA System Commands* for details of all operator commands.

2.2.2 Console Attention Processing

When a console operator enters a command, an I/O interrupt occurs which causes the communications task's attention ECB to be posted. The communication task then calls through SVC 72 the appropriate device service processor (DSP) to read the operator input from the device that signalled the interrupt.

2.2.3 I/O Complete Processing

The I/O complete processing handles I/O interruptions that occur for console I/O operations:

- A message is sent to a console device and there is no I/O error. The communications task flags the message serviced by this console and releases freed resources.
- An operator command is received from a console device and there is no I/O error. The command is entered to command processing through MGCR(E) macro (SVC 34).
- An permanent I/O error occurred during the previous I/O operation. The communications task attempts to switch the console to an alternate.

2.2.4 Console Output Processing

After a WTO or WTOR macro service has prepared a message (internally called WTO queue element (WQE)) for delivery to the consoles, the message is queued to all appropriate consoles and extended MCS consoles, as well as to any other systems in the sysplex. Messages are sent to other systems in the sysplex using the MVS cross-system coupling facility (XCF) services (see 2.3, "Message Flow in a Sysplex" on page 25). Queueing is performed as follows:

- If the message is routed by console name or ID, queue the message to the target console. Messages can also be routed with a special console ID 0 (zero), and they are sent to the master console. Messages are delivered to TSO users running the TSO/E OPERATOR command through address space TPUTs.
- If the message is being hardcopied, queue it to hardcopy and to all consoles receiving the hardcopy message set.

  Note: Extended MCS consoles can also be defined to receive the hardcopy log message set.

- All other messages are queued using their normal routing attributes; console name or ID, routing codes, message levels, message types, and so forth.

- If no recipients are found for a message meeting the UD criteria, the message is queued to UD recipients (see 2.2.4.1, “Undelivered Messages” on page 25), SYSLOG, and OPERLOG.

Once all message queueing is complete, the console device processing is invoked by issuing SVC 72.

2.2.4.1 Undelivered Messages

If action messages (those with descriptor codes 1, 2, 3, or 11), informational messages with descriptor code 12, or WTOR messages are not delivered to any console in the configuration, you can specify through the UD parameter of the CONSOLE statement and on the HARDCOPY statement in the CONSOLxx parmlib member the consoles that are to receive undelivered messages. If there are no consoles receiving UD messages, they are sent to the system console. If UD=(Y) is specified on the HARDCOPY statement, they will not be sent to the system console.

Note: The master console defaults to UD(Y).

Operators can use the following command to change the UD attribute for a console:

```
VARY CN(*),UD=Y
```

To change the UD attribute for the hardcopy log, specify:

```
VARY HARDCPY,UD=Y
```

2.2.5 External Interrupts

When no full-capability consoles are available in a system or sysplex, an operator can select a console from an alternate console group specified on the INIT statement and activate it as the master console. When the operator presses the attention interrupt key on any console device that is a member of the alternate console group and then presses the external interrupt key of the system console, the communications task can activate the console as the master console.

The NOCCGRP parameter on the INIT statement of the CONSOLxx parmlib member specifies a master console group and is used to prevent some consoles from becoming the master.

2.3 Message Flow in a Sysplex

In a sysplex, a message is routed to all active consoles on all systems that are eligible to receive that particular message. Figure 4 shows a typical message flow within a sysplex.

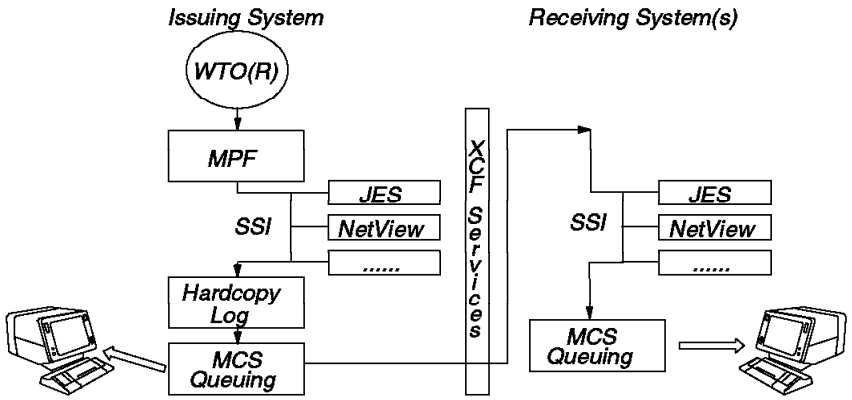


Figure 4. Message Flow in a Sysplex

The following sections discuss the message flow shown in Figure 4. It is important to understand this flow since in a sysplex environment, message flow is changed to send messages issued on one system to other systems in the sysplex using XCF services.

2.3.1 WTO(R) Processing

The MVS write to operator (WTO) and the write to operator with reply (WTOR) macro services cause messages to be routed to the consoles and the hardcopy log.

2.3.2 Message Processing Facility

First on the issuing system, the message is processed by the message processing facility (MPF). This processing is based on entries in the MPFLSTxx parmlib member.

MPF processing allows an installation to influence how WTO and WTOR messages are to be processed. Through the MPFLSTxx member, you can specify some processing options for a message:

- Suppress message display - if a message is suppressed, it is logged to the hardcopy log, and does not appear on any console. For MPF to suppress messages, the hardcopy log must be active.
- Retain action message - retention means that action messages are saved by the action message retention facility (ARMF) so that operators can view them later.
- Automation eligibility - specifying that the message is eligible for automation. An automation subsystem, such as NetView, can look at the message and can perform predefined operator actions. Note that TSO/E and other EMCS consoles may be activated to receive automation messages.
- Invoke an installation-written exit - you can write exit routines to process WTO and WTOR messages. MPFLSTxx can specify which messages are to be processed by which exit routine. The exit can alter the message text and the way in which the message is to be processed.
- Message presentation - the MPFLSTxx member may also control the color, intensity and highlighting options of messages displayed on 3270-X consoles.

For full details of MPFLSTxx options and parameters, see *MVS/ESA Initialization and Tuning Reference*.

The message processing facility is activated as part of communications task initialization. The initial set of MPFLSTxx members are indicated on the MPF parameter on the INIT statement in CONSOLxx parmlib member. You can specify multiple MPFLSTxx members on the MPF parameter. In a sysplex, MPF processing has system scope; thus, you must plan MPF processing on each system in the sysplex.

Multiple MPFLSTxx members allow you to define separate MPF members for specific message processing functions or sets of messages. For example, you might specify two MPFLSTxx members to handle different automation procedures. Or you might have one MPFLSTxx member handle messages for suppression and another to handle messages for automation.

Note: The system default considers all messages as eligible for automation.

Operators can use the SET MPF command to activate or deactivate sets of MPFLXTxx members as required (for example, during shift changes or for workload balancing).

If a system does 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, and 11).
- MPF does not suppress messages.
- No installation message processing exits other than the IEAVMXIT exit can gain control to process messages.

2.3.2.1 Message Suppression Exit

The IEAVMXIT exit, when activated, gains control for all WTO and WTOR messages that do not have a message specific exit specified on the message management statement in the MPFLSTxx member. It can change routing codes, descriptor codes, and message texts and perform other message processing; it can also override the MPF processing.

Note: IEAVMXIT and MPF exit routines are not invoked during the initial processing of synchronous WTOs and WTORs. The exit routine will be invoked when the message is later issued through SVC to the hardcopy log.

The status of the IEAVMXIT exit can be queried with the following command:

```
CONTROL M,REF
```

The system displays the status of the action message retention facility, the status of installation exit IEAVMXIT, and the limit of the number of WTO and WTOR buffers. You can dynamically activate or deactivate the IEAVMXIT exit with the following command:

```
CONTROL M,UEXIT=
```

See *MVS/ESA Installation Exits* for details of the IEAVMXIT exit and the installation message processing exits.

2.3.3 SSI Processing

Following MPF processing, the message is broadcast to all active subsystems that request to receive control for the WTO SSI function code 9. The susbsystem must use the IEAVG700 interface to indicate that all WTO and WTORs are to be broadcast. The message is presented to each subsystem in turn. Each subsystem may inspect the message and process it as appropriate. A subsystem can alter write-to-operator queue element (WQE) fields, in which case later subsystems on the SSI will see the changed WQE. A WQE is an internal control block that contains the message text and all related information for that message. The IHAWQE macro maps the WQE fields. See also *MVS/ESA Data Areas, Volume 5 (SSAG-XTLST)*.

For example, when NetView is using the SSI rather than an extended MCS console for MVS communication, NetView on the SSI inspects all messages to see whether they are marked by MPF as eligible for automation. NetView intercepts automation message text and selected attributes from the WQE and sends the data to the NetView address space for further processing. NetView does not modify the actual WQE.

2.3.4 Hardcopy Log Processing

After the message has been inspected by all active subsystems, it is written to the hardcopy log (usually the SYSLOG data set, the operations log (OPERLOG), or both) unless hardcopy logging is suppressed by an exit. OPERLOG is a log stream maintained in a coupling facility that uses the system logger to record and merge communications about programs and system functions from each system in a sysplex. See 5.1, “Defining the System Logger” on page 124. The messages are logged using message data blocks (MDBs), which provide more data than is recorded in the SYSLOG. The IEAVM105 macro maps the MDB fields. See *MVS/ESA Data Areas, Volume 3 (IVT-RCWK)*.

2.3.5 MCS Queueing Processing

Finally the message is routed for display on the appropriate MCS and extended MCS consoles. The routing may require message transportation using XCF services to other systems in the sysplex because some receiving consoles may not be physically attached to the system where the message was issued.

After the XCF transportation on the receiving system, the message goes through the SSI loop, but it is not logged, and finally the message is processed by the message queueing tasks to be displayed on the consoles.

If a message is destined for a specific console that is not active in the sysplex, it is logged and discarded unless it is an action message or WTOR message, in which case it is processed as an undelivered message. It is sent to all active consoles receiving UD messages. The master console is a UD receiver by default.

Messages that are already “delivered” to an active extended MCS console, but not yet “retrieved,” are purged from the MCS queues when the console is deactivated; that is, unprocessed queued messages are not rerouted.

The MSCOPE specification on the CONSOLE statement in the CONSOLxx parmlib member allows you to screen those systems in the sysplex from which a console is to receive messages not explicitly routed to the console.

For a description of how each message is queued to a console on each system in the sysplex, see 2.2.4, “Console Output Processing” on page 24.

2.4 Operator Commands

MVS system commands are used for system status displays and to control the system itself and MCS consoles. Subsystems, for example, JES2 and JES3, provide their own sets of operator commands for subsystem control.

MVS operator commands enter processing through the MGCR(E) (SVC 34) macro service, as shown in Figure 5. If the commands are entered from a console device, the communications task internally issues the MGRCE macro for the command. Authorized programs, when they enter an operator command, must use the same MGCR(E) macro service. Programs, before entering commands, should activate an extended MCS console using the MCSOPER macro service. Activating the extended MCS console allows the program to receive command responses using the MCSOPMSG macro service.

Note: Programs can issue commands through the pseudo master console (console ID 0); however, in this case command responses are not routed back.

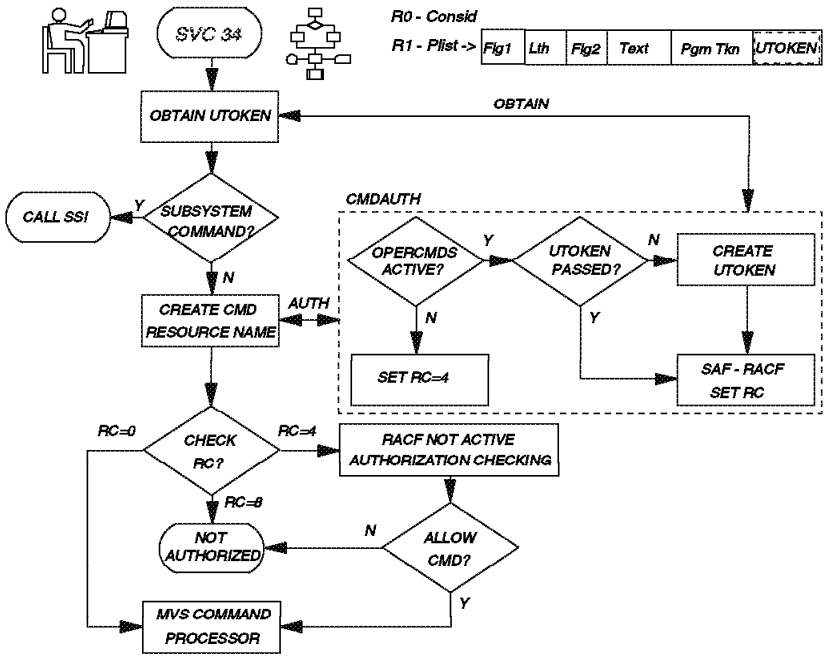


Figure 5. Overview of MVS Command Authorization Processing

MVS command processing consists of command scheduling and command execution. Command scheduling provides for a command task’s synchronization with other events in a system. Command execution is the performance of the function specified by the command itself. The execution of the command functions are implemented through a number of individual command processors, each performing a specific function.

The front end of command processing is common for all commands. It includes translation of the input command to upper case, transporting the command to the target system when required, writing the command (in most cases) to the

hardcopy log, invoking command installation exits, broadcasting the command through the subsystem interface, command authorization checking, and routing the command to the appropriate command processor.

2.4.1 MCS Consoles Command Authorization Checking

The AUTH parameter on the CONSOLE statement of CONSOLxx parmlib member defines the initial command authority of a full-capability console.

Console security means controlling which commands operators can enter on their consoles to monitor and control MVS.

MVS commands are assigned to one of five *command groups* according to command function and the console authorization required to issue them. The command groups and the MVS command in each group are:

- Informational commands - INFO command authorization

CONTROL [4]	LOG	REPLY [5]	STOPMN
DEVSERV	MONITOR	ROUTE	STOPTR [1]
DISPLAY [1]	MSGRT [1]	SEND	TRACK [1]

- System control commands - SYS command authorization

ACTIVATE	MODE	SET GRSRNL	START
CANCEL	MODIFY	SETDMN	STOP
CHNGDUMP	PAGEADD	SETETR	SWITCH SMF
DUMPDS	PAGEDEL	SETPROG	TRACE (with CT, ST, or STATUS)
HALT [3]	RELEASE [2]	SETSMF	WRITELOG
HOLD [2]	RESET	SETSMS	
LIBRARY	SET	SLIP	

- I/O control commands - IO command authorization

ASSIGN	UNLOAD	VARY ONLINE/OFFLINE [6]
MOUNT	VARY NET [3]	VARY name or devnum
SWAP	VARY PATH	

- Console control commands - CONS command authorization

CONTROL [4]	VARY ONLINE/OFFLINE [6]
VARY CN...,ALTGRP=...	VARY PATH
VARY CONSOLE,ALTCONS=...	VARY name or devnum

- Master level authority commands - MASTER command authorization

CONFIG	QUIESCE	VARY CN/CONSOLE,AUTH=...
CONTROL [4]	RESET CN	VARY GRS
DUMP	SETLOGRC	VARY HARDCPY
FORCE	SETXCF	VARY MSTCONS
IOACTION	SWITCH CN	VARY OFFLINE,FORCE
	TRACE (MT)	VARY XCF

Notes:

[1] CONS command group when message routing is specified.

[2] HOLD and RELEASE are related to the Telecommunications Access Method (TCAM).

[3] HALT NET and VARY NET are related to the Virtual Telecommunications Access Method (VTAM).

[4] CONTROL is in the INFO command group except when:

- Rerouting the message queues of any other full-capability MCS console -- MASTER.
- Message routing is specified -- CONS.
- Changing or displaying the status of the action message retention facility, the number of allowed message buffers, or the status of WTO user exit IEAVMXIT -- MASTER.

- In a sysplex, changing the maximum time to wait for aggregated command responses -- MASTER.
- Increasing the number of reply IDs -- MASTER.

5 An operator can reply to any message that the console is eligible to receive. Any console with MASTER authority can reply to any message.

6 VARY CN,OFFLINE and VARY CN,ONLINE require CONS. Without the CN keyword, VARY OFFLINE and VARY ONLINE require IO authority.

You can enter informational commands from any full-capability console. However, to enter system control, I/O control, or console control commands from a secondary console, that particular command group must be assigned to that console. If you enter a command at a console where it is not authorized, MVS rejects the command and sends an error message to the issuing console.

2.4.1.1 Master Authority Consoles

At a console with master authority, you can enter *all* operator commands. Any console with AUTH(MASTER) in the CONSOLxx parmlib member has master console authority. See 3.3.1, “Master Console in a Sysplex” on page 82.

2.4.2 Command Authorization without RACF Checking

When RACF is not active or when OPERCMDS RACF class is not active, MVS command processing accepts all commands from a console that belongs to command groups with the same or lower authorization requirement as assigned for the console. For example, consoles with MASTER authority can issue all commands, including those that affect other consoles (including extended MCS consoles), and consoles with IO authority can issue in the IO command group.

2.4.3 Command Authorization with RACF Checking

RACF command authorization checking requires RACF 1.9 or later to be installed and active, and OPERCMDS class to be active. RACF command authorization checking overrides MCS command authority. The command issuer′s RACF profile and group authority determine what commands can be successfully entered into the system (through the MGCR(E) macro service).

When a user requests access to the MVS system through a TSO logon, a batch job′s submission through JES, an APPC transaction, or through a START command, RACF performs user authentication (verification). Based on the specifications on the verify request, RACF determines whether the requesting user is authorized to enter the system. If the user is authorized to enter the system, RACF creates an accessor environment for the user that contains the security characteristics of a user.

MVS command processing extracts the encapsulation (representation of the security characteristics) of a user (UTOKEN) during the command authorization checking when the command (MGCR(E)) originates from an address space other than the CONSOLE address space. The UTOKEN is passed to the RACROUTE processing that uses it for command authorization along with the resource name for the command and the command text. However, when the command issuer provides a UTOKEN as part of MGCR(E) macro parameters, command processing passes that UTOKEN to RACF to validate the authority of the issuer.

Commands originating from an MCS console (CONSOLE address space) are processed differently depending on whether an operator is logged on to the console or not. If the operator is:

logged on The operator's UTOKEN is used to validate the authority.

not logged on An internal UTOKEN is used that causes a CMDAUTH return code 4 (the security monitor could not make an authority decision). The command authority checking is then based on the console's authority level and the authority level required by the command group. The process is effectively the same as described in 2.4.2, "Command Authorization without RACF Checking" on page 31.

See Appendix B, "Defining Command Resource Names to RACF" on page 183 for an example on how to define OPERCMDS class command resource names to RACF.

2.4.3.1 Authorizing Console Access

With RACF 1.9 or later installed, the installation can require or allow the operators to log on to any MCS console. The operator's RACF profile and group authority determines what commands can be issued from the console when the RACF OPERCMDS class is active. For a list of MVS commands and their profile names, see *MVS/ESA Planning: Operations*.

Operators can be required to log on to and log off from MCS-managed consoles by specifying LOGON (REQUIRED) or LOGON (AUTO) options on the DEFAULT statement in the CONSOLxx parmlib member. When the RACF CONSOLE class is active and a console being used is protected by a profile in the CONSOLE class, RACF ensures that the person attempting to log on has the proper authority to do so. To be able to log on, an operator on an MCS console must have READ access to the profile in the CONSOLE class named:

console-name

LOGON (REQUIRED) LOGON (REQUIRED) specifies that an operator must log on to an MCS console before issuing commands from that console. If an operator is not logged on to the console, the system rejects commands issued from that console except under the following conditions:

- When issuing commands from the master console before RACF is active
- When issuing the VARY devnum,MSTCONS command from any console to assign the master console (when there is currently no master console assigned) before RACF is active.

LOGON (AUTO) LOGON (AUTO) specifies that consoles are automatically logged on when the consoles are activated. The automatic LOGON uses the console name as the logon user ID.

Note: If you do not explicitly name a console (NAME parameter on CONSOLE statement in CONSOLxx parmlib member), the system generates a name by using the EBCDIC representation of the console identifier. This identifier is valid for the duration of the IPL, and may change on subsequent IPLs. The console name generated by the system is used in system messages to identify the console, but you cannot use that name in commands.

LOGON (OPTIONAL) LOGON (OPTIONAL) specifies that the operators can optionally log on to the MCS consoles. This option is the default. You are required to log on to a console if you have to enter a command that belongs to a higher command group than what is accepted from the console. After logon you can enter all commands that are authorized for your user ID. The opposite is also true. If your user ID is not authorized to enter any command, after logon all commands entered from the console are rejected.

The activation of extended MCS consoles can be controlled through the RACF OPERCMDS class. When the OPERCMDS class is active and a console being activated is protected by a profile in the OPERCMDS class, RACF ensures that the user attempting to activate an extended MVS console has the proper authority to do so. The user of the extended MCS console must have READ access to a profile in the RACF OPERCMDS class named:

```
MVS.MCSOPER.console-name
```

See Appendix C, “Controlling MCS Console LOGONs through RACF” on page 185 for steps to activate RACF CONSOLE class controls.

2.4.3.2 Command Resource Names

Command resource names, defined in your RACF data base, are used by the CMDAUTH service to verify whether an operator is authorized to issue a given command. The command resource names (sometimes called entity names) enable you to logically group operator commands and name each group for the purpose of controlling RACF access to the commands.

A command resource name can include up to four parts: a system identifier, the command or a variation of the command, a command qualifier, and a command object. These parts enable you to define a naming hierarchy that can identify specific commands or command subsets. IBM recommends you use the syntax:

system_identifier .command [.command_qualifier [.command_object]]

Where:

system_identifier Identifies the system, subsystem, or application to which the command belongs. For example, IBM uses MVS to identify MVS operator commands, JES2 to identify JES2 commands, and JES3 to identify JES3 commands.

command Identifies a specific command or some variation of a command. Where possible, use the command name. In cases where the command name does not provide the level of identification you require, use a variation of the command name.

command_qualifier Allows you to more precisely identify the command variation in question.

command_object Identifies the object of the command. Including the command_object as part of the command resource name enables you to control access to commands based on the object the command affects.

The following is an example of the MVS CANCEL command and its command resource name with a command_qualifier and a command_object:

Command	Command resource name
CANCEL jobname	MVS.CANCEL.JOB.jobname MVS.CANCEL.STC.jobname

All command resource names must include a system identifier and the command name or a variation of the command name. The use of the command qualifier and the command object is optional and depends on the naming structure you want to define.

For a list of the MVS provided command resource names, see *MVS/ESA Planning: Operations*.

Figure 5 on page 29 shows an overview of the MVS command authorization processing.

2.4.4 EMCS Command Authorization Checking

An extended MCS console is established dynamically by an authorized program through the MCSOPER REQUEST=ACTIVATE macro service. Once the extended MCS console is activated, the program can receive messages and command responses by issuing the MCSOPMSG macro, and can issue commands by issuing the MGCRE macro.

When activating the extended console, the console attributes, such as its command authority, routing codes, message dataspace size, need for a migration ID, and whether it is to receive the hardcopy message set are specified:

- In the OPERPARM segment of the user's RACF profile. See 4.1.3, "EMCS With RACF Control" on page 101.
- In the OPERPARM data area MCSOP, mapped by IEZVG111, when it is passed as the parameter to the MCSOPER macro service. See 4.1.3.1, "OPERPARM Segment" on page 101 and 4.1.3.3, "MCSOP Data Area" on page 103 for more details.
- Through system defaults. See 4.1.3.2, "OPERPARM Segment Defaults" on page 103.

As the default, the "MCSOPER activate an extended MCS console" request checks for the attributes in the order listed above. First it looks for the OPERPARM segment in the user profile of the requestor. If present, the OPERPARM segment definitions are used. Then it checks the OPERPARM data passed with the extended console activation request. If specified, they are used. Otherwise the system defaults are applied.

Note: The program activating an extended MCS console can force the OPERPARM data to be used by controlling the setting of MCSOVRDY and MCSOVRDN bits in the OPERPARM data area. The following is the order of processing for OPERPARMs as determined by the bits:

MCSOVRDY If the MCSOVRDY bit is on, this indicates to override the security product. Processing uses the OPERPARM data area to set the extended console's attributes.

MCSOVRDN If the MCSOVRDN bit is on, this indicates to not override the security product, which is the default option. Processing searches the security product for an OPERPARM segment. If no segment exists, processing then uses this data area to set the extended console's attributes.

2.4.5 Command Authorization Service

Command processors that need to find out whether a user is SAF-authorized to issue a command should use the command authorization service (CMDAUTH). To use the command authorization service, a command processor issues the CMDAUTH macro. The service provides a return code that indicates the user's authorization for the command:

RC=0 Successful completion - the command is authorized

RC=4 The security monitor could not make an authority decision

RC=8 Successful completion - the command is not authorized

Under certain conditions, the command authorization service is unable to make a decision. For example, a decision cannot be made if RACF 1.9 or later is not installed, is inactive, OPERCMDS RACF class is not active, a profile is not found for the command in the RACF data base, or generic profiles are defined but generic processing is not active for OPERCMDS class. A return code and reason code together indicate why a decision could not be made. IBM recommends the use of this service by all command processors that must verify the authorization status of a user.

You must supply to the CMDAUTH request the resource name of the command being checked and an indication of the access authority for which the service is to check. In addition, the CMDAUTH macro requires either the command text that is to be authority checked and the user security token (UTOKEN) that is to be used for the authority check or a command input buffer (CIB).

The user security token is passed to a command installation exit through the exit's parameter list (CMDXUTOKN field). Applications that intercept their commands from the subsystem interface receive the user security token through the subsystem option block (SSOB) extension for command processing (SSCM field SSCMUTOK).

The CMDAUTH macro has a number of parameters that functionally correspond to the RACROUTE parameters. The CMDAUTH macro and its use is described in *MVS/ESA Programming: Authorized Assembler Services Reference, Volume 1 (ALESERV-DYNALLOC)* and *MVS/ESA Programming: Extended Addressability Guide*. The RACF macros are described in *External Security Interface (RACROUTE) Macro Reference*.

See Appendix D, "Display System Groups Command Exit" on page 187 for an example on how to use the CMDAUTH service.

2.5 System Symbols in Commands

MVS/ESA 5.2.0 provides support for symbolic substitution in SYS1.PARMLIB member parameters, MVS and JES commands, and started task JCL.

Static system symbols are defined at system initialization and remain fixed for the life of an IPL. Static system symbols are used to represent fixed values such as system names and sysplex names. There are two types of static system symbols:

- System-defined static system symbols have their names defined to the system. You define substitution texts or accept system default texts for static system symbols.
- Installation-defined static system symbols are defined by you. You specify their names and substitution texts in the IEASYMxx parmlib member. See Figure 6 and Figure 16 on page 54 for installation defined static system symbols.

System symbols represent the values in commands that are assigned to the symbols in the IEASYMxx parmlib member. Each system uses its own values for the system symbols, and replaces the system symbols with those values when processing the commands. Figure 6 shows you a sample definition of parmlib member IEASYMxx.

```
SYSDEF          SYMDEF(&CLNLST='AA')              /* CLONING NAME */
                SYSCLONE(&SYSNAME(3:2))
                SYMDEF(&IEASYSP='XX')
                SYMDEF(&APPCLST1='&SYSCLONE.')
                SYMDEF(&CMDLIST1='&SYSCLONE.,00')
                SYMDEF(&CONSOLXX='00')
                SYMDEF(&COUPLEXX='00')
                SYMDEF(&SSNLST01='00')
SYSDEF          HWNAME(P101)
                LPARNAME(A1)
                SYSPARM(XX)
                SYSNAME(SC52)
                SYMDEF(&APPCLST1='XX')
                SYMDEF(&CMDLIST1='XX,00')
                SYMDEF(&SSNLST01='XX')
SYSDEF          HWNAME(P101)
                LPARNAME(A2)
                SYSPARM(53)
                SYSNAME(SC53)
                SYMDEF(&CMDLIST1='XX,00')
SYSDEF          HWNAME(P201)
                LPARNAME(A3)
                SYSPARM(50)
                SYSNAME(SC50)
                SYMDEF(&SSNLST01='XX')
```

Figure 6. SYS1.PARMLIB Member IEASYMxx Example

The current settings and values of the symbols can be displayed using the MVS command D SYMBOLS. The command shown in Figure 7 on page 37 displays the system symbols for member SC52.

```
D SYMBOLS
 IEA007I STATIC SYSTEM SYMBOL VALUES 495
         &SYSCLONE. = 52
         &SYSNAME.  = SC52
         &SYSPLEX.  = WTSCPLX1
         &APPCLST1. = XX
         &CLNLST.   = AA
         &CMDLIST1. = XX,00
         &IEASYSP.  = XX
         &LNKLIST2. = 00
         &LOGREC1A. = LOG
         &LOGREC2A. = STR
         &LOGREC3A. = EAM
         &LPALIST2. = 00
         &LPALSTJ1. = AA
         &MLPALST1. = AA
         &RSULST01. = 00
         &SSNLST01. = XX
```

Figure 7. D SYMBOL Command for SYSNAME SC52

2.5.1 System Symbols in Operator Commands

To display the static system symbols and the associated substitution texts that are in effect for a system, you enter the DISPLAY SYMBOLS command on that system, as shown in Figure 7.

The following rules apply for system symbol usage in operator commands:

- A system symbol cannot be used as part of the prefix or the command verb.
- Do not use command installation exits to add or change system symbols in command text. MCS does not substitute text for system symbols that are added or changed through those exits.
- Do not route commands that contain system symbols to pre-MVS/ESA 5.2 systems in the sysplex.

System symbols are supported for most MVS commands. Do not specify system symbols, as no substitution is done, for the following MVS commands:

- LOGON
- VARY CN(*),ACTIVATE
- REPLY (in the text for a synchronous WTOR)
- Commands with no keywords

Both the original and substituted command are logged in the hardcopy log.

User exits and subsystems receiving commands from the SSI see the command after substitution has occurred. The original command is available, but should not be modified with any additional system symbols.

Assume that you want to display the status of the system consoles on the system group P301 systems (SC42 and SC43 as defined in Figure 11 on page 46). The system consoles on the systems in group P301 have default names defined, which is the system name. Instead of entering a different command to display the unique system consoles, you can route one command with system-defined static system symbol &SYSCLONE to the system group to display the consoles.

```
RO P301,D C,CN=SC&SYSCLONE

 SC42     RESPONSES ----------------------------------------------------
 IEE889I 19.04.30 CONSOLE DISPLAY 469
 MSG: CURR=6    LIM=1500 RPLY:CURR=4    LIM=999  SYS=SC42      PFK=00
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS --------------
  SC42              102  COND=A      AUTH=MASTER       UD=N
   SYSCONS               MFORM=M         LEVEL=ALL
   SC42                  ROUTCDE=NONE
                         CMDSYS=SC42
                         MSCOPE=*ALL
 SC43     RESPONSES ----------------------------------------------------
 IEE889I 19.04.30 CONSOLE DISPLAY 761
 MSG: CURR=4    LIM=1500 RPLY:CURR=4    LIM=999  SYS=SC43      PFK=00
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS --------------
  SC43              110  COND=A      AUTH=MASTER       UD=N
   SYSCONS               MFORM=M         LEVEL=ALL
   SC43                  ROUTCDE=NONE
                         CMDSYS=SC43
                         MSCOPE=*ALL
```

&SYSCLONE Represents a one or two character value that can be set through parmlib member IEASYMxx or it defaults to the last two non-blank characters of the system name if no explicit value is assigned. &SYSCLONE is intended to be an abbreviated identifier for a MVS system image. During IPL, the &SYSCLONE value for a system may be checked against the values for the active systems in the sysplex (at the MVS/ESA 5.2 level), and if a duplicate &SYSCLONE value already exists on any system, it is rejected.

2.6 Command Flow and Routing in a Sysplex

Figure 8 describes the command flow and routing in a sysplex environment.

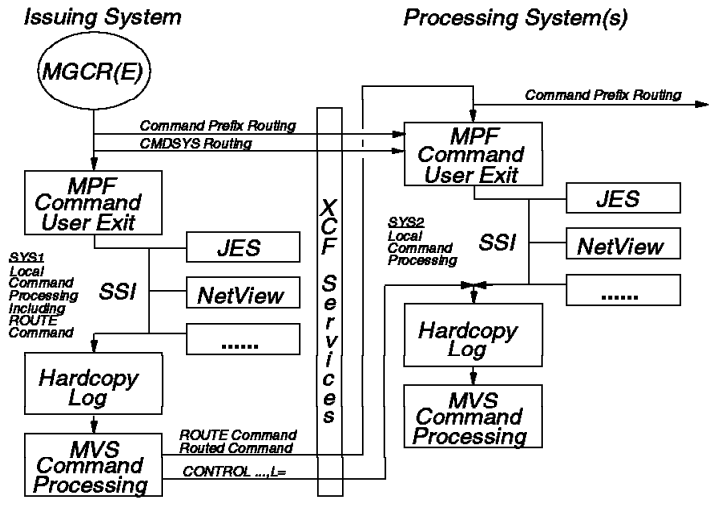


Figure 8. Command Flow and Routing in a Sysplex

When an operator command is entered through the MGCR(E) macro service (see Figure 8), the following processing flow takes place:

- **Command prefix and CMDSYS routing**

  If the command contains a prefix or the console has a CMDSYS specification that directs the command to another system, the command is immediately transmitted using XCF services to the processing system. See 2.7, “Command Routing in a Sysplex” on page 40.

  Note: For command prefix and CMDSYS routing, the command is first transported to the receiving system before any system symbol substitution takes place.

  A REPLY command is sent to the system where the WTOR was issued. If the REPLY command text contains system symbols, substitution occurs on the receiving system.

- **MPF command user exit**

  If the command is not transmitted to another processing system, it is processed on the issuing system by the installation MPF command exit routines. The exits are specified using the .CMD statement in the MPFLSTxx parmlib member. These exits can perform authority checking, modify the command text, or the command processing.

  Note: For commands containing system symbols, substitution has occurred before the exit is entered.

- **SSI processing**

  The command is then broadcast on the subsystem interface (SSI) to all active subsystems. Each subsystem inspects the command and decides whether to process it. The subsystems base the decision on the command prefix characters of the command string. For example, by default, NetView looks for a percent sign (%) and processes the commands starting with the % sign.

  When a subsystem decides to process the command, the command is passed to subsystem processing, and a return code is set to indicate that the command was processed by a subsystem.

  Note: At this point in processing, all system symbol substitution has occurred. The original command text is also available.

- **Hardcopy log**

  Once the command has been examined by all active subsystems, it is logged to the hardcopy log (usually SYSLOG or OPERLOG).

  Note: The hardcopy log contains the command before any system symbols contained in the command have been substituted and, it also contains the command after substitution has occurred.

- **MVS command processing**

  If none of the subsystems has marked the command as having been processed, it is assumed to be an MVS command and is passed to the appropriate MVS command processor. If a command processor does not exist, an error message is issued stating that the command is invalid.

  Note: For system symbols in a ROUTE command, see 2.7.3.2, “MVS ROUTE Command and System Symbols” on page 45.

2.7 Command Routing in a Sysplex

Command routing is controlled by several options in a sysplex. There are four options that can be used to send commands to another processor; they are:

- CMDSYS parameter on the CONSOLE statement in the CONSOLxx member of parmlib or the K V operator command
- Using the command prefix facility (CPF), an example is the IEECMDPF program in SYS1.SAMPLIB
- The ROUTE command
- CONTROL command with the L= command operand

2.7.1 CMDSYS Parameter

Consoles, both MCS and extended MCS, can have an associated CMDSYS value, which specifies the system to which commands issued on the console are to be sent. This provides an implicit command routing allowing a console that is physically attached to one system to be logically associated with another system.

The CMDSYS is specified for:

MCS consoles Use the CMDSYS parameter on the CONSOLE statement in CONSOLxx parmlib member. CMDSYS can also be specified using the `CONTROL V,CMDSYS=sysname,L=name` operator command.

EMCS consoles For extended MCS consoles, use the OPERPARM segment of the user's RACF profile or specify the OPERPARM data in the MCSOPER macro request. CMDSYS can also be specified using the `CONTROL V,CMDSYS=sysname,L=name` operator command.

Notes:

There are some commands, LOGON/LOGOFF, TRACK/STOPTR, ROUTE and some variations of CONTROL, which are not affected by the CMDSYS values. These commands are always processed on the issuing system.

The REPLY command is always processed on the system where the WTOR was issued.

The installation command exits and the active subsystems are invoked only on the target system and the command is logged on the target system. The command response is sent back to the originating console.

2.7.2 Command Prefix Facility

The command prefix facility (CPF), whose entries are registered with MCS through the CPF macro, allows an application to use command prefixes to associate an operator command with a "target" system. The registered command prefixes are made known to all systems in the sysplex. A registered command prefix allows operators or authorized applications to route a prefixed command to the target system in the sysplex. The application can be an installation exit, a subsystem, or an installation-written program.

Note: The CPF processing is independent of CMDSYS processing and is taken before the CMDSYS routing.

2.7.2.1 IEECMDPF Sample Program

You can run IEECMDPF (an IBM-supplied sample program in SYS1.SAMPLIB) to define system names as a command prefix. This program is intended to be executed on each system in a sysplex (for example, through a START command in a common COMMNDxx PARMLIB member) to create a command prefix for each system equal to its system name. This allows an installation to direct a single-system command to a given system by simply preceding the command with the system name.

CPF also ensures that two or more systems do not have the same or overlapping prefixes, which helps prevent confusion.

2.7.2.2 Sysplex Configuration

Figure 9 is the configuration that is used in this book for all the commands and illustrations. The sysplex consists of a JES2 MAS configuration and two JES3 complexes.

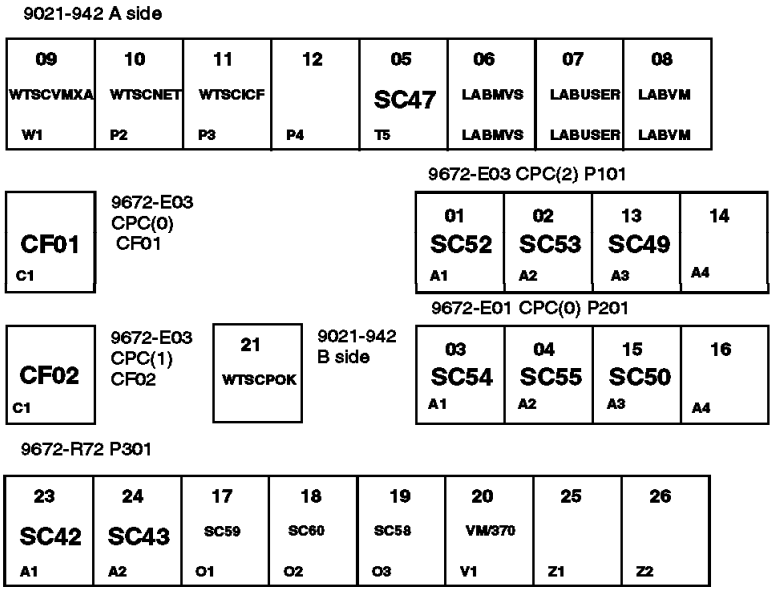


Figure 9. Sysplex Configuration

Figure 9 shows the following MVS images in the sysplex with the following information in each box:

01 The system image number

SC52 The system name (SYSNAME) of the MVS image

A1 The LPAR name

The sysplex consists of the JES2 MAS and two JES3 global configurations as follows:

JES2 MAS The JES2 MAS consists of systems SC42, SC43, SC47, SC49, SC50, SC52, SC53, and SC54.

JES3 This JES3 complex consists of SC50 as the global and SC49 as the local.

JES3 This JES3 complex consists of SC43 as the global.

Note: On systems SC50, SC49, and SC43, JES3 is the primary job entry subsystem and JES2 is a secondary job entry subsystem.

Appendix E, "CONSOL00 Parmlib Member" on page 191 contains the CONSOLxx definitions for the sysplex configuration shown in Figure 9 on page 41.

2.7.2.3 Displaying Command Prefixes

The DISPLAY OPDATA command lists all currently registered command prefixes and their attributes.

```
D OPDATA
IEE603I 14.42.58 OPDATA DISPLAY 840
 PREFIX     OWNER      SYSTEM     SCOPE     REMOVE   FAILDSP
 $          JES2       SC50       SYSTEM    NO       SYSPURGE
 $          JES2       SC52       SYSTEM    NO       SYSPURGE
 $          JES2       SC54       SYSTEM    NO       SYSPURGE
 $          JES2       SC53       SYSTEM    NO       SYSPURGE
 $          JES2       SC49       SYSTEM    NO       SYSPURGE
 $          JES2       SC43       SYSTEM    NO       SYSPURGE
 $          JES2       SC42       SYSTEM    NO       SYSPURGE
 $          JES2       SC47       SYSTEM    NO       SYSPURGE
 $          JES2       SC55       SYSTEM    NO       SYSPURGE
 *          JES3       SC50       SYSTEM    NO       PURGE
 *          JES3       SC49       SYSTEM    NO       PURGE
 *          JES3       SC43       SYSTEM    NO       PURGE
 %          JES3       SC50       SYSPLEX   NO       PURGE
 @          JES3       SC43       SYSPLEX   NO       PURGE
 SC42       IEECMDPF   SC42       SYSPLEX   YES      SYSPURGE
 SC43       IEECMDPF   SC43       SYSPLEX   YES      SYSPURGE
 SC47       IEECMDPF   SC47       SYSPLEX   YES      SYSPURGE
 SC49       IEECMDPF   SC49       SYSPLEX   YES      SYSPURGE
 SC50       IEECMDPF   SC50       SYSPLEX   YES      SYSPURGE
 SC52       IEECMDPF   SC52       SYSPLEX   YES      SYSPURGE
 SC53       IEECMDPF   SC53       SYSPLEX   YES      SYSPURGE
 SC54       IEECMDPF   SC54       SYSPLEX   YES      SYSPURGE
 SC55       IEECMDPF   SC55       SYSPLEX   YES      SYSPURGE
```

For the definitions of the command prefixes for the following owners:

JES2 See 7.1.1, "Defining the JES2 Command Prefix" on page 153.

JES3 See 8.7, "JES3 5.2.1 and MCS CPF" on page 173 and Figure 55 on page 176.

IEECMDPF See 2.7.2.1, "IEECMDPF Sample Program" on page 41.

2.7.2.4 Command Prefix Attributes

Registered command prefixes have several associated attributes:

SCOPE The scope of a command prefix can be either SYSPLEX or SYSTEM:

- **SYSPLEX** The prefixed command is routed to the system on which the prefix is defined.
- **SYSTEM** The prefixed command is executed on the system on which the command is entered.

REMOVE Specifies whether the command prefix is removed from the command text prior to being executed on the receiving system:

NO Indicates that both the command prefix and the command are presented to the receiving system.

YES Indicates that the command prefix is to be removed from the command before it is presented to the receiving system.

FAILDSP Specifies the failure disposition of the prefix being defined:

PURGE The command prefix is automatically deleted when the receiving system is removed from the sysplex, or the defining address space terminates.

SYSPURGE The command prefix is automatically deleted when the receiving system is removed from the sysplex, but not when the defining address space terminates.

RETAIN The command prefix persists even if the receiving system is removed or the defining address space terminates.

The installation command exits and the active subsystems are invoked only on the target system and the command is logged on the target system. The command response is sent back to the originating console.

Note: If an operator command has multiple registered command prefixes, the command is transported only once using the first prefix. Once the command is at the target system, it invokes command processing. No attempt is made to transport it again using the other prefixes, even if the first prefix is removed.

2.7.3 MVS ROUTE Command

The MVS ROUTE command explicitly routes another operator command for execution on another system in a sysplex. It can be issued from both MCS and extended MCS consoles. The response to the command is returned to the issuing console (unless redirected by an L=) operand.

The syntax of the ROUTE command is:

```
RO {sysname,text                                                  }
   {[T=nnn,]{*ALL,text                       }[,L={a     }]   }
   {        {sysgrpname,text                 }    {cc    }    }
   {        {*OTHER,text                     }    {cca   }    }
   {        {(sysname[,sysgrpname...]),text  }    {name  }    }
   {                                              {name-a}    }
```

Figure 10. MVS ROUTE Command Syntax

Where:

sysname The target system name that receives and processes the command. The command response is returned to the issuing console unless redirected by the L= parameter on the routed command or by a MSGRT command.

text The command and specific operands of the command being routed.

T = Specifies an optional timeout interval. T= is valid with *ALL, *OTHER, sysgrpname, or a list of system names or sysgrpnames. The T= value indicates the maximum number of seconds MVS waits for responses from each system before aggregating the responses.

***ALL** Specifies that the command is to be routed to all systems in the sysplex.

***OTHER** Specifies that the command is to be routed to all systems in a sysplex except the system on which the command is entered.

sysgrpname Specifies that the command will be routed to a subset of systems in the sysplex. System group names are defined by the installation. IBM provides an IEEGSYS member in the SYS1.SAMPLIB to define installation named system groups. See 2.8, “Defining System Groups” on page 46 to create the console system groups.

L = Specifies the display area and console where the system is to display the command responses.

Note: Regardless of the CMDSYS value in effect for the console that issues the ROUTE command, the ROUTE command itself is processed on the system on which it was issued.

The ROUTE command is processed in two stages; the ROUTE command is processed first on the issuing system causing the routed command to be transported to the receiving system, where the routed command is processed as if locally issued. If the routed command is prefixed, the prefix may request rerouting of the command before it can be processed.

The installation command exit and the active subsystems on the issuing system see only the ROUTE command and its operands. The ROUTE command is logged on the issuing system. The routed command is processed on the target system as if it had been issued on the that system. The command response is sent back to the originating console.

Note: Directly nested ROUTE commands are not supported. However, if an operator enters a ROUTE command where the command text is a REMOVE=YES prefixed ROUTE command, which has a REMOVE=YES prefixed ROUTE command as the text and so on, the routing is performed for each ROUTE command and for each command prefix. Once the command is at the target system, it invokes command processing. An example:

```
D XCF
 SC47      IXC334I  14.39.08  DISPLAY XCF 798
 SC47         SYSPLEX WTSCPLX1:   SC42              SC43              SC47
 SC47                             SC49              SC50              SC52
 SC47                             SC53              SC54              SC55
D OPDATA
 SC47      IEE603I 14.47.53 OPDATA DISPLAY 802
 SC47       PREFIX     OWNER      SYSTEM     SCOPE     REMOVE   FAILDSP
 SC47       SC42       IEECMDPF   SC42       SYSPLEX   YES      SYSPURGE
 SC47       SC43       IEECMDPF   SC43       SYSPLEX   YES      SYSPURGE
 SC47       SC47       IEECMDPF   SC47       SYSPLEX   YES      SYSPURGE
 SC47       SC49       IEECMDPF   SC49       SYSPLEX   YES      SYSPURGE
 SC47       SC50       IEECMDPF   SC50       SYSPLEX   YES      SYSPURGE
 SC47       SC52       IEECMDPF   SC52       SYSPLEX   YES      SYSPURGE
 SC47       SC53       IEECMDPF   SC53       SYSPLEX   YES      SYSPURGE
 SC47       SC54       IEECMDPF   SC54       SYSPLEX   YES      SYSPURGE
 SC47       SC55       IEECMDPF   SC55       SYSPLEX   YES      SYSPURGE
```

```
RO SC42,SC43RO SC47,SC49RO SC50,SC52RO SC53,SC54RO SC55,D T
 SC55  IEE136I LOCAL: TIME=14.41.00 DATE=1995.291 GMT: TIME=18.41.00 DATE=1995.291
```

2.7.3.1 Command Response Aggregates for the ROUTE Command

When a ROUTE command sends a command to more than one system (using *ALL, *OTHER, or sysgrpname), the command responses returned to the originating console could be very confusing if they were simply presented at the console in the order they were received. ROUTE command processing solves this problem by collecting the messages into an *aggregated response* and presenting the aggregate as one multi-line message. The aggregated messages are sorted in alphabetical order by system name.

If some messages arrive too late to be aggregated, MCS first displays the names of the systems from which messages have not yet arrived, then displays the aggregated messages.

As a default, MCS waits as long as 30 seconds before displaying aggregated messages. However, MCS does not always make the operator wait the maximum time. MCS displays the aggregated messages a short time after receiving at least one response from each system to which the command was routed.

You can change the maximum aggregate wait time by:

- Specifying a new ROUTTIME parameter on the INIT statement in CONSOLxx. This affects the entire sysplex.
- Changing the current ROUTTIME value by entering the CONTROL M command. This affects the entire sysplex.
- Entering the T= operand on the ROUTE command itself. This affects only the ROUTE command on which it is specified.

Note: If a command processor does not use the console ID of the command issuer when issuing a command response or the aggregate wait time is zero, an aggregated command response cannot be returned.

MCS uses extended MCS consoles to process ROUTE commands sent to more than one system and to build command response aggregates. These extended consoles have a common key MVSROUTE, and the console names are *ROUTnnn, as shown in the following example:

```
D C,KEY=MVSROUTE
 IEE892I 12.18.37 CONSOLE DISPLAY 486
 MSG: CURR=5    LIM=1500  RPLY: CURR=4    LIM=999  KEY=MVSROUTE
   NAME       NAME       NAME       NAME       NAME       NAME       NAME
 *ROUT085  *ROUT086  *ROUT087  *ROUT088  *ROUT089  *ROUT090  *ROUT091
 *ROUT092  *ROUT093
```

2.7.3.2 MVS ROUTE Command and System Symbols

If a command is a ROUTE command and, it contains system symbols, on the system where the ROUTE command is issued, the command is substituted up through the specification of the destination system or systems.

The command is then routed with its text and substitution occurs on the receiving system.

System symbols cannot be used in the ″L = ″keyword when the command response is to be aggregated (that is, when the command is being sent to more

than one system and T is greater than zero). In this case, the keyword value is not substituted.

If the destination is a list of systems, and T is not zero, the list is not substituted.

2.7.4 CONTROL Command L= Operand

Using the L= operand on certain MVS commands, like CONTROL or DISPLAY, allows you to specify a target console name on any system in the sysplex. For example, an operator can enter the CONTROL command with L= on one console to change the console characteristics of another console on a different system.

The CONTROL command with the L= operand affects another console on the same system or another system. The command processing on the issuing system is follows:

- The command processing exits and the subsystems of the issuing system get control for the command.
- The hardcopy log processing records the command.
- The command processor for the command on the issuing system is invoked. When the target console is on a different system, the command is routed to that system.

On the target system the command processing continues as follows:

- The command processing exits and the subsystem interface processing is bypassed.
- The hardcopy log processing records the command.
- The command processor on the system processes the command.

2.8 Defining System Groups

IBM provides an IEEGSYS member in the SYS1.SAMPLIB to define installation named system groups. The IEEGSYS program builds groups only on a single system, and must be executed on every system to which the group definitions apply. The system group name can be specified on the MVS ROUTE command, as shown in 2.7.3, “MVS ROUTE Command” on page 43. To create the system groups:

- Create a SYS1.PARMLIB member that defines the group names and the systems in each group, as shown in Figure 11.

```
GROUP(P101) NAMES(SC52,SC53,SC49)
GROUP(P201) NAMES(SC54,SC55,SC50)
GROUP(P301) NAMES(SC42,SC43)
GROUP(CMOS) NAMES(SC52,SC53,SC49,SC54,SC55,SC50,SC42,SC43)
GROUP(CMOS2) NAMES(SC42,SC43)
GROUP(P942) NAMES(SC47)
GROUP(JES3) NAMES(SC49,SC50,SC43)
GROUP(JES2) NAMES(SC47,SC52,SC53,SC54,SC55,SC42)
GROUP(AOC) NAMES(SC47,SC49,SC55)
```

Figure 11. Sample Group Name Specification for IEEGSYS Program

- Create a SYS1.PROCLIB procedure.

```
//IEEGSYS  PROC MEMBER=GSYS00
//IEEGSYS  EXEC PGM=IEEGSYS
//SYSIN    DD   DSN=SYS1.PARMLIB(&MEMBER),DISP=SHR
```

Figure 12. Sample JCL for IEEGSYS Procedure

- Issue the following ROUTE command to each system in the sysplex to create the system groups on each system in the sysplex:

  ROUTE *ALL,START IEEGSYS,MEMBER=GSYS00

  Or, you can place the START command in the COMMNDxx parmlib member.

2.8.1 Command to Display System Groups

See Appendix D, “Display System Groups Command Exit” on page 187 for an example program on how to display the system groups defined by the IEEGSYS program in SYS1.SAMPLIB. The command created for this program is GSYS. In a sysplex, a command can be routed from one system to another. You can use a .CMD statement in a MPFLSTxx member to specify an MVS command exit routine for the GSYS command as follows:

```
.CMD USEREXIT(CMDGSYS)
```

The command installation exit specified, CMDGSYS, receives control when an MVS command is issued. Each installation exit is invoked in the order specified on this .CMD statement.

Figure 13 displays the system groups defined in the SYS1.PARMLIB member, as shown in Figure 11 on page 46.

```
GSYS
 UGSYS01I DEFINED SYSTEM GROUPS
 GROUP P101     - SYSTEM(S):
       SC52     SC53     SC49
 GROUP P201     - SYSTEM(S):
       SC54     SC55     SC50
 GROUP P301     - SYSTEM(S):
       SC42     SC43
 GROUP CMOS     - SYSTEM(S):
       SC52     SC53     SC49     SC54     SC55
       SC50     SC42     SC43
 GROUP CMOS2    - SYSTEM(S):
       SC42     SC43
 GROUP P942     - SYSTEM(S):
       SC47
 GROUP JES3     - SYSTEM(S):
       SC49     SC50     SC43
 GROUP JES2     - SYSTEM(S):
       SC47     SC52     SC53     SC54     SC55
       SC42
 GROUP AOC      - SYSTEM(S):
       SC47     SC49     SC55
 END OF DEFINED SYSTEM GROUPS
```

Figure 13. Display Command for System Groups

This command provides the operator a way to display the installation defined system groups and the system names in each group. These system groups can

be used with the ROUTE command when using the ROUTE command to send a command to a subset of the systems in the sysplex.

```
RO P101,D T
 IEE421I RO P101,D T 503
 SYSNAME  RESPONSES ----------------------------------------------
 SC49     IEE136I LOCAL: TIME=14.36.33 DATE=1996.047  GMT:
         TIME=19.36.33 DATE=1996.047
 SC52     IEE136I LOCAL: TIME=14.36.33 DATE=1996.047  GMT:
         TIME=19.36.33 DATE=1996.047
 SC53     IEE136I LOCAL: TIME=14.36.33 DATE=1996.047  GMT:
         TIME=19.36.33 DATE=1996.047
```

The D T command is routed to the systems defined by group P101.

2.8.2 Installing Command Exits

You can specify command exits that enable you to:

- Modify the text of commands
- Suppress commands
- Change the MCS authority of commands

These command exits are given control every time a command is issued. To activate a command exit:

- Compile and link-edit the command exit in a LINKLST library.
- Add the following .CMD statement to an MPFLSTxx PARMLIB member:

  ```
  .CMD USEREXIT(CMDGSYS)
  ```

- Issue the MVS MODIFY LLA,REFRESH command.
- Issue the MVS SET MPF=xx command.

If you enter the MVS DISPLAY MPF,CMD command, you see the command exits that are currently active.

```
D MPF,CMD
 COMMAND-USEREXIT    COMMAND-USEREXIT    COMMAND-USEREXIT       MPF=XX
     CMDGSYS
```

2.9 Wildcards in Commands

Wildcards are characters that indicate a command applies to all resources whose names match a specified character string. You can also use the asterisk wildcard when specifying job names and identifiers in the following commands:

- D A, D PROG, D R, D SSI, and D XCF
- MODIFY
- SLIP

The wildcards are:

***** The system matches 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.

? The system matches one character. One or more question marks can start the string, end it, appear in the middle, or appear in several places in the string.

Note: You can mix wildcards in any combination.

See *MVS/ESA System Commands* for the rules and examples of wildcards in operator commands.

Chapter 3. MCS Sysplex Consoles

Enhancements to MCS support for the sysplex environment began in MVS/ESA Version 4. Figure 14 depicts the MCS sysplex environment and its features.

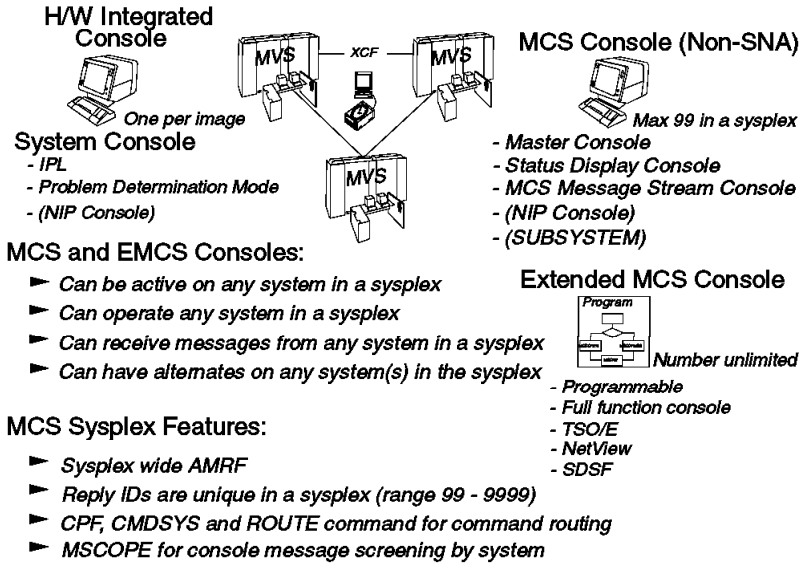


Figure 14. MCS Sysplex Environment

Figure 14 shows the three types of consoles:

MCS MCS consoles (locally attached and SUBSYSTEM) - up to 99 defined through the CONSOLxx parmlib member. See 3.2, “Defining the CONSOLxx Member” on page 55 and 3.3, “CONSOLxx Definitions in a Sysplex” on page 81.

EMCS Extended MCS (EMCS) consoles - programmable consoles defined and activated through an intended programming interface (the MCSOPER macro with REQUEST=ACTIVATE). The number of active EMCS consoles is not restricted. See Chapter 4, “Extended MCS Consoles” on page 97.

System Console integration is supported by MVS and allows the hardware system console to support both hardware functions and the operating system IPL, recovery, and problem analysis.

The system console in problem determination mode is automatically defined by MVS during system initialization. See 3.4, “System Console” on page 87.

In an MCS sysplex, a console can be active on any system in a sysplex and can provide sysplex-wide control. MCS uses XCF services for command and message transportation between systems and thus provides a single system image for the operators. MCS multisystem support features:

- Sysplex-wide action message retention facility (ARMF)

- Sysplex-wide unique reply IDs
- Sysplex-wide command routing through:
  - ROUTE operator command
  - Command prefix facility (CPF)
  - CMDSYS setting for a console (through CONSOLE statement in the CONSOLxx parmlib member or CONTROL V,CMDSYS= operator command)

Normal message presentation is controlled by a console's ROUTCDE and MSCOPE settings, and the UD (undeliverable message) attribute. Synchronous messages are routed according to the SYNCHDEST parameter setting on the DEFAULT statement in the CONSOLxx member. The SYNCHDEST parameter defines the alternate console group whose members can receive a synchronous message. See 3.2.2.3, "Defining Consoles for Synchronous Messages" on page 72.

Message presentation features on MCS consoles include:

- Messages are displayed in either ROLL mode or WRAP mode on locally attached MCS consoles.
- The HOLDMODE specification on the DEFAULT statement of the CONSOLxx member allows an operator to temporarily suspend or hold screen updates when in roll, roll-deletable, or wrap mode.

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

The console reliability, availability, and serviceability (RAS) features include alternate console association. A console may be associated with an alternate through a console group or an alternate console. When a failing console is switched to an alternate (or an operator enters the SWITCH command), MCS merges console attributes with those of the alternate. The switched attributes are added to those of the alternate console and do not replace the existing attributes. Thus, the command authority, message scope, and UD status of the alternate console are not permanently affected by the addition of the failing console's attributes. Once the failed console is switched back, its attributes are removed from the alternate console.

3.1 Defining MCS Consoles

Under normal conditions, operators should use MCS consoles to perform system operations activities in addition to console automation through programs like NetView. Even though you can use the system console to support MVS console functions on processor complex models with console integration, you should not plan to use the system console as a primary operator console.

You should also share your *input/output definition file* (IODF) between the sysplex systems. The MVS IODF, output data set from the HCD process, includes console definitions for the nucleus initialization program. During an MVS IPL, NIP uses the first available NIP console to display messages. If no NIP consoles are available or defined, the system console is used for NIP message display. The NIP consoles are mostly defined to be the same devices as are used for the MCS consoles. See *MVS/ESA Hardware Configuration Definition: User* for details on how to define NIP consoles.

The CONSOLxx parmlib member provides centralized definitions for the console MCS configuration. Each system in the sysplex processes CONSOLxx members at initialization. Therefore in a sysplex, a separate CONSOLxx member could be defined for each system; however, it is not required and not even recommended.

3.1.1 CONSOLxx Parmlib Member

MVS multiple console support allows up to 99 consoles, including subsystem allocatable consoles, to be defined in a CONSOLxx parmlib member for the entire MVS sysplex. Note that extended MCS consoles are not counted towards the sysplex 99 MCS consoles limit.

Figure 15 summarizes the console related parmlib members, their "chaining," and also lists the various statements in each member.

Note: You must be at the MVS/ESA SP Version 5.2 level for the LOADxx to point to the IEASYMxx. See 3.1.2, "LOADxx Parmlib Member" on page 54.

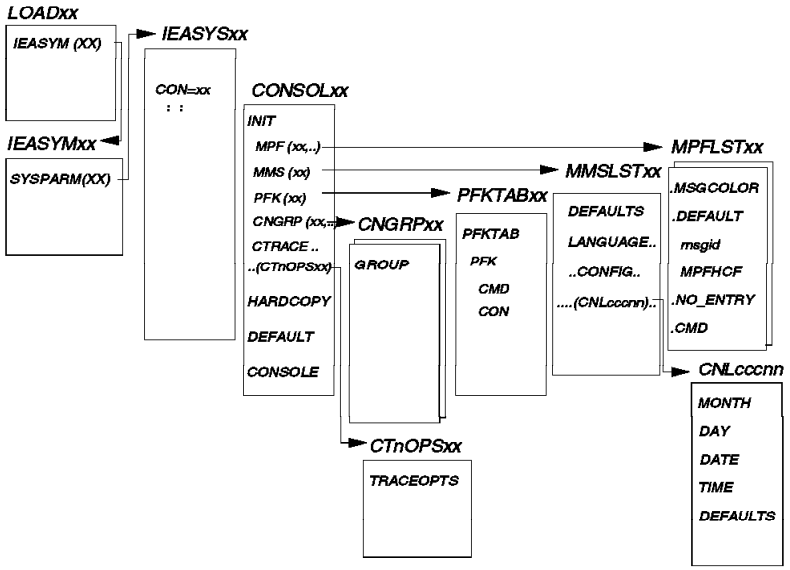


Figure 15. Console Related Parmlib Members

Besides the CONSOLxx parmlib member, several other related parmlib members affect console operations and are shown in Figure 15. Within the CONSOLxx member, you may reference:

CNGRPxx The CNGRPxx parmlib member defines console groups for alternate console switching. See 3.3.2, "Defining Console Groups" on page 82.

CTnOPSxx The CTnOPSxx parmlib member specifies the component tracing options for the operations services (OPS) component.

MMSLSTxx The MMSLSTxx parmlib member is used to display translated U.S. English messages into another language that your installation has provided.

The MMSLSTxx parmlib references the message configuration member CNLcccxx parmlib member which defines how translated

messages are to be displayed at your installation. In a message configuration member, you specify the time and date format for translated messages using the MONTH, DAY, DATE, and TIME statements.

The primary objective of the MVS Message Service is to provide a programmable interface for message translation. The MVS Message Service allows components of, or products running on, MVS to translate U.S. English messages into the user's native language.

Note: MCS messages displayed on locally attached MCS consoles are not translated. When the *MVS message services* (MMS) is active, the TSO/E CONSOLE command attempts to translate console messages to the primary language specified in the user's profile (UPT).

MPFLSTxx The MPFLSTxx parmlib member is used for message processing control.

PFKTABxx The PFKTABxx parmlib member is used to define any PFK tables for the MCS consoles.

CNLcccxx The CNLcccxx parmlib member is used to define how translated messages are to be displayed.

3.1.2 LOADxx Parmlib Member

In order to use a different SYSNAME in IEASYSxx before MVS/ESA 5.2, multiple LOADxx members had to exist in SYSx.IPLPARM or SYS1.PARMLIB to IPL multiple images that pointed to different IEASYSxx members via the SYSPARM statement. With MVS/ESA 5.2, it is now possible to specify one LOADxx member pointing to a new parmlib member (IEASYMxx), as shown in Figure 16.

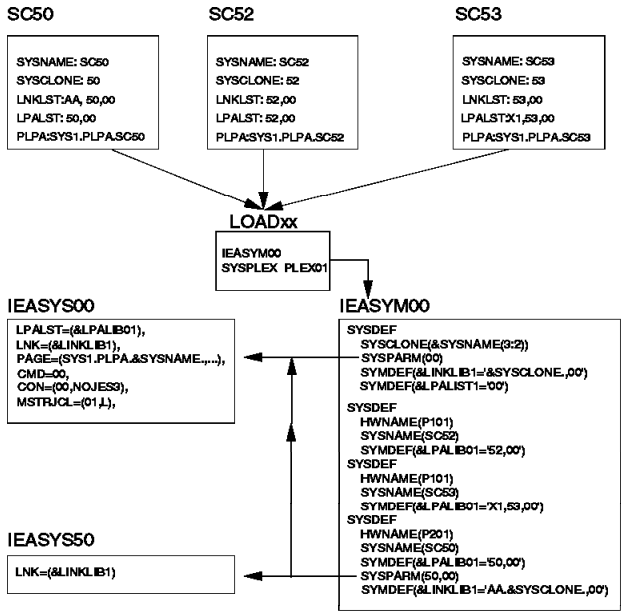


Figure 16. Single LOADxx Parmlib Member for a Sysplex

In Figure 16, three systems are shown pointing to one common LOADxx member in SYSx.IPLPARM or SYS1.PARMLIB. This LOADxx member then points to one IEASYMxx member in SYS1.PARMLIB.

3.1.3 IEASYMxx Parmlib Member

In the IEASYMxx member, four SYSDEF statements are shown. The first SYSDEF statement is a global SYSDEF statement, as no HWNAME, LPARNAME or VMUSERID is specified. The next three SYSDEF statements apply to the three systems shown. As there is a SYSPARM statement in the global SYSDEF stating the default IEASYSxx member to use (IEASYS00), this is the IEASYSxx member that system SC52 and SC53 will use. The local SYSDEF statement for SC50 specifies a SYSPARM statement, and therefore, system SC50 uses IEASYS50 and IEASYS00.

3.1.4 IEASYSxx Parmlib Member

The CON parameter, shown in Figure 15 on page 53, in the IEASYSxx member specifies which CONSOLxx member is selected for the system initialization. The selected CONSOLxx member contains the initialization values for communication tasks, the characteristics for the hardcopy log, and default routing codes for messages that do not have routing information. The CON parameter is also used to allow a system with both JES2 and any level of JES3 installed to run with JES2. This allows function that is incompatible with JES3 to operate successfully.

If CON=NONE is specified, the system is initialized with a default CONSOLxx member.

3.2 Defining the CONSOLxx Member

The CONSOLxx member contains four optional statement types:

- CONSOLE statement
- DEFAULT statement
- HARDCOPY statement
- INIT statement

Appendix E, “CONSOL00 Parmlib Member” on page 191 contains the CONSOLxx definitions for the sysplex configuration shown in Figure 9 on page 41.

3.2.1 CONSOLE Statement

The CONSOLE statement is optional for systems running on ES/9000 and S/390 9672 Parallel Enterprise Server processors. If not defined, the system console is used by MCS (and by nucleus initialization program (NIP)).

The CONSOLE statement defines MCS consoles and subsystem allocatable consoles (SUBSYS). It may also be used to assign attributes to the system console. Each MCS console and subsystem allocatable console must be defined on the CONSOLE statement.

A CONSOLxx member can include multiple CONSOLE statements; one for each console in the configuration, but you cannot have multiple statements specifying the same console name or with the same DEVNUM.

3.2.1.1 MCS Console IDs

MCS consoles are assigned both 4-byte and 1-byte console identifiers. When commands are entered through a console, the console ID is passed to the command processors. The command processor uses the console ID to route the command responses back to the issuing console.

3.2.1.2 EMCS consoles

EMCS consoles have a 4-byte console ID. When an EMCS console is activated, a 1-byte migration ID may be requested by the user. This is necessary for existing command processors, for example JES3 releases prior to JES3 5.2.1, who do not recognize a 4-byte console ID. Therefore, it is necessary for the EMCS console to have a user requested 1-byte migration ID that an old command processor uses to return command responses.

3.2.1.3 Key CONSOLE Statement Parameters

You use the following parameters on the CONSOLE statement to define the device number, console name, and unit type for an MCS console:

DEVNUM DEVNUM corresponds to the MVS device number of the console.

Note: During system initialization, MCS pins all devices (UCBs) defined in the CONSOLxx parmlib member. The pinned UCBs cannot be involved in dynamic configuration changes. If the sysplex systems share the same IODF, you can activate a new I/O configuration dynamically. The new configuration, however, cannot make changes to the console devices because of the outstanding pins. Note that MCS does not unpin console UCBs even if they are made inactive as consoles. If you have to change the devices belonging to your console configuration, in most cases, you should be able to re-IPL each MVS system in the sysplex one at the time without bringing down the entire sysplex using your new IODF and updated CONSOLxx definitions. See *MVS/ESA Hardware Configuration Definition: Planning* for more information with respect to dynamic configuration changes.

Note: Subsystem-allocatable consoles DEVNUM(SUBSYSTEM) are not associated with MVS devices and do not require any IODF definitions.

NAME Name assigns a *unique* name to the console.

The console name is used to identify a given console in system commands and system messages.

If you do not explicitly name a console, the system generates a name by using the EBCDIC representation of the console identifier. This identifier is valid for the duration of the IPL, and may change on subsequent IPLs. The console name generated by the system is used in system messages to identify the console, but you cannot use that name in commands.

Note: You are strongly encouraged to name your consoles, especially subsystem allocatable consoles. If you do not use console names, MCS assigns the next available console ID for the console whenever it rejoins the sysplex. For example, an unnamed console could be assigned a console ID of 05 in a sysplex with four already active consoles (IDs 01, 02, 03, and 04). If the system leaves the sysplex and rejoins later, MCS does not reassign the console ID 05 to

the console, but instead assigns the next available ID 06. Even if you have only five consoles in the sysplex, MCS assigns the next available ID. Sooner or later, MCS will have used all available 99 console IDs, no new nameless consoles can be defined, and you have to bring down the entire sysplex to make new console IDs available.

With named consoles, MCS assigns the first available console ID to a new console when it is defined to MCS for the first time. Once the named console is redefined during a subsequent IPL, the same console ID is always associated with the console as long as the entire sysplex has not been IPLed.

Console names can be chosen freely. You could use names that indicate a specific function for the consoles. For example, you could define a console name of TAPE for a console that receives messages about tape mounts.

Operators and system programmers can use console names instead of console IDs on MVS commands and macros. You also identify consoles by names in alternate console group definitions.

The following names are reserved for MCS and should not be used as installation console names:

- HC
- INSTREAM
- INTERNAL
- OPERLOG
- SYSIOSRS
- SYSJ3xxx
- SYSLOG
- UNKNOWN

Note: You can use a single CONSOLxx member and have unique console names on each system by using system symbols. See 2.5.1, “System Symbols in Operator Commands” on page 37 and 3.3.4, “Shared CONSOLxx Member with System Symbols” on page 86.

UNIT Specifies the type of device to be used as the MCS console.

If you make mistakes when defining CONSOLE statements, MCS issues an informational message, ignores the erroneous statement, and continues processing the valid statements in the CONSOLxx parmlib member.

3.2.1.4 Usability Enhancements

MVS/ESA SP Version 4 introduced an option to the message deletion mode of a console called WRAP mode. The WRAP mode display allows new messages to overlay the oldest messages on the console screen. A separator line appears between the new and the old messages when messages are being overlaid. WTOR and action messages are also overlaid if WRAP mode is in effect.

You can activate WRAP mode by specifying DEL(W) on the CONSOLE statement of the CONSOLxx parmlib member.

```
CONSOLE  DEL(W)
```

If this option is not specified in parmlib, activate WRAP mode with the following command:

```
CONTROL S,DEL=W
```

Note: You might want to change the RTIME value, which is the interval for message roll, to 1/4 second. If so, enter:

```
CONTROL S,DEL=W,RTIME=1/4
```

Figure 17 shows a sample wrap mode display.

Note the IEE163I message at the bottom of the screen. Also note in Figure 17 the separator line and the number 20. The separator line marks the place on the screen where the next message is displayed. That is, the message immediately above the line is the newest message displayed and the one immediately below is the oldest one. The number 20 means that there are an additional 20 message lines to be displayed.

```
        ALLOCAS  ALLOCAS           NSW  *   A=000A   PER=NO   SMC=000
                                               PGN=001  DMN=001  AFF=NONE
                                               CT=000.012S  ET=25.10.46
        SMF      SMF      IEFPROC  NSW  *   A=000B   PER=NO   SMC=000
                                               PGN=000  DMN=000  AFF=NONE
                                               CT=001.022S  ET=25.10.30
        LLA      LLA      LLA      NSW  S   A=000C   PER=NO   SMC=000
                                               PGN=001  DMN=001  AFF=NONE
                                               CT=003.291S  ET=25.10.25
        RMF      RMF      IEFPROC  NSW  S   A=000E   PER=NO   SMC=000
                                               PGN=001  DMN=001  AFF=NONE
                                               CT=063.910S  ET=25.09.28
        VLF      VLF      VLF      NSW  S   A=000F   PER=NO   SMC=000
                                               PGN=001  DMN=001  AFF=NONE
                                               CT=003.276S  ET=25.10.25
        VTMLCL   VTMLCL   VTMLCL   NSW  S   A=0010   PER=NO   SMC=000
                                               PGN=001  DMN=001  AFF=NONE
  20 ------------------------------------------------------------------
                                               CT=078.828S  ET=25.10.30
        GRS      GRS              NSW  *   A=0006   PER=NO   SMC=000
                                               PGN=000  DMN=000  AFF=NONE
                                               CT=000.206S  ET=25.10.46
        DUMPSRV  DUMPSRV  DUMPSRV  NSW  *   A=0007   PER=NO   SMC=000
                                               PGN=000  DMN=000  AFF=NONE
                                               CT=001.691S  ET=25.10.34
        CONSOLE  CONSOLE           NSW  *   A=0008   PER=NO   SMC=000
                                               PGN=001  DMN=001  AFF=NONE
                                               CT=006.515S  ET=25.10.46
  IEE612I CN=SYS6M500 DEVNUM=5E2  SYS=SYS6     CMDSYS=SYS6


  IEE163I MODE= W
```

Figure 17. Sample Wrap Mode Display

3.2.1.5 Multiple Command Recall

Previously in MVS it was possible to recall the last command entered at an MCS console by pressing PA1. An APAR introduced an option that allows an installation to buffer multiple commands and call them up at will. This support began with MVS/ESA SP 4.2.2 with PTFs providing the support for previous releases. It is possible to set up MCS consoles so that up to 15 commands are stored for each console. If an operator wants the last command entered, he has to press PA1 once. If he wants to recall the next to the last command then he presses PA1 twice, and so on up to the maximum specified by the installation.

The installation defines the maximum number of commands that are to be stored for each MCS console in the RBUF parameter of the CONSOLE statement of the CONSOLxx parmlib member, as follows:

```
CONSOLE  DEVNUM(5E0)
         RBUF(15)
         NAME(C6MASTER)
```

In this example, a command recall buffer large enough to hold the last 15 commands is reserved for the console at device number 5E0.

3.2.1.6 Other CONSOLE Statement Parameters

Besides defining the console device number and the console name, the CONSOLE statement also allows you to define the following for a console:

- Unit type
- Command group
- Operating mode
- Alternate console status/alternate console group status
- Routing codes
- Message deletion mode
- Size and number of out-of-line display areas
- The PFK table name
- Message routing instructions
- Message scope
- Monitor specifications

CONSOLE Statement: CONSOLE starts a CONSOLE statement that defines the characteristics of a console. The full syntax of a CONSOLE statement is:

```
CONSOLE   DEVNUM  {(devnum)   }
                  {(SUBSYSTEM)}
                  {(SYSCONS)  }
            UNIT  {(unittype)}
                  {(PRT)     }
            NAME  (conname)
            AUTH  {(MASTER)           }
                  {(INFO)             }
                  {([SYS][,IO][,CONS])}
                  {(ALL)              }
            USE   {(FC)}
                  {(MS)}
                  {(SD)}
            ALTERNATE  {(devnum) }
                       {(conname)}
            ALTGRP(groupname)
            ROUTCODE  {(ALL)                   }
                      {(NONE)                  }
                      {(nnn[,nnn-nnn][,nnn]...)}
            LEVEL  {(ALL)                       }
                   {([ALL][,NB])                }
                   {([R][,I][,CE][,E][,IN][,NB])}
            CON   {(Y)}
                  {(N)}
            SEG(nn)
            DEL   {(Y) }
                  {(R) }
                  {(RD)}
                  {(N) }
                  {(W) }
            RNUM  {(nn)}
                  {(5) }
            RTME  {(nnn)}
                  {(2)  }
            MFORM  {(M)              }
                   {([J][,S][,T][,X])}
            AREA  {(nn[,nn]...)}
                  {(NONE)      }
            PFKTAB(tablname)
            MSGRT(['inst'][,'inst']...)
            [MONITOR({JOBNAMES[-T]}{,SESS[-T]}{,STATUS})]
            UTME  {(nnn)}
                  {(30) }
            MSCOPE  {([sysname|*][,sysname|*]...)}
                    {(*ALL)                      }
            CMDSYS  {(sysname)}
            UD    {(Y)}
                  {(N)}
            RBUF  {(nn)}
                  {(15)}
            SYSTEM  {(sysname)}
```

Figure 18. CONSOLE Statement in CONSOLxx Member of Parmlib

DEVNUM DEVNUM specifies the device number of the console.

```
DEVNUM   {(devnum)    }
         {(SUBSYSTEM)}
         {(SYSCONS)   }
```

DEVNUM is required and must be the first keyword on the CONSOLE statement:

devnum *devnum* must specify an existing console device number defined through HCD.

SUBSYSTEM SUBSYSTEM indicates that this console is reserved for subsystem use, such as by JES3 prior to JES3 5.2.1 or NetView.

SYSCONS SYSCONS indicates that this console is the system console attached to this processor. For more detail, see 3.4.3, “Defining the System Console” on page 88.

UNIT UNIT specifies the unit type of the console.

```
UNIT   {(3270-X) }
       {(3277-2) }
       {(3278-2) }
       {(3278-2A)}
       {(3278-3) }
       {(3278-4) }
       {(3279-2A)}
       {(3279-2B)}
       {(3279-2C)}
       {(3279-3A)}
       {(3279-3B)}
       {(PRT)    }
```

When you specify 3270-X as the devices, the console device has to meet the following criteria:

- The device supports the 3270 data stream and Read Partition Query (such as a 3471 or 3472 IBM InfoWindow terminal).
- The device is attached to a locally attached non-SNA control unit that supports Read Partition Query.

PRT assigns an output only console to a printer device.

NAME NAME assigns a name to the console that uniquely identifies the console. You should not use the same value specified on the DEVNUM parameter for this console as the *conname*. Also, do not use HC, INSTREAM, INTERNAL, SYSIOSRS, SYSJ3xxx, OPERLOG, SYSLOG, or UNKNOWN for conname; these are reserved for the system or JES3 5.2.1.

It is strongly recommended that you specify a name for the system console in CONSOLxx. Select a unique name for the system console that cannot be confused with a valid device number.

AUTH AUTH specifies the command groups that can be entered from the console.

```
AUTH   {(MASTER)          }
       {(INFO)            }
       {([SYS][,IO][,CONS])}
       {(ALL)             }
```

See 2.4.1, “MCS Consoles Command Authorization Checking” on page 30 for operator commands in each command group.

In a sysplex, the first console with MASTER authority to come active is the sysplex master console. There is only one master console in a sysplex at any one time. To find out which console is the master console, issue a DISPLAY CONSOLES (or D C) command. The master console has COND=M in the output from this command. You can switch the master console to its alternate through the VARY MSTCONS command. The master console cannot be a subsystem console.

To change a console′s authorization, you can use the following command:

```
VARY {conname|[/]devnum},AUTH=
```

USE USE specifies how the display console is used.

```
USE   {(FC) }
      {(MS) }
      {(SD) }
```

FC FC. defines a full-capability console able to enter commands and receive status displays and messages. For display console devices USE(FC) is recommended.

MS MS defines a message stream console. If the console is a printer, specify USE(MS).

SD SD defines a status display console.

You can use the following command to change the operating mode of a console:

```
CONTROL V,USE=
```

ALTERNATE ALTERNATE specifies the device number or the symbolic console name of the alternate console.

```
ALTERNATE  {(devnum)  }
           {(conname) }
```

The parameter remains for compatibility. The preferred way to specify console alternates is through ALTGRP.

You can use the following command to change a console′s alternate:

```
VARY {conname|[/]devnum},CONSOLE,ALTCONS=
```

ALTGRP ALTGRP specifies the name of the alternate group for this console.

```
ALTGRP(groupname)
```

The alternate group lists those consoles that are eligible to back up this console in case of a failure. You define the group using the CNGRPxx parmlib member. See 3.3.2, “Defining Console Groups” on page 82. An alternate group can contain both MCS and extended MCS consoles:

```
GROUP     NAME(group name)
          MEMBERS(console name[,console name,...])
```

In the GROUP definition the NAME(group name) specifies the name of the console group and the MEMBERS(console name [,console

name,...]) specifies an ordered list of console names belonging to the specified group.

Once you have defined a set of CNGRPxx parmlib members, you can activate them with the following command:

```
SET CNGRP=(xx,[xx]...)
```

The command has sysplex scope. The following command removes all active console group definitions from the sysplex.

```
SET CNGRP=NO
```

After your command group definitions are activated, you can use the following command to set an alternate console group for the console:

```
VARY CN{(*|consname|consid),ALTGRP={name|{*NONE*}
```

The *NONE* ALTGRP operand value causes the system to remove alternate group or alternate console definitions for the console.

ROUTCODE ROUTCODE specifies the routing codes assigned to the console. Most messages have one or more routing codes.

```
ROUTCODE  {(ALL)                    }
          {(NONE)                   }
          {(nnn[,nnn-nnn][,nnn]...)}
```

The system uses routing codes, decimal numbers from 1 to 128, to determine which console or consoles should receive a message. You assign routing codes to consoles so that only the appropriate messages are routed to the console. The master console always receives routing codes 1 and 2.

The MVS use of message routing codes does not provide enough granularity to implement sophisticated “functional” consoles. As an example, if you want messages related to a set of devices to be routed to a specific console, you could use MPF message exits to add routing codes to these messages and set the functional console to receive only the messages with the customized routing codes. See Appendix F, “MPF Exit - Add Routing Codes” on page 193 for an example on an MPF exit that adds routing codes to an action message.

To change the routing code assignment to a console, you can use the following command:

```
VARY CN{(*|consname|consid),AROUT=(rtcode...)  |
     DROUT=(rtcode...) | ROUT={ALL|NONE|(rtcode...)}
```

LEVEL LEVEL specifies the message levels for the console.

```
LEVEL   {(ALL)                   }
        {([ALL][,NB])            }
        {([R][,I][,CE][,E][,IN]) }
```

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 some of them correspond to the message levels. You may specify the following message level settings:

ALL Indicates that the console is to receive all messages.

NB This console is to receive no broadcast messages.

R This console is to receive the messages that require an operator reply.

I This console is to receive immediate action messages.

CE This console is to receive critical eventual action messages.

E This console is to receive eventual action messages.

IN This console is to receive informational messages.

You can use the following command to change the level of messages received by a console:

```
CONTROL V,LEVEL
```

CON CON indicates whether the console is to function in conversational or nonconversational mode:

```
CON   {(Y)  }
      {(N)  }
```

Y Y indicates conversational mode; on this console you must verify all messages selected for message deletion with the cursor, or selector pen or through the CONTROL command.

N N indicates nonconversational mode; all messages selected for deletion are automatically deleted.

You can use the following command to change the conversation mode of a console:

```
CONTROL S,CON
```

SEG SEG (nn) specifies the number of lines in the message area that can be deleted when the following command is entered:

```
CONTROL E,SEG
```

You can use the following command to change the number of lines to be deleted by the CONTROL E,SEG command:

```
CONTROL S,SEG
```

DEL DEL specifies the message deletion mode of the console:

```
DEL   {(Y)  }
      {(R)  }
      {(RD) }
      {(W)  }
      {(N)  }
```

Y Y indicates that all messages selected for deletion are deleted when the screen becomes full.

R R indicates roll mode. In roll mode, the system deletes a specified number of messages from the screen when a time interval elapses. Deletion occurs only if the screen is full and messages are waiting to be displayed.

RD RD specifies roll mode except for messages awaiting actions. These messages are displayed at the top of the screen.

W W indicates wrap mode. In wrap mode, after the screen is filled with messages, new messages overlay the old messages beginning at the top of the screen. A separator line appears following the most recent message on the screen. When a

console is in wrap mode, WTORs and action messages are not retained on the screen.

Note: When wrap mode (DEL(W)) is specified, the areas defined by the AREA keyword are created, but are unavailable until the console is put into a non-wrap mode.

N N indicates that no automatic message deletion is in effect. Messages must be deleted manually.

You can use the following command to change the message deletion mode of a console:

```
CONTROL S,DEL
```

RNUM RNUM specifies the maximum number of lines included in one message roll.

```
RNUM (nn)
```

You can use the following command to change the roll amount of a console:

```
CONTROL S,RNUM
```

RTME RTME specifies the number of seconds between message rolls or wraps.

```
RTME (seconds)
```

You can use the following command to change the roll interval of a console:

```
CONTROL S,RTME
```

MFORM MFORM specifies the display format of the messages.

```
MFORM   {(M)               }
        {([J][,S][,T][,X]) }
```

M M indicates that the system is to display the message text only. The message display does not include a time stamp, job ID, or job name information, or the system name. M is the default.

J J specifies that the display is to include the job ID or name.

S S specifies that the display is to include the name of the system originating the message.

T T specifies that the display is to include a time stamp.

X X specifies that the system suppress the job name and system name for JES3 messages issued from the global processor.

You can use the following command to change a console's message display format:

```
CONTROL S,MFORM
```

AREA AREA specifies the size of the out-of-line display area on the display console.

```
AREA    {(nn[,nn]...)}
        {(NONE)      }
```

You may set up multiple out-of line display areas for a console. nn is a decimal number specifying the number of lines in one display area. The first number defines the bottom area of the screen. Subsequent numbers define areas working toward the top of the

screen. The minimum number of lines in an area is 4. The maximum number of areas is 11. The sum of the lines in all of the areas must not exceed the screen size.

The out-of-line display areas are named starting from the bottom of the screen to "A," "B," and so on. The general message area is called "Z." The L= operand on an some MVS command (like DISPLAY, and MONITOR) allows you to direct the command output to a console's out-of-line display area.

You can use the following command to dynamically define or resize an out-of-line display area for a console:

```
CONTROL A,nn...{,L=cc|name}
```

The following command removes all out-of-line display area specifications from a console:

```
CONTROL A,NONE{,L=cc|name}
```

Displays are presented in frames in the out-of-line areas. You display the next frame of an area with the following command:

```
CONTROL D,F,L=
```

You delete a display from an area with the following command:

```
CONTROL E,D,L=
```

PFKTAB PFKTAB specifies the name of the table that defines the PFK definitions to be set for the console.

```
PFKTAB (tablname)
```

The initial PFK definitions are retrieved from the PFKTABxx parmlib member that is identified in the PFK parameter on the INIT statement in CONSOLxx.

The SET PFK= command allows you to dynamically change the PFKTABxx member in use and the following command assigns a PFK table to a console for use by operators:

```
CONTROL N,PFK=nnnnnnnn
```

You use the following command to change the definition of a PFK:

```
CONTROL N,PFK=(nn,CMD='...')
```

This command:

- Assigns one or more commands to a PFK
- Assigns one or more other PFKs to a PFK
- Assigns a PFK table to your console

The PFKTABxx parmlib member is installation created and defines one or more program function key (PFK) tables. The following rules apply to the creation of PFKTABxx:

- You may define multiple PFK tables in a member.
- The first statement type in a member must be PFKTAB TABLE(tablname).
- For each PFK table, define each program function key only once.
- The syntax of the statements in the PFKTABxx member is:

```
PFKTAB TABLE(nnnnnnnn)

[         [{CMD({"cccccc[;cccccc]..."})}]          ]
[PFK(xx)  [{    {'cccccc[;cccccc]...'} }][CON({Y})]]
[         [{                          }][    {N} ]]
[         [{KEY(kk[,kk]...)           }]          ]
```

MSGRT MSGRT allows you to specify the routing instructions to direct routable system displays to a specified message area, console or both.

```
MSGRT(['inst'][,]['inst']...)
```

inst can be one routing instruction or a string of routing instructions. Each instruction must be enclosed in single quotes and must be less than 123 characters. As an example, inst could specify:

```
MSGRT('D=(A,CONSOLES,S),L=M02-Z')
```

As a result the output generated by the following commands you enter on this console appears in the general message area (Z) of the console M02:

```
D A
D C or D CONSOLES
D S or D SLIP
```

The list of DISPLAY commands that can be specified through the D= operand is: A, ASM, C, CONSOLES, D, DUMP, IOS, M, R, S, U, SMF, SMS, GRS, MPF, OMVS, and PFK.

If an operator has not specified any MSGRT instructions for this console, and no values are specified for the MSGRT parameter on the CONSOLE statement, MCS displays messages as follows:

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

You stop a console's message routing with the MSGRT NONE command and resume it with the following command:

```
MSGRT {D=(operand...)|TR=A|K|CF|MN},L=
```

MONITOR MONITOR is an optional parameter that allows you to have the system report on selected events.

```
MONITOR({JOBNAMES[-T]}{,SESS[-T]}{,STATUS})
```

You can specify one or more parameter options for MONITOR.

You use the STOPMN command to stop the continual display of job status, data set status, or time-sharing user session activity and the MONITOR command to resume the ongoing status displays.

UTME UTME specifies the time interval in seconds for updating status displays.

```
UTME (nnn)
```

The following command can be used to change a console's status display update interval:

```
CONTROL T,UTME
```

MSCOPE MSCOPE allows you to specify those systems in the sysplex from which this console is to receive messages not explicitly routed to the console.

```
MSCOPE  {(sysname|*[,sysname]...)}
        {(*ALL)                  }
```

An asterisk (*) indicates the system on which this CONSOLE is active and *ALL indicates all systems in the sysplex.

The following command adds, deletes, or changes a console's scope of systems from which messages are received:

```
VARY CN(cn),AMSCOPE=(sy..)|DMSCOPE=(sy..)|MSCOPE={(*ALL)|(sy..)}
```

CMDSYS CMDSYS names the system in the sysplex where commands entered on this console are to be sent implicitly for processing.

```
CMDSYS  {(sysname) }
        {(*)       }
```

An asterisk (*) indicates that commands entered on this console are to be processed on the system where this console is defined.

The following command changes a console's implicit command routing destination:

```
CONTROL V,CMDSYS
```

UD UD specifies whether this console is to receive undelivered messages (those with descriptor codes 1, 2, 3, 11, or 12 which were not delivered to any other consoles) and WTOR messages.

```
UD    {Y    }
      {(N)  }
```

The following command changes a console's designation to receive undelivered messages.

```
VARY CN(cn),UD={Y|N}
```

RBUF RBUF specifies the number of previously entered commands that can be retrieved on this console by pressing the PA1 key.

```
RBUF (nn)
```

Set RBUF to a value greater than 1 to allow the operator to retrieve multiple commands, without having to retype each command. Commands are retrieved as they were entered on the console. That is, commands entered with delimiters (or "stacked commands") are treated as a single command. Commands entered through PF keys cannot be retrieved, unless the keys are in conversational mode (CON=Y).

SYSTEM SYSTEM specifies the system in the sysplex you want to activate the console.

```
SYSTEM (sysname)
```

The SYSTEM parameter is most useful when you share a CONSOLxx member between systems in a sysplex and you want to control which system activates the console rather than letting it be activated on the system where it is currently attached and "ready."

If you don't share the CONSOLxx member, you should not use the SYSTEM parameter. See 3.3.3.1, "SYSTEM Parameter" on page 84 and 3.3.3.2, "Sample Configuration With Shared CONSOLxx" on page 84.

Note: When the CONSOLxx member is shared for all systems in the sysplex, this parameter can be used to prevent activation on a system where the same device address may be a terminal and not a console.

In case the console device is not ready or the device is attached to another system when the system specified through the SYSTEM parameter is initialized, the console is not activated. When you decide to activate the console, make the console device available on the system where you want the console activated and issue the following command for the device:

```
VARY devnum,CONSOLE
```

Or you may issue a VARY command for the console, as follows:

```
VARY CN(cn),ONLINE[,SYSTEM=sys]
```

Note: A console must have been defined to the system where you attempt the activation. Note also that the SYSTEM= on the VARY CN(cn),ONLINE command is used to transport the VARY command to the system where the console is to be activated rather than modify the SYSTEM parameter.

3.2.2 DEFAULT Statement

The DEFAULT statement is optional and only one can be used in a CONSOLxx member.

The DEFAULT statement allows you to:

- Define the default routing codes for unsolicited WTO and WTOR messages that have no routing codes, no descriptor codes, and no console IDs assigned.
- Define whether operators must log on to the MCS consoles before commands are accepted.
- Specify whether you allow hold mode for consoles. When hold mode is in effect, an operator can press Enter to hold the console screen updates. To release the screen and return to roll, roll deletable, or wrap mode, the operator presses Enter again.
- Specify the maximum value of a reply ID in the sysplex.
- Define a console group to receive synchronous WTO or WTOR messages that the system issues.

3.2.2.1 Usability Enhancements

MVS/ESA SP Version 4 introduced a HOLD mode display capability. HOLD mode allows the operator at an MCS console to "freeze" a message display. Output is stopped by pressing Enter, and resumed by pressing Enter a second time.

To be able to use HOLD mode, add the following parameter to the DEFAULT statement in the CONSOLxx PARMLIB member by specifying:

```
DEFAULT  HOLDMODE(YES)
```

This specifies that HOLD mode is activated for all MCS consoles in the system. There is no way to selectively specify it for individual consoles.

HOLD mode is also triggered by logically pressing Enter. For example, if your command delimiter is a semicolon, and you have a console PFK setting that ends with two semicolons, the last semicolon causes the console to toggle in and out of HOLD mode. If you had PF12 defined as shown below and you were in hold mode, pressing PF12 again would release your console from HOLD mode.

```
CONTROL N,PFK=(12,'D A,L;;')
```

When HOLD mode is in effect at a console and messages are queued for display at that console, there is a red (default on a color screen) blinking message (IEE159E) at the bottom right reminding you that there are messages waiting.

If an MCS console is in HOLD mode and the message buffer becomes full, HOLD mode is released automatically. If an operator "forgets" to release the hold mode, MCS automatically releases the holding console at "80% full WTO buffer shortage." The offending console cannot be held again until the message backlog is processed.

DEFAULT statement: The DEFAULT statement is optional. If you specify DEFAULT, you can have only one DEFAULT statement in a CONSOLxx member. If you do not code the DEFAULT statement, the system assigns routing codes 1 through 16 to messages that have no other routing attributes, and automatically logs on MCS consoles, allowing commands to be issued from those consoles.

DEFAULT starts a DEFAULT statement that defines some communications task default processing options. Syntax for a DEFAULT statement is:

```
DEFAULT    ROUTCODE  {(ALL)                    }
                     {(NONE)                   }
                     {(nnn[,nnn-nnn][,nnn]...)}
           LOGON     {(REQUIRED)}
                     {(OPTIONAL)}
                     {(AUTO)    }
           HOLDMODE  {(YES)}
                     {(NO) }
           RMAX      {(nnnn)}
                     {(99)  }
           SYNCHDEST(groupname)
```

Figure 19. DEFAULT Statement in CONSOLxx Member of Parmlib

ROUTCODE ROUTCODE specifies one or more default routing codes for messages that do not have any routing information. ALL specifies 1 through 128 are to be assigned. NONE specifies that no routing codes are to be assigned. nnn is a decimal number from 1 to 128. You can specify a range of routing codes by coding an ascending range of numbers.

LOGON LOGON lets you define whether operators must issue LOGON and LOGOFF commands on MCS consoles:

(REQUIRED) Specifies that an operator must log on to an MCS console before issuing commands from that console except under the following conditions:

- When issuing commands from the master console before RACF is active.
- When issuing the VARY devnum,MSTCONS command from any console to assign the master console (when there is currently no master console assigned) before RACF is active.

If an operator is not logged on to the console, the system rejects commands issued from that console.

(OPTIONAL) Specifies that the operators can optionally log on to the MCS consoles.

(AUTO) Specifies that consoles are automatically logged on when the consoles are activated. The user ID is the console name.

HOLDMODE HOLDMODE specifies whether you want hold mode for the console when the console is ROLL, ROLL-DELETABLE, HOLD, or WRAP mode. Specifying HOLDMODE YES allows an operator to press Enter to suspend or hold messages on the console screen. An operator can enter and exit HOLD mode by pressing the ENTER key with no input.

RMAX RMAX specifies the maximum value of a reply ID in the sysplex. RMAX also determines the size of the reply ID displayed in the message text. For example, specifying an RMAX value of 9999 results in all messages having a four character reply ID.

Use the following command to dynamically increase the maximum number of reply IDs:

```
CONTROL M,RMAX
```

SYNCHDEST SYNCHDEST indicates that synchronous messages (those messages issued using WTO and WTOR macros with SYNCH=YES) should be sent to the first available console in the group specified. A console is available if it is attached to this member of the sysplex. Synchronous messages cannot be sent to another member of the sysplex.

Consoles in this console group must be MCS consoles or the system console. The following reserved console names have special meanings when used as part of this console group:

SYSCON *SYSCON* when used in a group specified on the SYNCHDEST parameter routes a synchronous message to the system console on the system where the message is issued.

MSTCON *MSTCON* when used in a group specified on the SYNCHDEST parameter routes a synchronous message to the master console if it is attached to the system where the message is issued.

The SYNCHDEST parameter is optional. If you do not specify a SYNCHDEST value, the system routes synchronous messages to the master console. If the master console is not attached to this system, or is unavailable, the system routes synchronous messages to the system console, as the console of last resort.

3.2.2.2 Synchronous Messages

There are situations in which system operations cannot continue until the system operator takes some external action. An example might be an authorized application detecting a critical problem that warrants stopping the entire system to correct. Using the LOADWAIT macro and the WTO macro with the WSPARM and SYNCH=YES parameters stops the system so the operator can correct a problem, if possible. By using LOADWAIT and WTO, you issue a synchronous message to the operator and place the system into a restartable or nonrestartable wait state. See 6.1.4, “WTO and LOADWAIT Macro Services” on page 145 for use of these macros.

The SYNCH=YES option of the WTO(R) request without the WSPARM parameter processes synchronously with the caller. SYNCH=YES indicates that the request is to be processed synchronously. This form of the WTO(R) request is used in error and recovery environments, when normal message processing cannot be used.

A WTO message is sent to a console, where it is held on the screen for up to ten seconds, before control is returned to the caller. A copy of the message is also queued for transcription to the hardcopy log.

A WTOR message is sent to a console, and the reply is obtained, before control is returned to the caller. A copy of the message and reply are queued for transcription to the hardcopy log.

Note: The SYNCH option of the WTO macro and using LOADWAIT is a replacement for disabled console messages (DCCF).

3.2.2.3 Defining Consoles for Synchronous Messages

You define which consoles can receive synchronous messages through the DEFAULT statement SYNCHDEST parameter in the CONSOLxx parmlib member. SYNCHDEST(groupname) parameter indicates that synchronous messages should be sent to the first available console in the named console group specified on the CNGRPxx parmlib member. A CNGRPxx member defines an ordered list of consoles either as candidates for console switch selection in the event of a console failure or for synchronous messages. A console group can include both MCS and extended MCS consoles. A console group for synchronous messages must contain only MCS consoles or the system console. See 3.3.2, “Defining Console Groups” on page 82.

A console in the SYNCHDEST console group is available for receiving synchronous messages if it is locally attached to the system where the message is issued. Synchronous messages cannot be sent to consoles locally connected to other systems in a sysplex.

If an attempt to write the synchronous message to the selected console fails, the next console in the SYNCHDEST console group is tried. As the final resort the system console is used. If the synchronous message cannot be delivered to any console, the message is queued for hardcopy, but not delivered for display.

If the SYNCHDEST console group is not specified or if a system in a sysplex does not have any locally attached MCS consoles, the synchronous messages are directed to the system console. Mostly, operators do not work at the system console, which increases the risk that rare, but important synchronous WTO and especially WTOR messages may not be detected. The recommendation is to

always specify a SYNCHDEST console group and check the SYNCHDEST consoles if you believe there is a serious system problem.

3.2.3 INIT Statement

The INIT statement is optional and only one can be defined in a CONSOLxx member. The INIT statement allows you to:

- Specify the limits for WTL (LOGLIM), WTO (MLIM), and WTOR (RLIM) buffers. MCS keeps some WTO and WTOR messages in buffers in virtual storage. The WTO buffers hold the messages that the system has not yet displayed at the eligible consoles; the WTOR buffers each hold one WTOR message that the system has already displayed but that an operator has not responded to.
- Designate one or more MPFLSTxx members to use with this CONSOLxx member.
- Define the PFKTABxx member to use with this CONSOLxx member.
- Specify the MMSLSTxx member to use with this CONSOLxx member.
- Define the CNGRPxx member to set up console groups. See 3.3.2, “Defining Console Groups” on page 82.
- Specify the CTnOPSxx member that contains component trace options for the operations services (OPS) component.
- Activate the action message retention facility.
- Activate the WTO installation exit IEAVMXIT.
- Specify the MONITOR command to display additional status information in mount messages.
- Define an MVS command delimiter so an operator can issue multiple commands with one enter.
- Specify the default maximum time the ROUTE *ALL, ROUTE *OTHER, or ROUTE *systemgroupname* command waits before aggregating responses.

INIT statement: The INIT statement is optional. If you do code an INIT statement, you can code only one in a CONSOLxx member.

The full syntax for an INIT statement is:

```
INIT       MLIM  {(nnnn)}
                 {(1500)}
           LOGLIM  {(nnnnnn)}
                   {(1000)}
           RLIM  (nnnn)
           AMRF  {(Y)}
                 {(N)}
           UEXIT  {(Y)}
                  {(N)}
           MPF  {(NO)}
                {(xx[,xx,...])}
           MMS  {(NO)}
                {(xx)}
           PFK  {(NONE)}
                {(xx)  }
           CNGRP {(xx[,xx,...])}
                 {(NO)}
           MONITOR([SPACE][,DSNAME])
           CMDDELIM  {(c)    }
                     {(X'hh')}
           NOCCGRP(groupname)
           CTRACE  {(CTnOPSxx)}
                   {(CTIOPS00)}
           ROUTTIME  {(nnn)}
                     {(30)}
```

Figure 20. INIT Statement in CONSOLxx Member of Parmlib

INIT specifies that the communications task initialization values follow:

MLIM MLIM specifies the maximum number of buffers that the system can use to process write-to-operator (WTO) messages. Each buffer is 352 bytes, and is obtained from private storage (subpool 229) in the communications task (COMMTASK) address space.

When messages fill up the buffers, most jobs that issue a WTO or WTOR go into a wait until buffers are available. To avoid WTO message buffer shortages, you can raise your WTO buffer limit (MLIM) and adjust message deletion specifications on your consoles. You can use the following command to change or display the number of allowed WTO or WTOR message buffers:

```
CONTROL M,MLIM
```

If there is a shortage of WTO buffers on a system in the sysplex, you can route the DISPLAY CONSOLES command to that system:

```
D C
 IEE889I 16.54.02 CONSOLE DISPLAY 441
 MSG: CURR=440  LIM=1500 RPLY:CURR=4    LIM=999  SYS=SC49      PFK=00
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS ---------------
  M04                07  COND=A      AUTH=MASTER       NBUF=436  UD=N
   0860                  AREA=Z          MFORM=T,S,J,X
   SC49                  DEL=R    RTME=1/4    RNUM=28   SEG=28    CON=N
                         USE=FC   LEVEL=ALL             PFKTAB=MCONPFK0
                         ROUTCDE=ALL
                         CMDSYS=SC49
                         MSCOPE=SC49,SC50
                         MONITOR=JOBNAMES
                         ALTGRP=MASTER
```

In the command response message IEE889I on the label line the "MSG: CURR=xxxx" shows the number of WTO message buffers in use by the system at this time. If xxxx is greater than 9999, asterisks appear. For each console, the "NBUF=nnnn" tells the number of WTO message buffers currently queued to this console. If nnnn is greater than 9999, nnnn is ****. "N/A" value is shown for consoles not active on the system where the DISPLAY CONSOLES is run. In this example, almost all the available WTO message buffers on system SC49 are in use for messages queued to console M04.

The D C,B command has more information, as shown below:

```
D C,B
 IEE889I 18.18.32 CONSOLE DISPLAY 535
 MSG: CURR=0    LIM=1500 RPLY:CURR=0    LIM=999  SYS=SC50      PFK=00
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS ---------------
 NO CONSOLES MEET SPECIFIED CRITERIA
  WTO BUFFERS IN CONSOLE BACKUP STORAGE =        0
  ADDRESS SPACE WTO BUFFER USAGE
  NO ADDRESS SPACES ARE USING MORE THAN   500 WTO BUFFERS
  MESSAGES BEING HELD FOR OTHER SYSTEMS - WTO BUFFER USAGE
  NO WTO BUFFERS ARE BEING HELD FOR OTHER SYSTEMS
  MESSAGES COMING FROM OTHER SYSTEMS - WTO BUFFER USAGE
  NO WTO BUFFERS ARE IN USE FOR MESSAGES FROM OTHER SYSTEMS
```

If the system has accumulated a large number of WTO messages, do one or more of the following:

- Enter the CONTROL Q command to reroute the messages.
- Enter the CONTROL E or CONTROL S command to delete a specified number of messages or change the console's message deletion mode.
- Enter the CONTROL M,MLIM command to increase the value of the WTO buffer limit.
- Use the VARY CN command to deactivate the offending console.
- Respond to any messages requesting an operator action.

In the worst case, during a WTO buffer shortage, the system processes only WTO messages that might be critical to relieving the buffer shortage, such as responses to the D C,B and D R commands. All other WTO messages are placed into backup storage. After 40,000 messages are in the backup storage, then WTO messages are discarded. Discarded messages are not sent to the console, the hardcopy log, the message exits, the SSI, or the joblog.

LOGLIM LOGLIM specifies the maximum number of buffers that the system can use to process write-to-log (WTL) messages. Each buffer is 140 bytes, and is obtained from the extended common service area (ECSA).

Note: Ensure that there is storage in extended CSA available for the LOGLIM value specified. Otherwise, an error results.

You use the following command to change or display the number of allowed WTL SYSLOG buffers:

```
CONTROL M,LOGLIM
```

Note: When you set LOGLIM=999999, you allocate over 100 megabytes of CSA storage for WTL SYSLOG buffer storage. Be also aware that the following command requests the system log task to free all outstanding WTL buffer storage and results in the potential loss of messages sent to hardcopy:

```
CONTROL LOGLIM=0
```

During a WTL buffer shortage the WTL processing ceases to allocate WTL queue elements. Subsequent WTL requests result in a return code of 4 to the user.

RLIM RLIM specifies the maximum number of buffers that the system can use to process WTOR requests. Each buffer (ORE) is 96 bytes, and is obtained from the extended common service area (ECSA).

During a critical WTOR buffer shortage, the system puts tasks issuing WTOR macros in a wait state (if they did not specify a busy exit return) until the operator either reduces the number of outstanding WTORs or increases the number of WTOR buffers.

Operators should enter a `DISPLAY R,R` command to see the accumulated WTOR messages and do one of the following:

- Reply to 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 that 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.

AMRF AMRF specifies whether the action message retention facility is to be active.

You can use the `CONTROL M,AMRF` command to change or display the status of the action message retention facility.

When there is a severe action message buffer shortage, the system no longer retains eventual action messages. The system issues message IEA360A if console message buffers (WQEs) begin to back up.

Operators can enter the `DISPLAY R` command to display the details of all outstanding immediate action and eventual action messages and delete messages by:

- Responding to messages requesting an action
- Entering the `CONTROL C` command.

If a shortage still exists, you might want to deactivate the AMRF by entering the following command:

```
CONTROL M,AMRF=N
```

UEXIT UEXIT specifies whether the WTO/WTOR installation exit IEAVMXIT is to be active.

You use the following command to change the status of the IEAVMXIT exit:

```
CONTROL M,UEXIT
```

MPF MPF indicates whether the message processing facility is to be active. The MPF processing options are defined in the MPFLSTxx parmlib members. Multiple members may be defined and they are concatenated.

You use the SET MPF command to change the set of active MPFLSTxx members. The DISPLAY MPF command lists the current MPF processing options.

MMS MMS indicates whether the MVS message service is to be active. The MMS processing options are defined in the MMSLSTxx parmlib members.

You use the SET MMS command to change the set of active MMSLSTxx members. The DISPLAY MMS command shows the current MMS configuration.

PFK PFK specifies the PFKTABxx parmlib member that contains PFK tables for your consoles.

You use the SET PFK command to change to a new PFKTABxx member and enter CONTROL N,PFK=nnnnnnnn to assign the PFK table that contains the PFK definitions you want to use for the console. To change a specific PFK definition for a console where the command is entered, enter CONTROL N,PFK=(nn1,CMD='...'). The DISPLAY PFK,CN command shows the current PFK assignment for a console and the DISPLAY PFK,T command lists the PFK definitions available or in a specified table.

CNGRP CNGRP indicates whether the system or sysplex is to activate console group definitions. See 3.3.2.1, “Console Switching and Console Groups” on page 82.

You use the SET CNGRP command to change the set of active CNGRPxx members. The DISPLAY CNGRP command shows the current console group configuration.

MONITOR MONITOR specifies whether the system is to add status information to mount messages displayed on consoles.

CMDDELIM CMDDELIM specifies an MVS command delimiter that the installation chooses. CMDDELIM allows the operator to enter multiple commands on the console at one time. Each command separated by the delimiter is processed separately. Do not use the command delimiter within a quoted string. If you do, the system considers the delimiter as part of the text and it does not process the delimiter. The order in which the system processes the commands may be different from the order in which the operator entered them. PF keys continue to use a semicolon (;) as the delimiter.

NOCCGRP In a system or sysplex, NOCCGRP specifies the console group whose members are eligible to be selected as the master console during a no-consoles condition. Members of this console group cannot be extended MCS consoles and the group must be defined in an active CNGRPxx member.

CTRACE CTRACE specifies the parmlib member that contains the tracing options for the operations services (OPS) component.

For more information about specifying options for the operations services component trace, see *MVS/ESA Diagnosis: Tools and Service Aids*.

ROUTTIME In a sysplex, ROUTTIME specifies the maximum number of seconds the ROUTE *ALL or ROUTE systemgroupname command waits for responses from each system to the command being routed before aggregating responses. The value specified for ROUTTIME in the INIT statement of the first system in the sysplex applies to all systems that join the sysplex, unless you change the value with a CONTROL M command.

The ROUTTIME value can be overridden with the T operand on the ROUTE command.

3.2.4 HARDCOPY Statement

The HARDCOPY statement is optional. If you do not specify HARDCOPY, the system uses SYSLOG as the hardcopy medium. You can have only one HARDCOPY statement in a CONSOLxx member.

The HARDCOPY statement allows you to:

- Specify whether the hardcopy medium active at initialization is an MCS printer, SYSLOG, or OPERLOG.
- Define routing codes for messages to be included in the hardcopy message set.
- Specify the kinds of messages (commands, responses or status display) to be included in the hardcopy message set.
- Define the console group from whose members the system can select an alternate console device as the hardcopy medium.
- Specify whether the system console should receive undelivered action messages and WTOR messages.

HARDCOPY statement: The syntax for a HARDCOPY statement is:

```
HARDCOPY   DEVNUM  {(devnum)         }
                   {(SYSLOG)         }
                   {(OPERLOG)        }
                   {(devnum,OPERLOG)}
                   {(SYSLOG,OPERLOG)}
           ROUTCODE  {(ALL)                    }
                     {(NONE)                   }
                     {(nn[,nnn-nnn][,nnn]...)}
           CMDLEVEL  {(NOCMDS)}
                     {(INCMDS)}
                     {(STCMDS)}
                     {(CMDS)  }
           HCPYGRP(groupname)
           UD  {(Y)}
               {(N)}
```

Figure 21. HARDCOPY Statement in CONSOLxx Member of Parmlib

HARDCOPY identifies the hardcopy medium, defines the hardcopy message set, specifies the name of the alternate console group, or specifies whether UD messages not sent to any other console are to be delivered to the system console, depending on the options specified.

DEVNUM DEVNUM specifies the output device.

```
DEVNUM   {(devnum)           }
         {(SYSLOG)           }
         {(OPERLOG)          }
         {(devnum,OPERLOG)   }
         {(SYSLOG,OPERLOG)   }
```

Note: When you specify both subparameters, they can appear in either order; for example, (OPERLOG,SYSLOG) is equivalent to (SYSLOG,OPERLOG).

devnum Specifies the device number of a printer console that is to be the hardcopy medium.

Note: In a sysplex, it would be very unusual to do logging on a console printer.

SYSLOG Indicates that the system log is to be the hardcopy medium.

OPERLOG Indicates that the operations log is to be activated and to receive the hardcopy message set. You can specify OPERLOG alone or with devnum or SYSLOG. When you specify OPERLOG with devnum or SYSLOG, both OPERLOG and devnum or SYSLOG receive the hardcopy message set.

Note: A display console cannot be the hardcopy medium. The device you specify for the hardcopy medium must be defined as a console in CONSOLxx.

To deactivate a hardcopy medium, use the following command:

```
VARY {devnum|SYSLOG|OPERLOG},HARDCPY,OFF
```

To activate a hardcopy medium, use the following command:

```
VARY {devnum|SYSLOG|OPERLOG},HARDCPY
```

ROUTCODE ROUTCODE specifies the routing codes the system is to use to select messages for the hardcopy message set. At system initialization, processing of the HARDCOPY statement sets up a minimum set of routing codes (1,2,3,4,7,8,10, and 42), in addition to any others specified for the hardcopy message set.

ALL ALL indicates that all routing codes (1-128) are used to select messages for the hardcopy message set.

NONE NONE indicates that no routing codes are used to select messages for the hardcopy message set. It contains the minimum set established at initialization.

(nn[**,nnn-nnn**][**,nnn**]**...)** Specifies routing codes by numbers. *nnn* is a decimal number from 1 to 128. You can specify a range of routing codes by coding an ascending range of numbers.

To change the hardcopy routing codes, issue the following command:

```
VARY {devnum|SYSLOG|OPERLOG},HARDCPY,{ROUT|AROUT|DROUT}
```

Note: *all* active hardcopy media are affected even if the command specifies one.

CMDLEVEL CMDLEVEL specifies the types of commands and command responses that are to be included in the hardcopy message set.

NOCMDS Specifies that the system is not to include operator or system commands or their responses in the hardcopy message set unless they meet other criteria, such as routing codes. The system ignores this option and assumes CMDS, unless all of the following conditions are met:

- Only one console is active on the system.
- The console is a non-display console, such as a printer.
- The system is not part of a sysplex.

INCMDS Specifies that the system is to include operator and system commands and their responses, excluding any status displays, in the hardcopy message set.

STCMDS Specifies that the system is to include operator and system commands, their responses, and static status displays in the hardcopy message set.

CMDS Specifies that the system is to include operator and system commands and their responses, and static and dynamic status displays in the hardcopy message set.

To change the hardcopy command level, issue the following command:

```
VARY {devnum|SYSLOG|OPERLOG},HARDCPY,cmd_level
```

Note: *all* active hardcopy media are affected even if the command specifies one.

HCPYGRP HCPYGRP specifies the name of the alternate console group that MCS uses to select a console device, or SYSLOG, as the hardcopy medium during a hardcopy switch.

If you do not specify groupname, or if the group you specified does not contain any eligible candidates (or if the group name is not valid), the system scans all consoles on this system to find an eligible hardcopy console.

UD UD specifies whether undeliverable messages (those with descriptor codes 1, 2, 3, 11, or 12, and WTOR messages) that are not received by any other console are to be delivered to the system console (for those processor models that support a system console).

N N indicates that the system console receives UD messages not sent to any other console. N is the default.

Y Y indicates that UD messages should be hardcopied. They are not sent to the system console.

When the system console is in PD (problem determination) mode, you can change the UD specification with the VARY CN(),UD command.

3.3 CONSOLxx Definitions in a Sysplex

Before you can set up a sysplex MCS console configuration, you must define the console hardware configuration to each MVS system in the sysplex using the hardware configuration definition (HCD) program. Once this configuration is defined, you can define your CONSOLE statements for the MCS consoles. You are encouraged to share your CONSOLxx member between all sysplex systems, which requires all systems to have all devices defined that are referenced on the CONSOLE statements in the CONSOLxx parmlib member.

Some of the CONSOLxx definitions have sysplex scope and are processed only by the first system entering the sysplex:

RLIM RLIM is specified on the INIT statement. You can use the CONTROL M,RLIM command to change or display the number of allowed WTOR message buffers. Note that you cannot raise the RLIM value past the maximum upper limit set by the current RMAX value. However, you can use the CONTROL M,RMAX command to display or dynamically increase the maximum number of reply IDs for the sysplex.

Note: MCS does not use the MLIM (WTO buffer limit) and RLIM (WTOR buffer limit) parameter values until the hardcopy medium becomes active.

AMRF AMRF is specified on the INIT statement. You can use the CONTROL M,AMRF command to dynamically turn the action message retention facility on or off. The status of the action message retention facility is modified for all systems in the sysplex.

CNGRP CNGRP is specified on the INIT statement. You can use the SET CNGRP command to activate new console group definitions or deactivate existing definitions. The SET CNGRP command has sysplex wide scope.

ROUTTIME ROUTTIME is specified on the INIT statement. You can override the default value with the T= operand on the ROUTE command.

RMAX RMAX is specified on the DEFAULT statement. RMAX specifies the maximum value of a reply id in the sysplex. RMAX also determines the size of the reply id displayed in the message text. You can use the CONTROL M,RMAX command to dynamically increase the maximum number of reply IDs for the sysplex.

The use of system symbols is supported in console definitions. When some systems in the sysplex 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. See 3.3.4, “Shared CONSOLxx Member with System Symbols” on page 86.

3.3.1 Master Console in a Sysplex

The master console in a sysplex is determined by the first system that IPLs and activates a console with AUTH(MASTER) in the CONSOLxx parmlib member. There is only one console that can be the master. This console is the principal way of communicating with the sysplex. Any console with master authority can enter all commands.

3.3.2 Defining Console Groups

Beginning with MVS/ESA SP 4.2, it is possible to define groups to serve as candidates for switch selection in case a console fails. Before this change you could specify only a single MCS console to take over in case a console failed. You can specify the name of a group as the alternate. Each console group can contain both MCS consoles and extended MCS consoles. Up to 38 console groups can be active at any one time. When a console fails, MVS chooses a console from the alternate console group to take over its functions.

3.3.2.1 Console Switching and Console Groups

Console groups are specified in SYS1.PARMLIB with a CNGRPxx member. The CNGRP parameter on the INIT statememt of the CONSOLxx parmlib member indicates whether the system or sysplex is to activate console group definitions. You can use console groups in the following situations:

- In the event of a console failure, the system searches for a switch candidate. You specify the name of the console group on the CONSOLE statement on the CONSOLxx parmlib member.
- In the event of a hard-copy failure, the system searches for a switch candidate. You specify this group on the HARDCOPY statement in the CONSOLxx parmlib member.
- In the event of a no-consoles condition, the system searches for a console eligible to be the master console. You specify this group on the INIT statement in the CONSOLxx parmlib member.
- To specify the order in which consoles are to receive synchronous messages. You specify this group on the DEFAULT statement in the CONSOLxx parmlib member.

3.3.2.2 Console Group Example

The following example shows a console group definition that is defined in SYS1.PARMLIB in a CNGRPxx member:

```
GROUP  NAME(GROUP1)
       MEMBERS(EXTCON1,EXTCON2,MASTER1)
```

This example shows a group named GROUP1 that has been defined to contain two extended MCS consoles, EXTCON1 and EXTCON2, along with an MCS console named MASTER1. A group can be activated either by the SET CNGRP=xx command or by coding the group name in the INIT statement of the CONSOLxx member in SYS1.PARMLIB.

Defining a group as an alternate to a console differs according to the type of console, MCS or extended MCS.

MCS An alternate for an MCS console is defined in the CONSOLxx member of SYS1.PARMLIB in the CONSOLE parameter.

EMCS An alternate for an extended MCS console must be defined using RACF and is included as a subset of the OPERPARM information. See 4.1.3.1, “OPERPARM Segment” on page 101.

Note: You can use the following command to set an alternate console group for either type of console:

```
VARY CN{(*|consname|consid),ALTGRP={name|{*NONE*}
```

The following example shows how you would add an alternate group to an extended MCS console using a RACF command:

```
ALTUSER USER1 OPERPARM(ALTGRP(GROUP1))
```

This example defines GROUP1 as an alternate group for console USER1.

In a sysplex, there can be only one master console. If it fails, the sysplex needs guidance as to which console on which system to use as the new master console.

Using console groups, it is possible to define a set of consoles that may be used as master consoles in the event of a no-consoles condition occurring. These console groups are defined by specifying the console group name on the NOCCGRP subparameter in the CONSOLxx INIT statement. Extended MCS consoles must not be included in these groups.

Systems joining a sysplex take on the console groups currently in force for that sysplex. Any groups defined in the joining system are ignored. The SET CNGRP command sets sysplex-wide console groups and the DISPLAY CNGRP command displays all console groups in the sysplex and which system activated them.

3.3.3 Shared CONSOLxx Member Considerations

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. By having the CONSOLxx member shared by all systems in the sysplex, it is easier to maintain the sysplex console configuration, especially when you also share the production IODF data set.

You have one CONSOLE statement for each named console in your CONSOLxx member. This results in a straightforward one-to-one mapping between console names and device numbers on all sysplex systems.

3.3.3.1 SYSTEM Parameter

You may use the optional SYSTEM parameter on the CONSOLE statement to specify the system in the sysplex where MCS attempts to activate this console. If the SYSTEM parameter is specified and it names the system currently being initialized, MCS activates the console if the device is attached and in ready status. If the SYSTEM parameter names a system other than the one currently being initialized, MCS does 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.

3.3.3.2 Sample Configuration With Shared CONSOLxx

Figure 22 shows an example of an MCS sysplex console configuration defined using a shared CONSOLxx parmlib member.

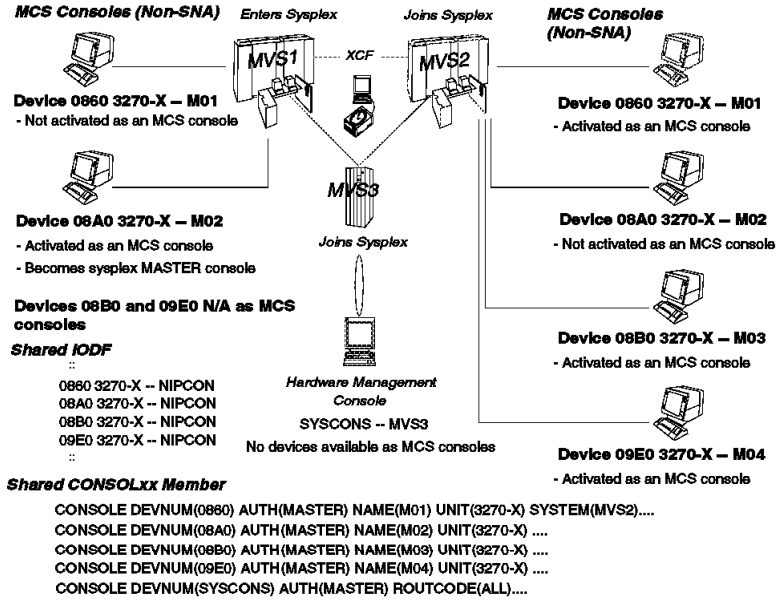


Figure 22. Sample MCS Console Configuration

IPL of MVS1: In the example configuration, shown in Figure 22, the MVS1 system IPLs first and enters the sysplex.

- It reads the CONSOLxx member, validates it, creates internal console descriptors for each defined console, sets the sysplex wide MCS processing options, and attempts to activate as many consoles as possible.
- Consoles M03 (08B0) and M04 (09E0) cannot be activated on the MVS1 system because the required devices are not attached to the system.
- Console M01 (0860) is not activated because the SYSTEM parameter specifies a different system then the one that is initializing.

- The system console is activated with the default name MVS1 (MVS system name).

  Note: An attempt is always made to activate the system console even if it is not defined in the CONSOLxx member.

- Console M02 becomes the sysplex master console as it is the first activated console with MASTER authority.
- An attempt is made to use device 0860 first as the NIP console during the system initialization. If the 0860 console is not attached or powered on, console 08A0 is attempted and so on.
- If none of the NIP consoles are available, the system console is used for system initialization.

IPL of MVS2: When the MVS2 system IPLs and joins the sysplex, it reads the same CONSOLxx parmlib member as did the MVS1 system.

- It validates the definitions and obtains the sysplex wide MCS options from the MVS1 system.
- It also verifies the consistency of the console definitions against the MVS1 definitions, issues informational messages if discrepancies exist, and creates an internal “composite” view of consoles which is also made known to the MVS1 system.
- Finally the MVS2 system attempts to activate the consoles defined to it:
  - Console M01 is activated (the SYSTEM parameter equals to the initializing system name).
  - Console M02 is not activated (it is already active on the MVS1 system).
  - Consoles M03, M04, and the system console MVS2 are activated.

In the example, no new consoles are added to the composite console configuration as all sysplex consoles have already been defined by the MVS1 system.

Note: The first system that defines a console assigns all its attributes.

IPL of MVS3: The MCS initialization process is the same in the MVS3 system as it is for the MVS2 system. A composite view of consoles is created and made known to the other systems in the sysplex.

- As the MVS3 system does not have any locally attached MCS consoles, only the system console MVS3 is activated.
- The system console is also used as the NIP console.

The normal operation should use any of the locally attached MCS consoles active on the MVS1 or MVS2 systems to control the MVS3 system.

3.3.3.3 Device Number Considerations

Having an all symmetric console device configuration tends to use the most device numbers for the number of active consoles. A console with a given name can be active only on one system at any point of time. On the other hand, it also provides maximum flexibility. The console can be switched to and activated on any system in the sysplex. Thus, if you lose a system, you don’t lose a named functional console. Being able to perform the same functional operations from the same set of named consoles helps in operations auditing.

If you want to save device numbers that you assign to consoles, MCS initialization allows different consoles (by name) on different sysplex systems to specify the same device number on the DEVNUM parameter. It is also possible to have the same console specify on the DEVNUM parameter different device numbers on different systems. Because only one CONSOLE statement is allowed for each device number in a CONSOLxx parmlib member, you must use system symbols to set up your shared CONSOLxx parmlib member.

3.3.4 Shared CONSOLxx Member with System Symbols

Figure 23 shows an example of an MCS sysplex console configuration defined using system symbols and a shared IEASYMxx and CONSOLxx parmlib member.

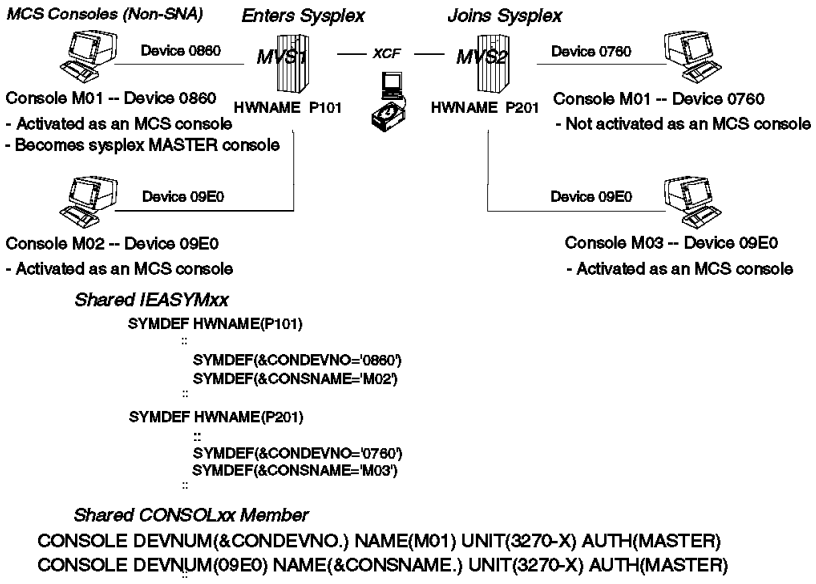


Figure 23. Sample MCS Console Configuration Using System Symbols

Console M01: In Figure 23:

- Console M01 uses device number 0860 when attached to the MVS1 system and device number 0760 on the MVS2 system.
- Console M01 can be activated on either MVS system, but not both.
- The shared IEASYMxx member assigns the value “0860” to the system symbol &CONDEVNO when MVS is initializing on the processor P101 and the value “0760” when MVS is initializing on the processor P201.
- The single CONSOLE statement for console M01 assigns either device number 0860 or 0760 to the console through the &CONDEVNO symbolic, depending on which processor MCS is initializing.

Consoles M02 and M03: In Figure 23:

- Consoles M02 and M03 have the same device number 09E0 on the MVS1 and MVS2 systems.

- The shared IEASYMxx member assigns the value “M02” to the system symbol &CONSNAME when MVS is initializing on the processor P101 and the value “M03” when MVS is initializing on the processor P201.
- The single CONSOLE statement specifying DEVNUM(09E0) parameter assigns either console name M02 or M03 to the console through the &CONSNAME symbolic depending on which processor MCS is initializing.

Note: After both MVS1 and MVS2 systems have initialized, consoles M02 and M03 are known on all sysplex systems, but they are defined only on one system. If you deactivate consoles M02 and M03, you cannot reactivate M03 on the MVS1 system because it is defined to the MVS2 system.

When forming console names using system symbols, you can use any symbol that resolves to a unique and valid console name. For example, the default value (last two characters of the system name) of the &SYSCLONE static system symbol could have been used in the model “MC&SYSCLONE.” to create names like MCS1 and MCS2.

3.3.5 Multiple CONSOLxx Member Considerations

A shared CONSOLxx member does not allow you to tailor your INIT, HARDCOPY, or DEFAULT statement parameters to the needs of an individual system unless you use system symbols. If you want to tailor the system scope parameters on these statements for individual systems and you don′t want to use the IEASYMxx parmlib member, you probably should define unique CONSOLxx members for each sysplex system.

Unique CONSOLxx members for each system in a sysplex provides flexibility for console definitions. You can define consoles based on the needs of each system. However, depending on when a given system joins the sysplex, the lack of consistency, if any, between the CONSOLxx parameters can affect how the console processing is to be defined. At least, you should make sure that the sysplex wide parameters are the same on each of the CONSOLxx members. The final composite sysplex console configuration may also vary if you initialize all of your sysplex systems after a sysplex outage.

3.4 System Console

MVS/ESA SP Version 4.2 introduced console integration that permits the hardware system console of the ES/9000 processors and the HMC of the IBM 9672 processors to be used as an MCS console. The system console is the physical display used to control the hardware functions of the processor. Console integration reduces the impact of a no-MCS console condition.

The system console may be used with console integration in the following situations:

- During MVS system initialization as the nucleus initialization program (NIP) console when other consoles are not available.
- During MVS console recovery scenarios to diagnose an MVS console outage and reestablish a usable MVS console (for example, locally attached MCS, TSO, or NetView) when all MCS consoles have failed.

Note: The system console is not recommended to be used as the only primary operator console in the sysplex as performance can be affected. Under normal

conditions, the operation of the MVS system should be done from MCS or extended MCS consoles.

3.4.1 Problem Determination Mode

MVS/ESA SP Version 4.3 introduced problem determination (PD) mode for the system console. Problem determination mode changes the way message traffic to the system console is managed from the way it was managed in MVS/ESA 4.2 and MVS/ESA 4.2.2.

Initially, during MVS initialization, the system console is set to a non-problem determination (non-PD) mode. The system console receives minimal message traffic in the non-PD mode. In the non-PD mode, the system console is not capable of issuing MVS system commands.

3.4.1.1 Activating Problem Determination Mode

The operator can place the system console in the problem determination (PD) mode by issuing the following command from the system console:

```
VARY CN(*),ACTIVATE
```

The `VARY CN(*),ACTIVATE` command must be issued from the system console that is to be placed in PD mode; otherwise, the system rejects the command.

When the system console is in PD mode, the operator can issue MVS commands to change the console characteristics. After the first activation, the definitions are saved and are used on any subsequent activation.

In the PD mode, the system console behaves like any other MCS console.

To remove the system console from PD mode, the operator can enter the following command from the system console or any authorized console:

```
VARY CN(*|name),DEACTIVATE
```

3.4.2 Non-Problem Determination Mode

When the system console is in non-PD mode, the system ignores the routing codes, UD, MONITOR, LEVEL, MSCOPE and CMDSYS attributes specified in CONSOLxx parmlib member.

In summary, in the non-PD mode the system console:

- May receive synchronous messages and reply to them.
- Does not receive messages issued by ROUTCODE.
- If no other consoles are receiving UD messages, receives UD messages.
- Receives messages queued by system console ID or name.

3.4.3 Defining the System Console

The integrated console or system console is supported only on an IBM ES/9000 processor 9021 or 9121 model, or a S/390 9672 Parallel Enterprise Server processor.

There are two options for defining the system console to MVS:

- Use the CONSOLE statement in the CONSOLxx parmlib by specifying DEVNUM(SYSCONS).

This indicates that the console being defined is the system console attached to this processor. If the SYSCONS keyword is not specified, MCS ignores the definition.

The following CONSOLE statement considerations apply for the system console:

- The only parameters that are accepted with DEVNUM(SYSCONS) are NAME, AUTH, ROUTCODE, LEVEL, UD, MONITOR, MSCOPE, and CMDSYS; all others are ignored.

- Let the system console be defined automatically.

  The system console is defined automatically if one or more of the following is true:

  - No CONSOLxx member is selected for this IPL.
  - CON=NONE is specified either in IEASYSxx or as a system parameter.
  - No CONSOLE statement for the system console is specified in the CONSOLxx member.

  The system console is then assigned the following attributes:

```
DEVNUM(SYSCONS)
NAME(sysname)
AUTH(MASTER)
ROUTCODE(NONE)
LEVEL(ALL)
MFORM(M)
UD(N)
KEY(SYSCONS)
```

Figure 24. Default System Console Parameters

Note: These definitions are not used until the system console is placed into PD mode.

3.4.3.1 CONSOLxx Parmlib Definitions

When defining the system console using the CONSOLE statement, consider the following parameters:

NAME If you do not specify this name, the system generates a name for the system console. If your IEASYSxx member specifies CON=NONE, or if you do not name the system console on the CONSOLE statement in CONSOLxx, MCS tries to use the name of the system to which the console is attached as the name of the system console. (The system name is defined on the SYSNAME parameter in the IEASYSxx member.)

If you do not name the system console and if the system name can be interpreted as a valid device number, for example DEAD, MCS does not use the SYSNAME as the system console name. Instead, MCS assigns an SYSCNxxx name (xxx is a system generated suffix) to the system console. The SYSCNxxx name is also used if you assign a name that is not unique.

Note: If you try to activate a console whose name is the same as an existing system console name, MCS rejects the activation.

AUTH The system console has MASTER command authority by default if AUTH is not specified.

3.4.3.2 Display of System Console

If you display the system console information when the console is not in problem determination (PD) mode, only the default attributes are assigned to the console.

```
D C,CN=SC47
 IEE889I 17.33.03 CONSOLE DISPLAY 687
 MSG: CURR=7 LIM=1500 RPLY:CURR=4 LIM=999 SYS=SC47 PFK=00
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS --------------
  SC47              100  COND=A      AUTH=MASTER       UD=N
   SYSCONS              MFORM=M         LEVEL=ALL
   SC47                 ROUTCDE=NONE
                        CMDSYS=SC47
                        MSCOPE=*ALL
```

SC47 is the default system name for the system console.

3.4.4 Activating the System Console

After you vary the system console into PD mode by issuing the V CN(*),ACTIVATE command, you see the attributes assigned to the console that were defined through the CONSOLE DEVNUM(SYSCONS) statement:

```
V CN(*),ACTIVATE
 IEE712I VARY CN  PROCESSING COMPLETE
D C,CN=SC47
 IEE889I 17.44.30 CONSOLE DISPLAY 704
 MSG: CURR=7 LIM=1500 RPLY:CURR=4 LIM=999 SYS=SC47 PFK=00
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS --------------
  SC47              100  COND=A,PD   AUTH=MASTER       UD=Y
   SYSCONS              MFORM=M         LEVEL=ALL
   SC47                 ROUTCDE=ALL
                        CMDSYS=SC47
                        MSCOPE=*ALL
```

As shown in the command above, the CONSOLxx parmlib definition for the system console is:

```
CONSOLE  DEVNUM(SYSCONS)
         AUTH(MASTER)
         ROUTCODE(ALL)
         LEVEL(ALL)
         UD(Y)
```

The system console, implemented as an extended MCS console in MVS, is automatically defined to MCS during the system initialization. During normal operations, when the system console is not in problem determination mode, it receives a minimal set of messages. When the system console is in problem determination (PD) mode, after a VARY CN(*),ACTIVATE on the system console, an operator can:

- Enter commands and receive messages to help debug the system problem
- Control console attribute values for the system console

After the problems are solved, the V CN(*),DEACTIVATE should be issued on the system console.

3.4.4.1 System Console as an EMCS

The system console in each system in the sysplex has an extended MCS console created for it during system initialization. All system consoles in a sysplex are defined with the same extended MCS console key of SYSCONS:

```
D XCF
 IXC334I  08.04.13  DISPLAY XCF 472
    SYSPLEX WTSCPLX1:   SC42              SC43              SC47
                        SC49              SC50              SC52
                        SC53              SC54              SC55
D C,KEY=SYSCONS
 IEE892I 18.00.05 CONSOLE DISPLAY 725
 MSG: CURR=1    LIM=1500  RPLY: CURR=0    LIM=999  KEY=SYSCONS
   NAME       NAME       NAME       NAME       NAME       NAME       NAME
 SC47       SC55       SC42       SC52       SC43       SC50       SC53
 SC54       SC49
```

3.4.5 Using the System Console During IPL

If one or more NIP console devices are specified in the IODF, the first available device on the list is used as the NIP console. If none of the devices on that list are available, the system console is used as a NIP console.

An initialization message suppression indicator (IMSI) in the LOAD parameter controls the suppression of messages and system prompts during initialization.

Note: You should limit the number of messages to the system console.

3.4.5.1 Message Routing Considerations

The initial message routing during NIP is controlled by the IMSI parameter.

System Console as the NIP Console: When a system console is used as the NIP console consider the following IMSI values:

. or blank A blank or period (.) allows the system console to receive the following messages during initialization:

- Action messages with descriptor codes 1, 2, 3, and 11
- WTOR messages
- Informational messages with descriptor code 12
- Synchronous messages

A or M The system console receives all messages during and after initialization.

Note: This is not recommended because of the large volume of messages that could be routed to the system console. However, if you are testing an IPL or suspect errors in initialization, this is recommended.

NIP Console not the System Console: If a NIP console is used instead of the system console, consider the following IMSI values:

. or blank A blank or period (.) specifies that the system console does not receive messages during initialization. The NIP consoles receive the messages. After initialization, the system console can be used

to receive messages. IMSI is the same whether using SYSCONS or NIPCONS.

A or M The system console does not receive messages during initialization. The NIP console receives all messages during initialization. After initialization, the system console receives all messages unless it is controlled through the CONSOLxx member or an operator command.

3.4.6 System Console Message Traffic

You might consider using the system console for the following types of messages:

SYNCH WTO or WTOR messages that are issued during initialization or recovery situations, or by programs that want messages to bypass normal message queuing.

UD Any messages that have no console destination and are either WTORs or have descriptor code 1, 2, 3, 11, or 12.

Note: Message traffic should not be routed to the system console except for no-console conditions.

3.4.6.1 Minimum Message Traffic

When the system console is in non-PD mode, it receives only:

- UD messages:
  - When you specify UD=(Y) on your CONSOLE DEVNUM(SYSCONS) statement.
  - When no other active console specifies UD=(Y) and the CONSOLxx HARDCOPY statement specifies UD=(N).
- SYNCH messages:
  - When no other MCS consoles are available.
  - When a SYNCH WTOR goes to another MCS console but is not answered for 125 seconds.
  - When a SYNCH message is directed to the integrated console.
- Messages directed to it by console name or ID.

To prevent UD messages from being sent to the integrated console, specify HARDCOPY DEVNUM(SYSLOG),...,UD=(Y) in the CONSOLxx parmlib member.

To prevent SYNCH messages from being sent to the integrated console, specify DEFAULT SYNCHDEST(cngrp) in the CONSOLxx parmlib member.

Note: The system console is the backup if all the members of the SYNCHDEST group are unavailable.

3.4.6.2 Maximum Message Traffic

When the system console is in PD mode, it receives messages according to its ROUTCODE and MSCOPE settings in addition to the minimum message traffic set described in 3.4.6.1, “Minimum Message Traffic.” Refer to *MVS/ESA Planning: Operation* for more information regarding planning and usage for the system console.

3.5 Subsystem Consoles

Subsystem consoles is the old original MVS answer for the requirement to be able to write programs that act as operators.

Subsystem consoles, more accurately subsystem-allocatable consoles, are used primarily by JES3 releases prior to JES3 5.2.1 or NetView to enter operator commands and to get an identifiable response back. Subsystem consoles support, per se, does not provide any message presentation services. Messages routed to subsystem consoles are not queued for retrieval and there is no macro services to obtain the messages from the console. The users of subsystem consoles must extract solicited and unsolicited messages directed to a subsystem console from the WTO(R) subsystem interface broadcast loop.

When a command is issued through the MGCR(E) macro, the macro parameters include the console ID of the console that issues the command. When commands are processed, the processors route the command responses back to the originating console by using the console ID that was associated with the command. On the WTO(R) SSI loop, the response messages (WQE) include among other things the console ID of the console to receive the message. A subsystem console user that has issued a command must identify the response by matching the WQE console ID against the MGCR(E) console ID. Subsystem consoles have 1-byte console IDs.

Note: Commands can also be issued with a special console ID 0 (zero). However, there was no mechanism to uniquely relate commands issued with console ID 0 and their responses before MVS/ESA SP Version 4.

MVS/ESA SP Version 4 MCS introduced the concept of a command and response token (CART), which associates a response with a command. The command responses must still be extracted from the SSI loop. The problem with the CART is that not all old command processors support CART; that is, they do not include the CART in the command response WTOs. For example, JES3 releases prior to JES3 5.2.1 do not support CART or 4-byte console IDs used by extended MCS consoles.

3.5.1 Application Use of a Subsystem Console

To obtain ownership of a subsystem console, your application must use the IEAVG700 services which allow you to acquire or to alter the use of a subsystem console. The services include:

- Obtain a subsystem allocatable console for use by an authorized application.
- Release the console.
- Indicate that messages should be or should not broadcast to all subsystems.
- Change the routing codes of the console.
- Determine the status of a console (In use, not in use, not defined).
- Determine which type of protocol should be used to allow a subsystem to issue commands and monitor messages.
- Determine which system or subsystem commands can be used to configure the console.
- Indicate that a console dedicated to a system component should be considered a pseudo master console so that it can be able to issue a subset of master console only commands.

- Determine the authority of a console (master, pseudo-master, SYS, IO, and CONS). The console zero is considered to have SYS, IO, and CONS authority.
- Release one or more consoles by ASID.
- Broadcast updates to other systems in a sysplex.

See *MVS/ESA Using the Subsystem Interface* for more information.

Note: It is recommended that you use extended MCS consoles instead of subsystem consoles to issue commands and to trap messages from the MVS message stream.

3.5.2 Defining Subsystem Consoles

Subsystem-allocatable consoles are defined in the CONSOLxx parmlib member with the CONSOLE DEVNUM(SUBSYSTEM) statement. The only parameters that are valid with DEVNUM(SUBSYSTEM) are AUTH and NAME.

Note: You should always define the NAME parameter for subsystem consoles.

You can display the allocation status of the subsystem consoles with the D C,SS command:

```
D C,SS
 IEE889I 11.30.47 CONSOLE DISPLAY 541
 MSG: CURR=6    LIM=1500 RPLY:CURR=4    LIM=999  SYS=SC50      PFK=NONE
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS ---------------
  S01/VAIN0023       11  COND=SS     AUTH=ALL          NBUF=N/A
   SC50                   ROUTCDE=NONE
  S02                12  COND=SS     AUTH=ALL          NBUF=N/A
                         ROUTCDE=NONE
```

In the example, subsystem console S01 is allocated to the application VAIN. Console S02 is not in use by any applications.

3.6 Hardware Management Console Operation

There are various hardware configurations that make up a sysplex. Depending on the processors in the sysplex, the choice of consoles to run the sysplex becomes important.

1. A sysplex with all 9672s

   For a group of 9672s with a maximum of 32 system images to be coupled together and form a sysplex, you can connect all the 9672s in a single Local Area Network (LAN). Up to four hardware management consoles can be attached to the LAN and manage the whole sysplex through any of the HMCs.

2. A combination of 711-base and 511-base ES/9000 machines

   In this case, each individual machine or physical partition must be activated separately. After activation, you can use MCS consoles or extended MCS consoles to manage the sysplex operation.

3. A sysplex consisting of 9672s, 9674s, 711-base and 511-base ES/9000 all coupled together through coupling facility LPARs running in a 9674 or a 9672 or 711-base ES/9000, as shown in Figure 25

   In this case, the HMC can only manage the 9672 and the 9674 that are connected in a single LAN. Each 711-base and 511-base ES/9000 must be activated separately, unless you have automation products to automate the activation and deactivation of all members in the sysplex.

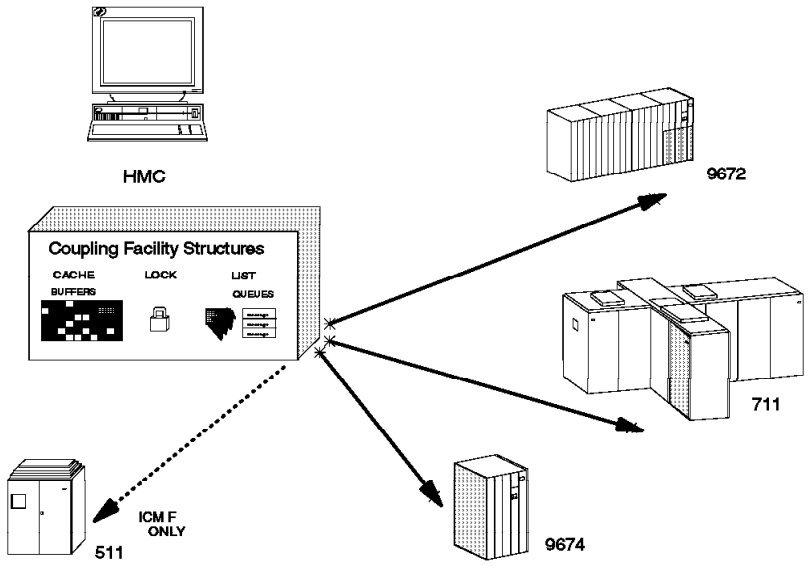


Figure 25. HMC in a Sysplex Environment

In each possible sysplex configuration, you should always have at least one channel-attached 3270 MCS console to bring up the first member in the sysplex. You should use the channel-attached console to bring down the last system in the sysplex for the following reasons:

- The starting of the first system and the shutdown of the last system are critical moments in the life cycle of a sysplex.
- Channel-attached 3270 MCS console provides a good monitor for the large number of sysplex console messages that occur during the initialization and shutdown stage of a sysplex member.

3.6.1 IPL Messages and the HMC

On S/390 9672 Parallel Enterprise Server models, the *hardware management console* (HMC) provides the console integration function through the *operating system messages* panel displays.

Chapter 4. Extended MCS Consoles

The extended MCS console (EMCS) is a set of programmable interfaces introduced in MVS/ESA SP Version 4 that makes it possible for a program to enter commands and receive messages as though it were an MCS console.

Examples of extended MCS consoles supporters:

- TSO/E users with the appropriate RACF authorization can use the CONSOLE command to establish an extended MCS console session. TSO/E extended MCS consoles can be authorized to issue commands requiring master authority. See 4.4, “TSO/E Extended MCS Console” on page 112 and 4.4.3, “RACF Control for TSO/E Commands” on page 115.

  TSO/E users can write REXX programs that act as an operator. The CONSOLE command is used to activate the EMCS console and to issue operator commands. The CONSPROF TSO/E command can be used to control message delivery, and the GETMSG REXX function is used to retrieve queued messages. See 4.5, “REXX EXECs for Operator Tasks” on page 117.

- System Display and Search Facility (SDSF) for JES2 is a program that runs on TSO/E and uses Interactive System Productivity Facility (ISPF) functions. SDSF allows users to issue commands to the MVS system and to view the hardcopy log (SYSLOG) to see the responses to their commands and to monitor the system. See 7.3.2.1, “SDSF Extended MCS Console” on page 158.

  SDSF ULOG, available with SDSF 1.4, allows you to display the MVS/JES2 commands and responses issued during your SDSF session, including commands generated by SDSF. The ULOG function uses EMCS console services. See 7.3.2, “User Session Log” on page 157.

- NetView Version 2 Release 3 has an option to use extended MCS consoles for MVS communication. Earlier versions of NetView could only use the SSI. In addition, individual NetView operators and autotasks can allocate extended MCS consoles by issuing the NetView GETCONID command. See 4.6, “NetView Extended MCS Consoles” on page 122.

It is recommended that you use extended MCS consoles when you write an authorized program that acts as an operator. Appendix G, “REXX EXEC for Extended MCS Console” on page 195 provides a sample program for an extended MCS console and includes an example of the ISPF panel you can use under your TSO session, as shown in Figure 31 on page 119. Extended MCS consoles also provide relief for the number of consoles that can be used in an MVS system as they are not included in the 99 MCS and subsystem consoles in a sysplex limit.

For an example on how to activate and use the extended MCS console related programmable macro services, see Appendix H, “Extended MCS Console Sample” on page 203.

Programmable Interface Macros: Extended MCS consoles are not pre-defined. When a program wants to establish an extended MCS console, it uses a set of programmable interfaces. Three authorized macros are available with this programmable interface:

- MCSOPER
- MCSOPMSG
- MGCRE

The MCSOPER macro is used to activate an EMCS console. Once the extended MCS console is activated, the program can receive messages and command responses by invoking the MCSOPMSG macro service, and can issue commands by issuing the MGCRE macro. As an example, JES3 5.2.1, TSO/E, NetView 2.3, and SDSF are programs that use EMCSs. Appendix I, “Release Migration IDs” on page 209 shows an example of an authorized assembler program that uses the MVS macro services for extended MCS consoles.

The programmable interface can be used by a:

- TSO/E user
- Batch job
- Started task address space

The TSO/E CONSOLE and CONSPROF commands (see 4.4, “TSO/E Extended MCS Console” on page 112) make use of the programmable interfaces and can be used by a TSO/E user or applications written in REXX.

4.1 MCSOPER Macro Service

The MCSOPER macro service allows you to activate and manage extended MCS consoles. You can specify only one of the following functions each time you invoke the MCSOPER macro:

REQUEST=ACTIVATE Initializes an extended MCS console session.

The MCSOPER macro with REQUEST=ACTIVATE defines and activates an extended MCS console to the system. When you activate an extended MCS console, MCS creates a dataspace to store messages and DOM requests. There is one dataspace for every address space with an active extended MCS console. Therefore, if an address space has two active extended MCS consoles, both share the same message dataspace. Note that the system deletes this dataspace upon deactivation of all the extended MCS consoles in the address space.

REQUEST=DEACTIVATE Terminates the session. Through MCSOPER with REQUEST=DEACTIVATE and ABTERM=YES, you can deactivate an active console and switch processing from an extended MCS console to another active MCS or extended MCS console. Before deactivating a console, you must define a valid alternate group for the console through operator attributes. Switching an extended MCS console to an alternate console allows processing to continue without interruption, and is useful if a

program representing an extended MCS console abnormally ends, and your installation needs to have its processing taken over by another console.

REQUEST=RELEASE Releases a migration ID from an extended MCS console that has been deactivated by issuing REQUEST=DEACTIVATE.

See Appendix I, “Release Migration IDs” on page 209 for an example on how to use MCSOPER REQUEST=RELEASE service.

Messages are stored as message data blocks (MDBs) in the dataspace. If a DOM is directed to a previously issued message, MCS creates a separate MDB for the DOM and also stores it into the dataspace.

4.1.1 Activating an EMCS

When a program establishes an extended MCS console, it invokes the MCSOPER REQUEST=ACTIVATE service to activate the console. When activating the extended MCS console, you need to specify its attributes, such as:

- Command authority
- Routing codes
- Message dataspace size
- Migration ID if required
- Hardcopy message set if required

  Note: This has a high potential of producing a tremendous number of messages, so be careful.

These attributes are known as *operator parameters*.

4.1.1.1 Extended MCS Consoles Attributes

You need to specify the extended MCS console’s attributes when it is activated. You specify the operator parameters as follows:

- In the OPERPARM segment of the user profile of a security product, such as RACF. See 4.1.3, “EMCS With RACF Control” on page 101.

  Note: You can override the console attributes specified in the user profile of the security product by turning on the MCSOVRDY bit in the MCSOP data area. Therefore, if you specify the extended MCS console attributes in the MCSOP data area and request security product override, the MCSOP data area specifications are used (even if the RACF OPERPARM data is also specified for the user). For a description of the OPERPARM override bits, see 4.1.3.3, “MCSOP Data Area” on page 103.
- In the OPERPARM data area MCSOP, mapped by IEZVG111, by specifying the OPERPARM parameter on the MCSOPER macro. See 4.1.3.3, “MCSOP Data Area” on page 103.
- Through system defaults. See 4.1.3.2, “OPERPARM Segment Defaults” on page 103.

MCS first checks for the presence of the MCSOP data area. If the MCSOP data area is coded on the MCSOPER REQUEST=ACTIVATE, as shown in Figure 26, and it does not request RACF processing override, or the MCSOP data area is not specified, the attributes are checked in the following order:

- First MCS looks in the RACF user profile for an OPERPARM segment and if it is defined, it supplies the extended MCS console attributes.
- If the RACF OPERPARM segment is not defined, the MCSOP data area definitions are used if present.
- Finally, if the MCSOP data area is not specified, MCS applies default values for the console attributes.

Note: Extended MCS consoles can receive the hardcopy message set. To request that an extended MCS console receive the hardcopy message set, specify the HARDCOPY attribute and the override security product bit MCSOVRDY in the MCSOP data area. Note that you cannot request an extended MCS console to receive the hardcopy message set through the RACF OPERPARM segment definitions.

4.1.1.2 MCSOPER Macro Example

The MCSOPER macro enables an authorized program to activate and deactivate an extended MCS console. The macro also provides a means for storing messages and command responses.

Figure 26 shows an example of how an extended MCS console can be activated.

```
         MCSOPER REQUEST=ACTIVATE,                                     +
               NAME=NAME,                                              +
               CONSID=CONSID,                                          +
               TERMINAL=TERMINAL,                                      +
               MCSCSA=CSA,                                             +
               MCSCSAA=CSAA,                                           +
               OPERPARM=OPERPARM,                                      +
               MSGDLVRY=FIFO,                                          +
               MSGECB=MSGECB
         .
NAME     DC    CL8'CONSO1'
CONSID   DS    CL4
TERMINAL DC    CL8'SDLC0001'
         DS    0F
CSA      DS    A
CSAA     DS    F
MSGECB   DS    F
OPERPARM DS    CL(MCSOPLEN)
         .
         IEZVG111
         .
```

Figure 26. MCSOPER Macro Example

When the MCSOPER macro with its associated parameters is issued, an extended MCS console is activated. Before the program terminates, or before an MCSOPER REQUEST=DEACTIVATE macro is issued, you can enter the MVS DISPLAY CONSOLES,CN=CONSO1 command to display information about the extended MCS console.

The OPERPARM= keyword specifies an area where you can define the operator parameters for an extended MCS console. IEZVG111 is the mapping macro for mapping this area along with EQU statements that you can use to set bit values such as console authority, and so on. See 4.1.3.1, “OPERPARM Segment” on page 101 and 4.1.3.3, “MCSOP Data Area” on page 103 for more details.

4.1.2 EMCS Without RACF Control

When an application program issues the MCSOPER REQUEST=ACTIVATE macro, you have the choice to pass all of your OPERPARM data to MCSOPER yourself. You need to pass all of the OPERPARM values that you do not want to be defaulted. Therefore, the system uses the values passed in place of those from an OPERPARM segment or any IBM supplied defaults.

Note: To request a security product override to prevent RACF authority verification from being performed, turn on the MCSOVRDY bit in the MSCOP data area when issuing the MCSOPER macro. Also, see 4.1.1, “Activating an EMCS” on page 99.

4.1.3 EMCS With RACF Control

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 an application program that issues the MCSOPER REQUEST=ACTIVATE macro, the console-name is the name specified on the MCSOPER macro. However, if the MCSOPER macro specifies OPERPARM data that requests the security product override, the RACF authority verification is not performed. The MSCOPER macro expects a user profile with the same name as the EMCS console, and it retrieves the OPERPARM data from that user profile.

4.1.3.1 OPERPARM Segment

The RACF OPERPARM segment includes the following parameters:

```
OPERPARM(
 [ ALTGRP(alternate-console-group) ]
 [ AUTH(operator-authority) ]
 [ AUTO( YES | NO ) ]
 [ CMDSYS(system-name) ]
 [ DOM( NORMAL | ALL | NONE ) ]
 [ KEY(searching-key) ]
 [ LEVEL(message-level) ]
 [ LOGCMDRESP( SYSTEM | NO ) ]
 [ MFORM(message-format) ]
 [ MIGID( YES | NO ) ]
 [ MONITOR(event) ]
 [ MSCOPE( system-names | * | *ALL ) ]
 [ ROUTCODE( ALL | NONE | routing-codes ) ]
 [ STORAGE(amount) ]
 [ UD( YES | NO ) ]
 )
```

Figure 27. RACF OPERPARM Segment of a User's Profile

The following parameters are implemented for extended MCS consoles because of their uniqueness and you should carefully evaluate the meaning and use. They do not have corresponding definitions on the CONSOLE statement:

AUTO(YES|NO) Specifies whether the extended console receives messages that have been automated by the message processing facility (MPF) in the sysplex.

DOM(NORMAL|ALL|NONE) Specifies whether this console receives delete operator message (DOM) requests.

NORMAL Specifies that the system queues all appropriate DOM requests to this console.

ALL Specifies that all systems in the sysplex queue DOM requests to this console.

NONE Specifies that no DOM requests are queued to this console.

See 4.1.4.4, “Message Queueing and DOMs” on page 106.

KEY(searching-key) Specifies a 1-to-8-byte character name that can be used to display information for all consoles with the specified key by using the MVS command DISPLAY CONSOLES,KEY. The default value for the KEY is “NONE.”

LOGCMDRESP(SYSTEM|NO) Specifies if command responses are to be logged in SYSLOG or OPERLOG.

Note: Because it is possible to flood the SYSLOG or OPERLOG if too many EMCS consoles were issuing commands, this option can be used to suppress the resulting messages from being logged. Specify LOGCMDRESP(NO) to suppress the messages.

MIGID(YES|NO) Specifies that a 1-byte migration ID is to be assigned to this console. The migration ID allows command processors that use the 1-byte console ID to route command responses to this console.

MCS consoles are assigned both 4-byte and 1-byte console identifiers. When commands are entered through a console, the console ID is passed to the command processors, which use it to route command responses back to the issuing console. EMCS consoles have a 4-byte console ID. When an EMCS console is activated, a 1-byte migration ID may also be requested. “Old” command processors, for example JES3 releases prior to JES3 5.2.1, were not programmed to use the 4-byte console ID and return command responses to the originating console that only has a 4-byte ID. Messages can be routed to an EMCS console with a migration ID assigned either using the 1-byte or 4-byte console ID.

STORAGE(amount) Specifies the size of the message dataspace, in megabytes, that is used for message queuing to this console. If specified, STORAGE must be a number between 1 and 2000.

4.1.3.2 OPERPARM Segment Defaults

The OPERPARM default values are:

Table 6. OPERPARM Segment Defaults

Parameter	Default
Authority	INFO
Routing codes	NONE
Message level	ALL
Message format	M
Message scope	*ALL
Command scope	* (current system)
Monitor information	NONE
Log command responses	SYSTEM
Migration ID	NO
Storage	1 (megabytes in a dataspace)
DOM	NORMAL
Extended MCS console key	“NONE”
Undelivered messages	NO
Automation messages	NO
Hardcopy	NO

4.1.3.3 MCSOP Data Area

When using the data area MCSOP, mapped by IEZVG111, you specify the OPERPARM parameter on the MCSOPER macro. Through the MCSOP data area fields, you can specify the same parameter data as through the RACF OPERPARM segment. In addition you can request to receive the hardcopy message set.

Note: The RACF OPERPARM segment does not allow you to request receiving the hardcopy message set.

The MCSOPER processing looks for information on operator parameters first in the RACF user profile. If the operator parameters are defined in the RACF user profile, they are assigned to the console unless OPERPARM data is passed on the MCSOPER macro and it specifically requests *not* to use user profile data. If the user profile does not include operator parameters, the OPERPARM data passed on the MCSOPER macro is used. If no OPERPARM data is passed on the MCSOPER macro, the system default values are assigned to the extended MCS console.

OPERPARM Override Bits: A program activating an extended MCS console can force the OPERPARM data to be used by controlling the setting of the MCSOVRDY and MCSOVRDN bits in the MCSOP data area. The following is the order of processing for OPERPARMs as determined by the bits:

MCSOVRDY If the MCSOVRDY bit is on, this indicates to override the security product. Processing uses the OPERPARM data area to set the extended console′s attributes.

MCSOVRDN If the MCSOVRDN bit is on, this indicates to not override the security product, which is the default option. The process searches the security product for an OPERPARM segment. If no segment exists, the process then uses this data area to set the extended console′s attributes.

4.1.4 Controlling EMCS Consoles

When a program wants to establish an extended MCS console, it invokes the MCSOPER REQUEST=ACTIVATE service to activate the console. Once the extended MCS console is activated, the MCSOPER macro provides options for:

- Controlling message delivery
- Controlling the number of messages
- Controlling the queueing of messages

The program receives messages and command responses by invoking the MCSOPMSG macro service, and can issue commands by issuing the MGCRE macro.

4.1.4.1 Controlling Message Delivery

The MCSOPER macro service provides the following options for controlling the message delivery through its MSGDLVRY parameter:

- If you want MCSOPER to queue messages to and extract messages from the message dataspace on a first-in first-out basis, specify MSGDLVRY=FIFO on MCSOPER.
- If you want to use the search arguments CMDRESP, CART, and MASK when issuing the MCSOPMSG macro with the REQUEST=GETMSG parameter, specify MSGDLVRY=SEARCH on MCSOPER.
- If you do not want to receive messages on a console, specify MSGDLVRY=NONE on MCSOPER. MSGDLVRY=NONE requires that you retrieve messages from either the subsystem interface or an MPF exit, rather than from the message dataspace. MSGDLVRY=NONE is useful if you want to issue commands from a console but do not want to receive messages on a console. The console ID identifies the particular console′s messages on the subsystem interface or MPF exit.

4.1.4.2 Controlling Number of Messages

The number of messages queued to an extended console and the size of the message dataspace is controlled as follows:

- The STORAGE specification in the MCSOP data area or in the OPERPARM RACF segment defines the size of the message dataspace created for an extended console. The range of the STORAGE specification is from 1 to 2048, where unit of measure is 1 Mb of dataspace storage. The number of messages going to your data space is limited by storage size. If you want to further control the number of messages going to your dataspace, use the QLIMIT parameter.
- The QLIMIT parameter on the MCSOPER macro specifies the maximum number of messages queued to an extended MCS console. The value can

range from 1 to 2147483647. 350 bytes can be used as a rough estimate for the size of a stored console message. The default QLIMIT of 2147483647 is likely to cause you to run out of storage in the message dataspace before the limit is hit.

- The ALERTPCT parameter on the MCSOPER macro specifies a percentage of the QLIMIT value. When the number of messages queued to an extended MCS console exceeds the number represented by this percentage, the system alerts the extended MCS console by posting an ECB, identified by the ALERTECB parameter. For example, if the QLIMIT value is 2000, and the ALERTPCT value is 50, the system alerts the extended MCS console user when the number of messages in the message dataspace exceeds 50%, or 1000 messages, of the QLIMIT.

Tailoring Storage Use: QLIMIT controls the maximum number of messages that can go to the extended MCS console. The STORAGE value in the OPERPARM controls the maximum number of megabytes for the dataspace allocated to the console. The dataspace is shared by all extended MCS consoles in the same address space. Thus, if you have three extended MCS consoles activated in an address space, they share the same dataspace. You can estimate around 250 messages per megabyte of storage in the dataspace. Now, when tailoring these two values, you may want to queue until the message dataspace fills up, or until a specified number of messages is delivered to your console. If you want to queue until your message dataspace fills up, you can code the maximum value for QLIMIT. If you want to queue until a specified number of messages is delivered, you code the appropriate number for QLIMIT. In both cases, queueing is done until either of these limits is reached. At that point, queueing stops and the disposition of the console is flagged in the MCSCSA (returned to the application on the MCSOPER REQUEST=ACTIVATE).

EMCS Storage Considerations: When an extended MCS console session becomes active in an address space, a dataspace is created. The dataspace contains message data blocks (MDBs), which are used to store messages and command responses for all extended MCS consoles sessions in the address space. There are approximately 250 messages per megabyte. When invoked, the TSO/E REXX GETMSG function or the MCSOPMSG macro returns the messages or command responses in the dataspace created for the extended MCS console in the form of an MDB. With the MCSOPMSG macro or GETMSG, you receive one message at a time. You can see the number of dataspaces created as the result of the active extended MCS consoles by using the MVS D A,CONSOLE command, as shown in Figure 28.

```
D A,CONSOLE
 IEE105I 08.45.55 90.207 ACTIVITY 258
  JOBS      M/S      TS USERS      SYSAS      INITS     ACTIVE/MAX VTAM
 00000     00008      00001        00013      00005      00001/00010
  CONSOLE  CONSOLE           NSW  *   A=0008   PER=NO   SMC=000
                                      PGN=001  DMN=001  AFF=NONE
                                      CT=011.899S  ET=17.33.03
                                      ADDR SPACE ASTE=01AE0200
                                      DSPNAME=00000IEE ASTE=0098E000
                                      DSPNAME=IEEMCS03 ASTE=01633500
                                      DSPNAME=IEEMCS02 ASTE=01633480
                                      DSPNAME=IEEMCS01 ASTE=01633400
```

Figure 28. DISPLAY A,CONSOLE Command Example

In the figure, there is currently one extended MCS console address space active. The dataspace name for that active extended MCS console session is 00000IEE. If a second and third extended MCS console address space becomes active later, they would have dataspaces with the names of 00001IEE and 00002IEE.

You should use the STORAGE parameter in the OPERPARM segment to limit storage allocation. See 4.1.3.2, “OPERPARM Segment Defaults” on page 103.

4.1.4.3 Controlling Message Queueing

Message queuing stops when no more storage remains or the maximum number of messages is reached in the message dataspace. The ALERTECB parameter on the MCSOPER macro specifies the ECB to be posted when message queuing stops. When ALERTECB is posted, check the MCS console status area, MCSCSA to determine why queuing has stopped. If the message dataspace is full, the MCSCMLIM field contains a one. If the maximum number of messages has been reached, the MCSCDLIM field contains a one. In either case, to start message queuing again, retrieve messages from the MDB by issuing the MCSOPMSG macro.

The ALERTECB may be posted for the following conditions as well:

- An internal queuing error has occurred and message queuing has stopped. The MCSCSA internal error field contains a one. Issue MCSOPMSG REQUEST=RESUME to restart queuing.
- The extended console session has been switched to another console by using a SWITCH CN command, and message queuing has stopped. The MCSCSA suspend operator field contains a one. Issue MCSOPMSG REQUEST=GETMSG to retrieve all the messages in the data space, then issue MCSOPER REQUEST=DEACTIVATE to deactivate the extended console.

By issuing MCSOPMSG REQUEST=RESUME, you can resume queuing to the message dataspace after any abend that suspends system message queuing, or after the failure of recovery routines invoked. Some or all messages currently queued might be lost. REQUEST=RESUME enables you to resume queuing without having to retrieve enough messages to reach the QRESUME percentage. If queuing does stop, MCS routes important messages not queued anywhere else to consoles with the UD (undelivered message) attribute.

4.1.4.4 Message Queueing and DOMs

The DOM specification in the MCSOP data area or in the OPERPARM RACF segment defines whether DOM requests should be communicated to the extended MCS console. If DOMs are requested, then in the DOM MDBs the MDBDOMFL field indicates whether the console specified DOM(NORMAL) or DOM(ALL).

Note: If you are not doing any special processing of messages that require a DOM, such as keeping them on a queue of your own, you do not need DOMs and you should specify DOM(NONE) for the EMCS console. If the EMCS console has been defined as:

DOM(NORMAL) MDBGMID in the DOM MDB matches the same field in the message MDB

Note: DOM(NORMAL) does not work if AMRF(N) is specified on the INIT statement.

DOM(ALL) The DOM field values in the control program object indicates which message or messages this DOM is to delete

DOM(NONE) Indicates that no DOM MDBs should be queued for the extended MCS console

Note: A single DOM MDB can delete more than one message.

4.1.5 Deactivating an Extended MCS Console

When you stop using an extended MCS console, you deactivate the console with the MCSOPER REQUEST=DEACTIVATE request. Note that the extended MCS console remains known to MCS in an inactive state. The effect is similar to a VARY OFFLINE of a MCS console. The console can be reactivated at a later time.

```
D C,CN=VAIN
 IEE889I 21.25.20 CONSOLE DISPLAY 004
 MSG: CURR=6    LIM=1500 RPLY:CURR=4    LIM=999  SYS=SC50      PFK=NONE
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS ---------------
  VAIN              110  COND=N      AUTH=MASTER       UD=Y
   EMCS                  MFORM=T,S,J,X   LEVEL=ALL
                         ROUTCDE=ALL
                         CMDSYS=SC50
                         MSCOPE=*ALL
```

The extended MCS console in the example was activated with the TSO/E CONSOLE command and the RACF OPERPARM segment requested a migration ID to be assigned to the console. When TSO/E deactivates the console, it does not release the migration ID. However, if the same TSO/E user reactivates the same console, the same migration ID is assigned by the system. If over time, there are more users requesting migration IDs in the sysplex than there are IDs, this creates a problem.

MCS has a limited number of migration IDs (range 100-127,129-250). However, MVS can reuse migration IDs that have been released.

4.1.6 Releasing an EMCS Migration ID

To release a migration ID you have to write an authorized assembler program that invokes the MCSOPER REQUEST=RELEASE service. You supply the migration ID you want to release as input on the MCSOPER REQUEST=RELEASE request. As there is no easy way the find out what migration IDs are assigned, your program could loop through the whole range of IDs and attempt to release every one of them. If a migration ID is currently in use or it is not assigned, the MCSOPER REQUEST=RELEASE request return code and reason code states the fact, but no damage will be done to the system.

Appendix I, “Release Migration IDs” on page 209 shows an example of a release migration ID program.

Note: The DISPLAY C,KEY=keyname command does not display extended MCS consoles that are inactive. A key is assigned to the console only when it is active and may vary from activation to activation.

4.2 MCSOPMSG Macro Service

Once your extended MCS console is activated, you can use it to perform the following functions:

- Receive unsolicited messages
- Receive command responses
- Receive the hardcopy message set
- Issue commands.

Messages are sent to an extended MCS console when WTO and WTOR macros are issued with the routing information that is received by your console. Messages are queued as message data blocks (MDBs) to the extended MCS console. If a message is too long to be stored as one MDB, additional MDBs are chained together to contain the whole message. The MDBs are chained through the MDB prefix.

Once a message is queued for the extended MCS console, the message ECB specified on MSGECB parameter of the MCSOPER macro is posted. Now, you can receive the message, and free storage for new messages, by issuing the MCSOPMSG macro with the REQUEST=GETMSG option. You can also receive messages by specifying certain message types. The CMDRESP option of MCSOPMSG allows you to retrieve the following messages:

- A command response (CMDRESP=YES)
- An unsolicited message or DOM (CMDRESP=NO)

If you choose to selectively receive command responses, you can further specify the message command and response token (CART), which associates a response with a command. You can also specify a CART with a mask. The CART is a user-defined value originally specified on the MGCRE macro. When you specify a CART and a mask, MCSOPMSG REQUEST=GETMSG logically ANDs the mask value with the CART value and also ANDs the mask with the CARTs associated with the messages that have been queued to your console. MCSOPMSG processing then compares the results of the AND operations. If a comparison matches, MCSOPMSG retrieves the message. Otherwise, no message is retrieved.

4.2.1 MCSOPMSG Macro Example

The MCSOPMSG macro enables an authorized program to retrieve messages or command responses for the extended MCS console that was set up using the MCSOPER macro. The message information is returned in a message data block (MDB), which is in a dataspace. A mapping macro IEAVM105 is available to map the MDB.

Figure 29 shows an example of how MCSOPMSG can be used to retrieve messages or command responses. This program assumes that MSGDLVRY=SEARCH was specified on MCSOPER.

```
         .
         MCSOPMSG  REQUEST=GETMSG,                                     +
               CMDRESP=NO,                                             +
               CONSID=CONSID
         .
CONSID   DS    CL4
         .
         IEAVM105      Required mapping for MCSOPMSG.
         .
```

Figure 29. MCSOPMSG Macro Example

The CMDRESP=NO parameter specifies the type of message search. NO indicates that MCSOPMSG obtains the next unsolicited message queued to the extended MCS console. If YES is specified, it indicates that MCSOPMSG returns the next command response message.

Note: MSGDLVRY=SEARCH must be specified on the activate request to use CMDRESP.

When you do not specify CMDRESP, MCSOPMSG retrieves the next message in the queue on a first-in first-out basis.

A successful MCSOPMSG REQUEST=GETMSG returns the address of the selected MDB in general-register (GR) 1 and its access-list-entry token (ALET) in access-register (AR) 1. User applications are expected to copy the MDB from the message dataspace before processing it. On the next MCSOPMSG invocation the MDB in the message dataspace is deleted.

4.2.1.1 Message Data Block

A message data block is a data area (mapped by IEAVM105) that contains either a message or a DOM directed to an extended MCS console, and any information related to that message. It also contains a prefix area (mapped by IEAVG132). The prefix area points to any subsequent MDBs for a given message. If there are no subsequent MDBs, the MDBPNEXT field is set to zero.

Figure 30 illustrates the structure of an MDB for a message that requires more than one MDB.

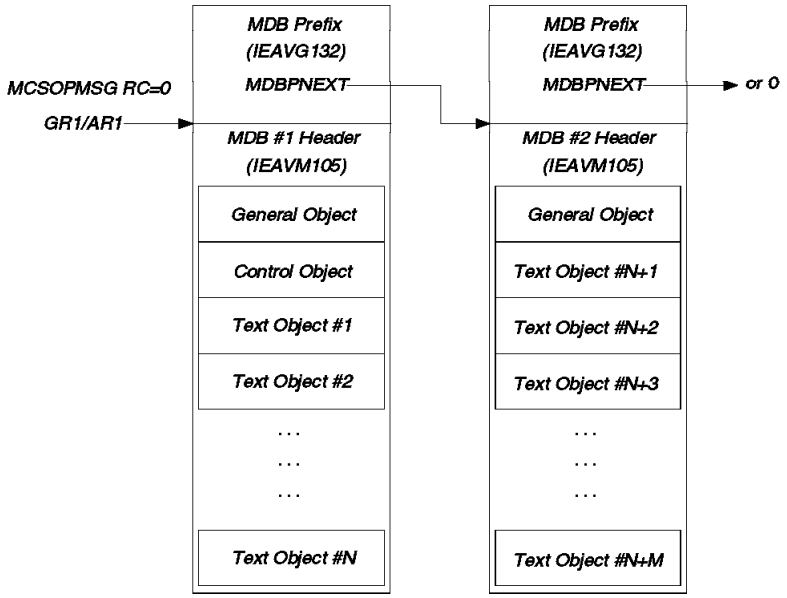


Figure 30. MDB Structure

The entire MDB is headed by a headers section that identifies the block as an MDB and includes the version (MDBVER) and the length of the entire MDB block including all the objects (MDBLEN). After the header comes one or more objects (sub-blocks). The objects currently used are:

- General object (MDBG)
- Control program object (MDBSCP)
- Message text object (MDBT)

The general object is always present. All others are optional. The objects can be in any order within the MDB.

Each object block is architected such that the first two bytes contains the length of that particular object and the second two bytes contains the object type.

To find the first object, add the length of the MDB header to the MDB address. The object type tells you whether it is a general, control, or text object. To find the end of the first object, which is also the beginning of the next object, add the length of that object to the end of the header. Subsequent objects can be found by adding the length of the current object to the address of the current object.

You can find the end of the MDB by adding the total length of the MDB, which is contained in the first field of the header, to the address of the MDB.

The three MDB objects are:

- The general object, which contains information about the message or DOM. It contains a message ID, time stamp, and indication of whether the MDB holds a WTO, WTOR, or a DOM. Each MDB has one general object.
- The control program, which contains information about the message or DOM that is specific to MVS. If the message is a WTO or WTOR, the control program object contains routing codes, message level, job name, and other

message information. If the MDB contains a DOM, the control program object contains the ASID, jobstep TCB, or system ID that deleted the message. Each MDB has one control program object.

- The text object, which contains the variable length message text. Each line of a message is represented by one text object. If the MDB cannot hold all text objects for a specific message, any remaining text objects are placed in one or more additional MDBs. The MDB prefix points to the next MDB. The continuation MDBs contain only the general object and the remaining text objects.

The general objects and the control program objects may appear anywhere in the first MDB, but text objects are sorted in order within and among MDBs.

If the system sends a DOM to an extended MCS console, the MDB contains only two objects: the general object and the control program object.

4.3 MGCRE Macro Service

To issue commands from an extended MCS console, you should use the MGCRE macro with the CONSID or CONSNAME parameter. You identify the console that issues the command by specifying a 4-byte console ID on the CONSID parameter or a console name on the CONSNAME parameter. MCSOPER returns the 4-byte console ID when you activate an extended MCS console. CONSNAME must be the same value you specified on the NAME parameter of MCSOPER. This console ID or name identifies the console that receives responses to the commands you issue with MGCRE. In general, the CONVCON macro service may be used to retrieve a console name for a console ID or vice versa.

You may also specify a value for the CART parameter if you want to identify the response message for a particular command, or if you want to retrieve response messages with the MCSOPMSG macro based on CARTs or masks.

Other MGCRE macro parameters include:

TEXT Specifies the address of a command area. The first 2 bytes of this command area contain the length of the command. The command text immediately follows this 2-byte area, and can be up to 126 characters.

CMDFLAG CMDFLAG=NOHCPY requests that no copy of the command appear in the hardcopy log. If you do not specify this option, the system logs the command in the hardcopy log.

TOKEN Specifies the optional input field that contains a 31-bit right-justified program token for the command specified in the TEXT parameter. The token field may contain any information, such as an identifier that indicates the issuing program. The token field is available for subsystem interface processing through the SSOB extension for command processing exit (mapped by IEFSSCM) and MPF command installation exits through the CMDX parameter area (mapped by IEZVX101).

UTOKEN Specifies the optional input field that contains the address of a security token for the command identified in the TEXT parameter. You can obtain the UTOKEN value by using the RACROUTE REQUEST=TOKENXTR, RACROUTE REQUEST=VERIFYX, or RACROUTE REQUEST=TOKENBLD macros. Command processing

passes the UTOKEN to RACF to validate the authority of the issuer. The UTOKEN should be that of the user on whose behalf the command is issued.

Note: This parameter is optional. If you do not specify it, the UTOKEN of the address space is used.

4.4 TSO/E Extended MCS Console

There are two TSO/E commands that allow users with CONSOLE command authority to establish an extended MCS console, they are:

CONSOLE The TSO/E CONSOLE command allows users with CONSOLE command authority to perform MVS operator functions from a TSO/E terminal. The CONSOLE command establishes an extended MCS console session. During an extended MCS console session, users can enter MVS system and subsystem commands and obtain responses to those commands. They can also enter TSO commands when the command is prefixed with the word “TSO.”

CONSPROF The CONSPROF command allows users with CONSOLE command authority to modify the console profile to tailor message processing during an extended MCS console session. If you have RACF 1.9 or later installed and you have a TSO segment defined in the RACF database, the settings defined on the CONSPROF command are maintained from session to session.

The information in your console profile is used to control message processing during a console session. You can:

- Specify whether or not solicited messages that are routed to your console are to be displayed at the terminal.
- Specify whether or not unsolicited messages that are routed to your console are to be displayed at the terminal.
- Assign a value for the maximum number of solicited or unsolicited messages that are to be held for later retrieval with GETMSG service or REXX function.

Note: All TSO/E users that are going to use the CONSOLE and CONSPROF commands should have a RACF defined user profile (that is, do not use UADS to control TSO/E logons). For these TSO/E users, you can control the access to the CONSOLE and CONSPROF commands through RACF. See 4.4.3, “RACF Control for TSO/E Commands” on page 115.

4.4.1 CONSOLE TSO/E Command

The CONSOLE command establishes an extended MCS console session in a foreground or background TSO/E user’s address space.

The syntax of the CONSOLE command is:

```
CONSOLE     {ACTIVATE | DEACTIVATE}
            CART(cartvalue)
            NAME(console_name)
            SYSCMD(operator_command)
```

Where the operands are:

ACTIVATE Specifies that the user would like to start a console session. The session is established provided that the user has CONSOLE command authority.

DEACTIVATE Specifies that CONSOLE command processing is to be discontinued. The CONSOLE command terminates the console session. If the user is currently in CONSOLE prompt mode, the prompt is changed back to ready.

CART Specifies a command and response token (CART) that is passed with all system commands. The value specified may be used by the user to associate operator command responses (messages) with the respective commands. This value may be entered as character or hex data, up to a maximum of 8 bytes.

NAME Specifies the name that is to be assigned to the extended MCS console. The console session is established with the specified name. The name must be 2 to 8 characters. The NAME operand can be specified only when the console session is activated.

SYSCMD Specifies a operator command to be executed. The command may have parentheses in it as long as each open parenthesis has a matching closing parenthesis.

4.4.1.1 Activating a Console Session

To activate a console session, enter the CONSOLE command and specify the ACTIVATE operand. When the CONSOLE ACTIVATE command completes, you are returned to the prompt mode you were in (for example, TSO/E ready mode or ISPF panel display). You may enter operator commands by using the CONSOLE command with the SYSCMD operand.

You can also begin a console session without the ACTIVATE operand. In this case, the CONSOLE command puts you in “CONSOLE” prompt mode, and everything you enter on your terminal is processed as an operator command. Note that you must use the “TSO” subcommand to enter TSO commands under CONSOLE prompt mode. You are returned to your normal prompt mode when the END subcommand or a CONSOLE DEACTIVATE command is entered. Note that the END subcommand with no operands leaves the console session active, but the prompt mode is changed.

Note: When you activate a console session and do not specify a NAME operand on the CONSOLE command, the name assigned to your console is your TSO/E user ID.

4.4.1.2 Retrieving Messages

The CONSOLE command may also be used to change the CART that is associated with the operator commands. The CART is passed to console services along with each operator command. The MVS and subsystem command processors, in turn, should propagate the CART back with the command responses they may issue. The CART can then be used to selectively retrieve command responses. When you associate CART with the operator commands, you should also use the CONSPROF SOLDISPLAY(NO) UNSOLDISPLAY(NO) command to indicate that messages should be queued rather than displayed on your terminal. Messages that have been queued to your console may be obtained selectively through the REXX GETMSG function or the GETMSG programmable service. When you are running the console session in the CONSPROF SOLDISPLAY(YES) UNSOLDISPLAY(YES) mode, the CONSOLE command displays messages queued to your console as they arrive.

4.4.1.3 Terminating Console Session

You terminate the console session with the CONSOLE DEACTIVATE command or the END DEACTIVATE subcommand in the CONSOLE prompt mode. If the user issues the LOGOFF command before an active console session is deactivated, the console session is terminated at logoff time.

4.4.1.4 Console Command Subcommands

The CONSOLE command supports three subcommands:

CART Specifies the command and response token that is associated with operator commands entered. The cart value may be up to 8 bytes. If the value is longer than 8 bytes, the first 8 bytes are used.

```
CART cart_value
```

END Specifies that the user wishes to leave CONSOLE prompt mode. The optional DEACTIVATE (DEACT for short) operand specifies that CONSOLE command processing is to be terminated. If END is specified with no operands, the console session remains active, but the prompt mode is returned back to the previous mode.

```
END [DEACTIVATE]
```

TSO Specifies a TSO command to be issued from CONSOLE prompt mode. The command is processed and at its completion, the user remains in CONSOLE prompt mode.

```
TSO TSO_command
```

4.4.1.5 TSO/E User Exits

TSO/E provides exits you may use to customize the use and processing of the CONSOLE command. These exits include:

Pre-parse Controls what the user is allowed to specify on the CONSOLE command and in response to the CONSOLE command prompts.

It can also allow installation-specific keywords on the CONSOLE command. If installation-specific keywords are allowed, they should be removed before the exit returns control to the CONSOLE command.

Activation Establishes a communication area for related exits, change the initial settings specified by the user, and grant or deny the user CONSOLE command authority.

Deactivation Performs cleanup after other exits.

TSO/E does not provide default exit routines for any of the CONSOLE exits.

4.4.2 CONSPROF TSO/E Command

The syntax of the CONSPROF command is:

```
CONSPROF SOLDISPLAY( YES | NO ) UNSOLDISPLAY( YES | NO )
         SOLNUM(n)              UNSOLNUM(n)
```

Where the operands are:

SOLDISPLAY Indicates that messages that are responses to commands should be displayed at the user's terminal. If the user is going to use either the REXX GETMSG function or the GETMSG programmable service to retrieve command responses, this

value should be set to NO. If the user wishes to have messages displayed as they are routed to the user's console, this value should be set to YES. The default value is YES.

UNSOLDISPLAY Indicates that messages that are not responses to commands should be displayed at the user's terminal. If the user is going to use either the REXX GETMSG function or the GETMSG programmable service to retrieve messages that are not command responses, this value should be set to NO. If the user wishes to have messages displayed as they are routed to the user's console, this value should be set to YES. The default value is YES.

SOLNUM The number of command responses (messages) that can accumulate before the console command takes action. As messages are routed to the user's console, messages will begin to accumulate in the solicited message table if they are not displayed by the console command or retrieved by the GETMSG services.

UNSOLNUM The number of messages (not command responses) that can accumulate before the console command takes action. As messages are routed to the user's console, messages will begin to accumulate in the unsolicited message table if they are not displayed by the console command or retrieved by the GETMSG services.

Usually you issue the CONSPROF command to tailor your console profile before activating a console session. However, you can also use CONSPROF during a console session to change the profile settings.

The CONSPROF command also has its own exits. These exits include:

Initialization Controls what the user is allowed to specify on the CONSPROF command and grant or deny the user CONSOLE command authority for the duration of the CONSPROF command.

Pre-Display Exit You may add information to message IKJ55351I or issue an installation-defined message.

Termination Performs cleanup after other exits.

TSO/E does not provide default exit routines for any of the CONSPROF exits.

4.4.3 RACF Control for TSO/E Commands

You can grant users access to the CONSOLE command using one of the following methods:

- Use the RACF RDEFINE command to define CONSOLE as a RACF resource belonging to the TSOAUTH RACF class. Then grant the intended CONSOLE command users access to the CONSOLE resource using the RACF PERMIT command.

  Appendix J, "Authorizing TSO Users for the CONSOLE Command" on page 215 lists steps required to set up the RACF environment for the CONSOLE and CONSPROF commands, and extended MCS console authorization.

- When RACF controls cannot be used or are not available:

- Through logon pre-prompt exit IKJEFLD or IKJEFLD1. Use the logon pre-prompt exit routine to grant or deny CONSOLE and CONSPROF command authority by supplying user attributes that authorize the use of the commands.
- Through CONSOLE command exit IKJCNXAC and CONSPROF command exit IKJCNXCI. IKJCNXAC grants users CONSOLE command authority for the duration of the console session. IKJCNXCI grants users CONSOLE command authority for the duration of the CONSPROF command.

For more information, see *TSO/E Version 2 Customization* and *TSO/E Version 2 Command Reference*.

You must also specify the extended MCS console attributes for each TSO/E user having CONSOLE command authority. The console attributes control various functions, including the types of MVS commands a user can issue during an extended MCS console session, the routing of MVS messages and commands, and the display of MVS message formats. If your installation has RACF 1.9 or later installed, you can optionally define an OPERPARM segment with the console attributes in the user's RACF profile. If an OPERPARM segment has been defined for a user, the user's console attributes are saved from session to session.

4.4.4 Message Processing Defaults

The IKJTSOxx parmlib member defines some message processing defaults for the CONSOLE command and its services. IKJTSOxx contains a CONSOLE statement you use to specify an initial and maximum number of solicited and unsolicited messages to be routed to a user's console:

unsolicited Unsolicited messages to be routed to a user's console without being retrieved or displayed.

initial Specify an initial limit for the number of unsolicited messages.

When the actual number of queued messages reaches 80% of the specified number, installation exit IKJCNX50 is invoked. When the actual number of queued messages reaches the specified number, installation exit IKJCNX64 is invoked.

maximum Specify the maximum number of unsolicited messages.

This value is used by console activation processing and is passed to the TSO/E CONSOLE command message capacity exits (IKJCNX50 and IKJCNX64).

solicited Solicited messages to be routed to a user's console without being retrieved or displayed.

initial Specify an initial limit for the number of solicited messages.

When the actual number of queued messages reaches 80% of the specified number, installation exit IKJCNX50 is invoked. When the actual number of queued messages reaches the specified number, installation exit IKJCNX64 is invoked.

maximum Specify the maximum number of solicited messages.

This value is used by console activation processing and is passed to the TSO/E CONSOLE command message capacity exits (IKJCNX50 and IKJCNX64).

4.4.4.1 CONSOLE Command Message Capacity Exits

The 80% and 100% message capacity exits can be used to control the size of the message tables for the user. When the user's solicited or unsolicited message table becomes full, these exits can take the following actions so that messages are not lost:

- Make the message table larger if the maximum table size specified in the IKJTSOxx parmlib member is greater than the current maximum table size.
- Indicate that the user should have messages displayed at the terminal. This should begin to reduce the number of messages in the message table. After the table decreases to the resume percentage level specified in the parameter list, the processing of messages resumes as normal.
- Indicate that the console session should be terminated.
- Change the dispatching priorities of the tasks so that messages are retrieved faster.

Note: TSO/E does not have default 80% and 100% message capacity exits. It starts displaying queued messages at the terminal when the 100% limit is reached. This reduces the number of messages in the message table. After the table decreases to the resume percentage level specified in the parameter list, the processing of messages resumes as normal. When you specify a large dataspace buffer, it may take quite some time before the resume percentage level is reached and you will see a lot of messages displayed at your terminal.

4.5 REXX EXECs for Operator Tasks

The TSO/E REXX language allows REXX EXECs to perform operator tasks. TSO/E provides the CONSOLE host command environment that allows you to issue MVS and subsystem commands from a REXX EXEC. The CONSOLE host command environment is available only to REXX EXECs that run in a foreground or background TSO/E session. To use the CONSOLE environment, you must have CONSOLE command authority.

Before you can use the CONSOLE environment, you must first activate an extended MCS console session using the TSO/E CONSOLE command. After the console session is active, use ADDRESS CONSOLE to issue operator commands. This lets you issue MVS commands from an REXX EXEC without having to repeatedly issue the CONSOLE command with the SYSCMD operand. If you use ADDRESS CONSOLE and issue an operator command before activating a console session, the CONSOLE environment will not be able to process the command you issued.

The set of MVS system and subsystem commands you can use during a console session depends on the MVS command authority defined for your console.

You can use the CONSPROF command to control the processing of messages during a console session. Usually you issue the CONSPROF command to tailor a console profile before activating a console session. However, you can also use CONSPROF during a console session to change the profile settings.

Most often you request operator messages be queued (CONSPROF SOLDISPLAY(NO) UNSOLDISPLAY(NO)) when you use a REXX EXEC as an operator.

4.5.1 REXX EXECs and Retrieving Messages

A REXX operator uses the TSO/E external function GETMSG to retrieve messages. The GETMSG function can selectively retrieve solicited messages, unsolicited messages, and messages associated with a CART.

The syntax of the GETMSG function invocation is:

```
GETMSG(msgstem,msgtype,cart,mask,time)
```

Where:

msgstem The stem of the list of variables into which GETMSG places the message text.

msgtype The type (SOL, UNSOL, or EITHER) of message you want to retrieve. *msgtype* specification is optional.

cart The command and response token (CART). *Cart* specification is optional.

mask The search argument that GETMSG uses as a mask with the cart argument for obtaining a message. *Mask* specification is optional.

time The amount of time that GETMSG should wait before returning back to the caller if the requested message has not yet been routed to the user's console. *Time* specification is optional.

For a full description of the GETMSG function, see *TSO/E V2: REXX/MVS Reference*.

The GETMSG function retrieves messages that have been issued during your console session and stores them into variables. On the call to the GETMSG function, you specify the *msgstem* argument. GETMSG places each line of the message text into successive variables identified by the msgstem. Once the messages have been stored into variables, you can process the messages.

In addition to the variables into which GETMSG places the retrieved message, GETMSG sets other variables that contain additional information about the message. One set of variables relates to the entire message (that is, to all lines of message text that GETMSG retrieves), regardless of how many lines of text the message has. Another set variables is set for each message line. The names of these variables correspond to the field names in the message data block (MDB). The full syntax of the variable name is the msgstem followed by the name of the field in the MDB.

4.5.2 TSO/E SYSVAR Function

The TSO/E external function SYSVAR(*console session arguments*) lets you obtain information related to running an extended MCS console session that you have activated using the TSO/E CONSOLE command.

The SOLDISP, UNSDISP, SOLNUM, and UNSNUM arguments of the SYSVAR function provide information about the console profile that is in effect for the console session. The arguments relate to values set on the TSO/E CONSPROF command. You can use the arguments to determine what options are in effect

before you issue operator commands or use the GETMSG function to retrieve a message.

In addition, the MFTIME, MFOSNM, MFJOB, and MFSNMJBX arguments of the SYSVAR function provide information about messages that are issued during a console session. These arguments are useful if you use the GETMSG external function to retrieve queued messages and you want to display a particular message that was retrieved. The arguments indicate whether certain types of information should be displayed with the message, such as the time stamp.

Note: GETMSG only obtains system messages issued via the WTO service.

4.5.3 Sample EMCS Programs

Appendix G, “REXX EXEC for Extended MCS Console” on page 195 provides a sample program for an extended MCS console and includes an example of the ISPF panel you can use under your TSO session, as shown in Figure 31.

```
------------------------- LIST MCS COMMAND OUTPUT ----------------  ROW 1
COMMAND ===>                                                 SCROLL ===> HALF
MODE: BOTH    SC47
LONG COMMAND ===>
                                              12:32 92/01/16 ENTER '?' FOR HELP
DISPLAY ===>   TIME STAMP  N  SYSTEM NAME  N  JOB NAME   N  HOLD ===> N  (Y/N)
------------------------------------------------------------------------------
```

Figure 31. Sample ISPF Panel for Extended MCS Console

The ISPF panel has two input command locations following COMMAND and LONG COMMAND. SC47 is the system name where the console is active. Once the console is active, all messages and commands entered are part of a log that is kept for this active console.

The ISPF panel shown in Figure 31, has the following overtypeable fields:

TIME STAMP The N indicates that the timp stamp is removed from the message when displayed on the extended MCS console. Overtyping with a Y indicates to make the time stamp part of all messages displayed.

SYSTEM NAME The N indicates that the system name is removed from the message when displayed on the extended MCS console. Overtyping with a Y indicates to make the system name part of all messages displayed.

JOB NAME The N indicates that the job name is removed from the message when displayed on the extended MCS console. Overtyping with a Y indicates to make the job name part of all messages displayed.

Figure 32, shows a D C,CN=(ROGERS) command. This command is actually displaying the extended MCS console itself. Notice that the system name has been changed to be displayed as part of the messages. The message dataspace where all the messages are kept contains the complete message, which includes the time stamp, system name, and job name. Therefore, specifying Y for the

system name changes all displayed messages the were on the console. The console can be scrolled up and down by using PF7 and PF8 respectively.

```
------------------------ LIST MCS COMMAND OUTPUT -----------  ROW 9
COMMAND ===> D C,CN=(ROGERS)                      SCROLL ===> HALF
MODE: BOTH    SC47
LONG COMMAND ===>
                                            11:22 91/05/09 ENTER '?' FOR HELP
DISPLAY ===>   TIME STAMP  N  SYTEM NAME  Y  JOB NAME   N  HOLD ===>  N  (Y/N)
-------------------------------------------------------------------------
SC47      IEE889I 12.32.42 CONSOLE DISPLAY 524
SC47      MSG: CURR=0    LIM=1500 RPLY:CURR=0    LIM=999  SYS=SC47      PFK=00
SC47       CONSOLE/ALT        ID  --------------- SPECIFICATIONS ---------------
SC47       ROGERS            ---  COND=A      AUTH=MASTER       UD=Y
SC47        ROGERS                MFORM=T,S,J,X   LEVEL=ALL
SC47        SC47                  ROUTCDE=ALL
SC47                              CMDSYS=SC47
SC47                              MSCOPE=*ALL
SC47                              MONITOR=JOBNAMES,SESS
SC47     D C,CN=ROGERS
```

Figure 32. Sample Command on Extended MCS Console

4.5.3.1 Sample TSO/E Operator Console

A TSO/E Operator Presentation Sample (TOPS) is an ISPF dialog written in REXX. TOPS presents a convenient way to use the TSO/E console services and provides a useful dialog for MVS operations from TSO/E. TOPS consists of a series of panels and two REXX EXECs. All of them are distributed as members of SYS1.SAMPLIB. The member IEATOPSD contains the documentation to help you understand and use TOPS.

4.5.3.2 Sample REXX Extended MCS Console Program

The following example highlights the REXX extended MCS console features:

```
/* REXX Extended MCS Console Sample         */
Trace "O"
/* Save CONSPROF settings                   */
sd = sysvar("SOLDISP")
sn = sysvar("SOLNUM")
ud = sysvar("UNSDISP")
un = sysvar("UNSNUM")
/* Set ADDRESS to TSO                       */
Address "TSO"
/* Set new CONSPROF values                  */
"CONSPROF SOLDISP(NO) SOLNUM(500) UNSOLDISP(NO) UNSOLNUM(1500)"
/* Activate EMCS console                    */
"CONSOLE ACTIVATE"
/* Set ADDRESS to CONSOLE                   */
Address "CONSOLE"
/* Mainline loop                            */
Do forever
  Say "CNCMD?"  /* Prompt                   */
  Pull cmd      /* Read response            */
  cmd = translate(cmd) /* Case upper        */
  /* Parse response                         */
  Select
    When cmd = "" then nop   /* Nop?        */
    When cmd = "END" then do /* Exit?       */
      /* Set ADDRESS to TSO                 */
      Address "TSO"
      /* Deactivate EMCS console            */
      "CONSOLE DEACTIVATE"
      /* Reset CONSPROF values              */
      "CONSPROF SOLDISP("sd") SOLNUM("sn") UNSOLDISP("ud") UNSOLNUM("un")"
      Exit 0    /*  Bye Bye                 */
    End
    When word(cmd,1) = "TSO" then do /* TSO?*/
      Parse var cmd . cmd
      If cmd ¬= "" then address "TSO" cmd
    End
    Otherwise cmd                    /* CONS*/
  End           /* End Select               */
  /* Extract queued console messages        */
  GC = 0      /* Init loop control          */
  do while GC = 0       /* For all messages */
    GC = GETMSG("R.")
    select
      when GC = 0 THEN do /* Messages?      */
        /* Loop through every message line  */
        do i = 1 to r.0
          rsy ="" ; rts = "" ; rjo = ""
          /* Extract system name, time stamp
             and jobname from MDB when console
             (MFORM) wants them displayed   */
          If SYSVAR(MFOSNM) = "YES" then ,
            rsy = R.MDBGOSNM || " "
          If SYSVAR(MFTIME) = "YES" then ,
            rts = R.MDBGDSTP R.MDBGTIMH || R.MDBGTIMH || " "
          If SYSVAR(MFJOB)  = "YES" then ,
            rjo = R.MDBGJBNM || " "
          rmh = rsy || rts || rjo
          /* Do not show the message header
             for label, detail, and end line
             of MLWTO                       */
          If (R.MDBTLABT.i = "YES" | ,
              R.MDBTDATT.i = "YES" | ,
              R.MDBTENDT.i = "YES") & i <> 1 then ,
              rmh = overlay(" ",rmh,1,length(rmh))
          Say rmh || R.i /* Display msg     */
        End             /* End message line */
      End               /* End messages     */
      /* Bad RC from GETMSG                 */
      When GC > 4  then SAY "GETMSG rc =" GC
      otherwise nop     /* E.O.Messages     */
    End                 /* Select           */
  End                   /* End message loop */
End
```

Figure 33. Sample REXX Code for an Extended MCS Console

4.6 NetView Extended MCS Consoles

With the use of extended MCS consoles, NetView automation can interact with the MVS system as if the NetView operator is an MVS operator. EMCS consoles allow NetView to interact with MVS without some of the previous restrictions such as defining the consoles in the CONSOLxx parmlib member.

NetView consoles can use either the SSI interface or EMCS consoles to retrieve messages. A major difference between the SSI interface and EMCS consoles is who decides whether or not NetView, (and therefore the automation table) sees a message.

SSI NetView extracts automation messages from the SSI (see Figure 4 on page 26 and 2.3.3, “SSI Processing” on page 28). NetView queues them over to the main NetView address space.

EMCS MCS routes messages to NetView EMCS consoles that match an extended console′s routing attributes.

To request a message for NetView automation, set the AUTO keyword for that message to AUTO(YES) or AUTO(token) in the MPF table.

For suggested parameter settings for EMCS consoles, see *NetView Automation Planning*.

Chapter 5. MVS System Logger

The MVS system logger executes in its own MVS address space, as shown in Figure 34, and provides a set of system services that allow an application to:

- Connect to a log stream
- Write data to a log stream
- Browse data from a log stream
- Delete data from a log stream
- Disconnect from a log stream
- Maintain an inventory of log streams and their associated characteristics

The system logger is introduced with MVS/ESA Version 5.2. You can use system logger services to merge data from multiple instances of an application, including merging data from different systems across a sysplex. The task name is IXGLOGR.

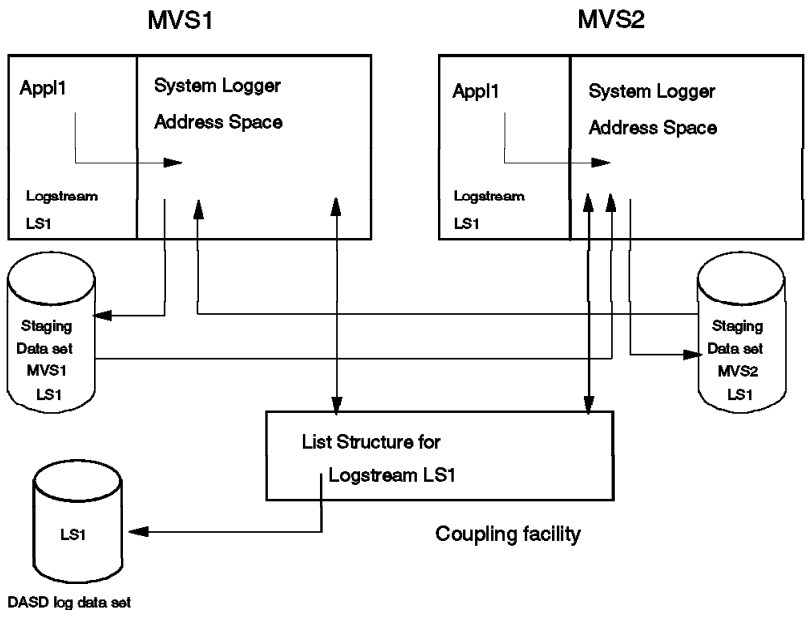


Figure 34. Overview of System Logger Processing

A log stream is a collection of one or more log records (also referred to as log blocks) written by an application using services provided by the MVS system logger. The application using MVS system logger services may or may not have multiple instances of itself executing in a sysplex. In the case of an application where each instance of the application writes log blocks to the same log stream, the result is a sysplex-wide merged log stream.

The MVS system logger uses a coupling facility DASD configuration. The coupling facility DASD configuration log stream consists of a structure resident in a coupling facility and one or more DASD data sets. When a log block is written to a coupling facility DASD configuration log stream, the log data is buffered in processor related storage and written to the coupling facility list structure. Once the log data is written to a local buffer and a coupling facility list structure, the writer is informed that the write request is complete.

As log data ages in the coupling facility, it is eventually migrated to a DASD data set so that coupling facility storage can be reclaimed to support incoming write requests. The migration process does not interfere with incoming write requests. Once data is successfully written to DASD, the log blocks are eligible for deletion from MVS system logger local buffers and the coupling facility structure.

When a DASD data set eventually becomes full, another data set is allocated and subsequent writes go to the new data set.

Operations Log (OPERLOG): Using the system logger, MVS provides a sysplex-wide consolidated SYSLOG called the operations log (OPERLOG).

The MVS operations log (OPERLOG) function of the MVS console services provides a sysplex-wide merged and chronologically ordered message log by using the MVS system logger services. The messages are logged in the form of *message data blocks* (MDB). An MDB is a structured control block that contains a complete representation of a message, including both text and control information.

MCS logs the hardcopy message set into the OPERLOG in an MDB format, SYSLOG (MVS record format), or both. The OPERLOG is maintained by the MVS system logger. To define the hardcopy specifications for the sysplex, see 3.2.4, “HARDCOPY Statement” on page 78 and 5.3, “Defining the Operations Log” on page 127.

5.1 Defining the System Logger

The MVS system logger maintains log stream information in a *LOGR inventory couple data set*. The LOGR couple data set is defined and formatted with the cross-system coupling facility (XCF) IXCL1DSU utility program. The following example shows utility control statements for the IXCL1DSU program.

```
DEFINEDS SYSPLEX(WTSCPLX1)
         MAXSYSTEM(16)
         DSN(SYS1.XCF.LOGR10) VOLSER(TOTCAT)
         CATALOG
    DATA TYPE(LOGR)
         ITEM NAME(LSR) NUMBER(20)
         ITEM NAME(LSTRR) NUMBER(20)
```

Figure 35. LOGR Couple Data Set Definition

ITEM NAME(LSR) NUMBER( ) Specifies the maximum number of log streams that can be defined to the LOGR inventory couple data set

ITEM NAME(LSTRR) NUMBER( ) Specifies the maximum number of structure names that can be defined to the LOGR inventory couple data set

Coupling facility resource management (CFRM) provides the XCF services that manage all coupling facility resources (coupling facility structure sizes and the allocation rules). This management includes the enforcement of CFRM policies

to ensure that the coupling facility and structure requirements are defined properly in each policy.

When you plan to activate system logger, your XCF CFRM policy must be updated to include the definitions for system logger structures. The CFRM policy defines the structures: the amount of coupling facility storage to be used for each structure, an ordered preference list of coupling facilities in which each structure should reside, and an unordered exclusion list of structure names that should not be allocated in the same coupling facility as the specified structure.

You use the IXCMIAPU utility to create or update the CFRM administrative data. The following example shows definitions for a dedicated OPERLOG structure:

```
DATA TYPE(CFRM) REPORT(YES)
DEFINE POLICY NAME(CFRM05) REPLACE(YES)
     ::: other structure definitions :::
     STRUCTURE NAME(SYSTEM_OPERLOG)
            SIZE(1024)
            PREFLIST(CF02,CF01)
     ::: other structure definitions :::
```

Figure 36. Updating CFRM Policy for OPERLOG

The SIZE(1024) of the OPERLOG structure SYSTEM_OPERLOG specifies the maximum amount of space to be allocated for the structure in the coupling facility. The number is specified in units of 1K (1024 bytes.) Given the above size definitions, the OPERLOG structure holds roughly 6000 hardcopy log MDBs, which in turn is in average 11,500 *SYSLOG format lines*. Note that the numbers may vary from installation to installation.

Once the definitions are in place, use the following operator command to activate the new policy:

```
SETXCF  START,POLICY,TYPE=CFRM,POLNAME=CFRM05
```

For more information, see *MVS/ESA Setting Up a Sysplex*.

5.2 Defining the Operations Log Stream

Use the IXCMIAPU utility to create or update the LOGR inventory couple data set log stream definitions. Figure 37 on page 126 shows the IXCMIAPU utility program report for the definitions of a *dedicated* OPERLOG log stream.

```
LSR (Log Stream)                    20              2

LSTRR (Structure)                   20              2

   LOGSTREAM NAME(SYSPLEX.OPERLOG) STRUCTNAME(SYSTEM_OPERLOG)
             LS_DATACLAS(SHARE33)
             LS_MGMTCLAS() LS_STORCLAS() HLQ(IXGLOGR) MODEL(NO)
             LS_SIZE(512)
             STG_MGMTCLAS() STG_STORCLAS() STG_DATACLAS() STG_SIZE(0)
             LOWOFFLOAD(0) HIGHOFFLOAD(80) STG_DUPLEX(NO) DUPLEXMODE()

   STRUCTURE NAME(SYSTEM_OPERLOG) LOGSNUM(1)
             MAXBUFSIZE(65532) AVGBUFSIZE(400)
             LOGSTREAMS CURRENTLY DEFINED TO THIS STRUCTURE(1)
```

Figure 37. LOGR Couple Data Set Log Stream Definitions

The following rules apply to the operations log definitions:

- The OPERLOG log stream NAME must be SYSPLEX.OPERLOG.
- The log stream STRUCTNAME must match the NAME in the STRUCTURE definition both in the LOGR and CFRM specifications.
- The log stream LS_DATACLAS must specify a SMS data class that assigns VSAM share option 3,3 to the log stream DASD data sets. The LS_SIZE specifies the size, in 4K blocks, of the log stream DASD data sets for the log stream.

  Note: The SMS subsystem must be active and capable of processing the DATACLAS definition on the dynamic allocation (SVC 99) requests issued by the system logger for the DASD log stream DASD data sets.

  Note: The log stream DASD data sets do not have to be on SMS managed volumes.

  IBM recommends that you set up SMS values for the log stream data sets in advance and simply omit the LS_SIZE parameter to use the default. If you omit LS_SIZE, or specify a field of zeros, system logger does one of the following when allocating space for staging data sets:

  1. Uses the LS_SIZE of the log stream specified on the LIKE parameter if specified.
  2. Uses the size defined in the SMS data class for the log stream data sets.
  3. Uses dynamic allocation rules for allocating data sets if SMS is not available.

  If you specify an explicit value for LS_SIZE, this value overrides any size characteristics defined in the SMS data class defined for the DASD staging data set for this log stream.
- On the structure definition, LOGSNUM specifies the number of log streams that you want to allocate to the same coupling facility structure. The value you specify determines how many pieces the structure is divided into and how much room there is in each piece. In the example the LOGSNUM is set to one that dedicates the structure for OPERLOG use only.
- On the structure definition, AVGBUFSIZE specifies the average size, in bytes, of individual log blocks (MDBs) that can be written to log streams allocated

to the coupling facility. A good starting estimate for the hardcopy message set MDB size is 350 bytes.

- The HLQ specifies the high level qualifier for both the log stream data set name (and the staging data set name if used). If you do not specify a high level qualifier, a high level qualifier IXGLOGR is defaulted. If you also specified the LIKE parameter, it will have the high level qualifier of the log stream specified on the LIKE parameter.

  The OPERLOG log stream data sets names have the following format:

  ```
  hlq.SYSPLEX.OPERLOG.Annnnnnn
  where nnnnnnn is a running sequence number
  ```

  You can determine how many OPERLOG log stream data sets are currently allocated by listing the catalog for the data set name level "hlq.SYSPLEX.OPERLOG."

  Note: System logger maintains for each log stream an inventory of log stream DASD data sets in the LOGR inventory couple data set. This inventory can hold up to 168 log stream DASD data sets for each log stream. It is strongly advised to use the IBM provided IEAMDGLG sample program (source in SYS1.SAMPLIB) to regularly copy the OPERLOG log stream data into archive data sets and delete the copied records from the log stream to *avoid log stream full conditions*.

Once the OPERLOG is defined into the LOGR inventory couple data set, issue the following command to dynamically activate it:

```
SETXCF COUPLE,TYPE=LOGR,PCOUPLE=couple.data.set.name
```

For subsequent IPLs, the COUPLExx parmlib member should also be updated to include the LOGR inventory couple data set.

5.2.1 IEAMBDLG Sample Program

SYS1.SAMPLIB contains a sample program, IEAMDGLG, to read records from an OPERLOG stream and convert them to the SYSLOG format. The program is an example of how to use the MVS system logger services to retrieve and delete records from the OPERLOG stream. The program reads the records created in a given time span, converts them from the MDB format to hardcopy log format (the SYSLOG format), and writes the the output to a data set. It also has an option to delete from the stream all the records created prior to a given date.

The IEAMDBBR program uses the IXGCONN, IXGBRWSE, and IXGDELET system logger macro services, rather than the LOGR subsystem, to extract and manipulate the OPERLOG log stream data.

Note: This sample program is important in that it is the only utility provided to clear out OPERLOG records.

5.3 Defining the Operations Log

The OPERLOG can be defined in the CONSOLxx parmlib member as follows:

```
HARDCOPY  DEVNUM  {(devnum)         },
                  {(SYSLOG)         },
                  {(OPERLOG)        },
                  {(devnum,OPERLOG)},
                  {(SYSLOG,OPERLOG)}
```

If OPERLOG is not defined in parmlib, it can be activated with the following operator command, providing the MVS system logger is active:

```
V OPERLOG,HARDCPY
 IEE889I 20.59.31 CONSOLE DISPLAY 153
 MSG: CURR=10   LIM=1500 RPLY:CURR=6    LIM=999  SYS=SC50      PFK=00
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS --------------
  SYSLOG                 COND=H      AUTH=CMDS         NBUF=2    UD=Y
                         ROUTCDE=ALL
  OPERLOG                COND=H      AUTH=CMDS         NBUF=N/A  UD=Y
                         ROUTCDE=ALL
```

The intended scope of an OPERLOG is a sysplex; however, a given system in the sysplex may or may not be using OPERLOG at any particular time. The operator commands that control the status of OPERLOG and the initialization parameter that activates it at IPL have a single system scope. Furthermore, a failure in OPERLOG processing on one system does not have any direct affect on the other systems. The result is that an OPERLOG may contain records from an entire sysplex or from only a subset of the systems, depending on the installation's requirements and on the environmental factors.

Note: You can partially control the definition of the hardcopy message set through the HARDCOPY statement on CONSOLxx parmlib member and the following command:

```
VARY ,HARDCPY,CMDS|,NOCMDS|,STCMDS|,INCMDS
```

Changes to the hardcopy message set affect the OPERLOG as well as SYSLOG.

The CONSOLxx parmlib member HARDCOPY statement specifies whether the hardcopy medium active at initialization is an MCS printer, SYSLOG, or OPERLOG.

Note: The DEVNUM parameter can specify both SYSLOG and OPERLOG, or devnum and OPERLOG subparameters if you want to activate your previous hardcopy medium and the OPERLOG.

5.4 Hardcopy Log

Hardcopy processing creates a permanent record of operator command and system message activity. Hardcopy logging records system messages and, optionally, commands, in the system log (SYSLOG), the operations log (OPERLOG), or an MCS printer (collectively called the hardcopy medium). The group of messages and commands that are recorded to the hardcopy medium is called the hardcopy message set.

The hardcopy log on the SYSLOG or the MCS printer is maintained separately for each system in a sysplex; thus it provides a system scope view of the hardcopy message set. The OPERLOG is a log medium that uses the *system logger* to record and merge hardcopy message sets from the systems in a sysplex that have activated the OPERLOG recording. The OPERLOG can be active simultaneously with the SYSLOG or the MCS printer.

Note: An extended MCS console may be activated to receive the hardcopy message set from one or more systems in a sysplex. However, this does not negate the MCS requirement for the hardcopy medium.

Figure 38 on page 129 shows an overview of the communications task OPERLOG and SYSLOG processing.

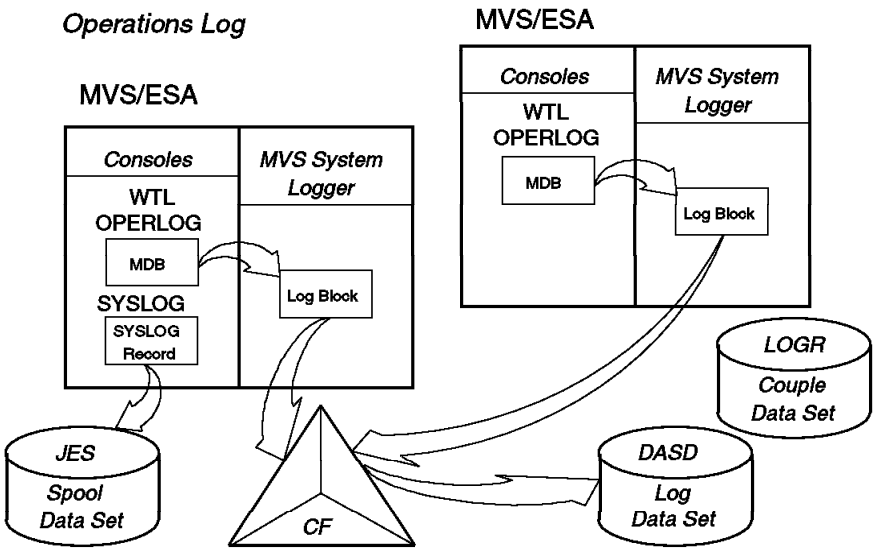


Figure 38. Overview of OPERLOG and SYSLOG Processing

The HARDCOPY statement in the CONSOLxx parmlib member identifies the hardcopy medium, defines the hardcopy message set, specifies the name of the alternate console group, and specifies whether UD messages not sent to any other console will be delivered to the system console, depending on the options specified.

Note: A display console cannot be the hardcopy medium. The printer device you specify for the hardcopy medium must also be defined as a console in CONSOLxx.

The system defaults to SYSLOG as the hardcopy medium in the following cases:

- You do not code a HARDCOPY statement.
- You specify an unusable hardcopy console.
- You specify OPERLOG alone but the operations log is unusable.

A hardcopy medium is required to be active in systems with at least one active display console or more than one active non-display console. Hardcopy processing is also required when MPF is used to suppress messages. Suppressed messages appear only in the hardcopy medium.

5.4.1 Hardcopy Message Set

The hardcopy message set represents messages that are recorded in hardcopy medium and includes messages with one or more of the following characteristics. Unsolicited messages in the hardcopy message set which are always included:

- 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

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

You define criteria for messages in the hardcopy message set at system initialization with the HARDCOPY statement in the CONSOLxx parmlib member:

- For commands and command responses, the CMDLEVEL option of the HARDCOPY statement controls the types of commands included in the hardcopy message set.
- For unsolicited system messages, the ROUTCODE option of the HARDCOPY statement controls the routing codes the system uses to select messages for the hardcopy message set. If an option is not specified, the default value is used for the hardcopy message set definition.

Once MVS has been initialized, you can modify the criteria of the hardcopy message set using the VARY HARDCPY command.

Unless you specify otherwise, the hardcopy message set includes all messages, except those that are explicitly omitted through the WTO macro (no hardcopy) or installation exits. You can request through the ROUTCODE parameter on the HARDCOPY statement that the hardcopy message set include only messages with certain routing codes. At system initialization, MCS sets up a minimum set of routing codes (1,2,3,4,7,8,10, and 42), in addition to any others specified for the hardcopy message set. If you attempt to eliminate any of these, the system includes messages with these routing codes in the hardcopy message set anyway.

To learn about the kinds of messages that the system includes in the hardcopy message set, but does not send to any console, use the DISPLAY CONSOLES,HCONLY command.

For more information, see *MVS/ESA Initialization and Tuning Reference* and *MVS/ESA System Commands*.

5.4.2 Hardcopy Mediums

The initial hardcopy medium at system initialization can be an MCS printer, the SYSLOG, or the OPERLOG. The medium is selected based on the the DEVNUM, UD, and HCPYGRP keywords on the HARDCOPY statement in the CONSOLxx parmlib member. Once the system has been initialized, operators can use the VARY HARDCPY command from MCS or extended MCS consoles with master authority to redefine the hardcopy medium.

The D C,HARDCOPY command displays information about the current hardcopy medium. The command output tells you whether the hardcopy medium is SYSLOG, OPERLOG or a device, the criteria that have been defined by the installation for selecting messages for the hardcopy message set, and the number of messages waiting to be placed on the hardcopy medium.

5.4.2.1 System Log

If the SYSLOG is initially specified as the hardcopy medium, it is started by MCS after the primary JES is active. First, the SYSLOG task requests the primary JES to assign a job ID for the SYSLOG job and then it allocates a SYSOUT data set to hold the hardcopy message set.

The LOGLIM parameter on the INIT statement in the CONSOLxx parmlib member specifies the maximum number of buffers that the system can use to process

write-to-log (WTL) messages. Each buffer is 140 bytes, and is obtained from the extended common service area (ECSA).

You can dynamically change the number of allocated WTL buffers with the CONTROL M,LOGLIM command.

The LOGLMT parameter in the IEASYSxx parmlib member specifies the maximum number of WTL messages allowed for each log data set. The value is used by log processing to determine when a log data set should be scheduled for SYSOUT processing by JES. When the value is reached, log processing issues an internal WRITELOG command to close and free the current log data set, and to allocate and open a new log data set. The LOGCLS parameter in the IEASYSxx parmlib member specifies the JES SYSOUT class for the log data sets.

You can force the system log data set to be queued for printing before the threshold is reached by issuing the WRITELOG command.

If the system log is defined as the hardcopy medium and SYSLOG fails (OPERLOG is not active), the system attempts to switch hardcopy processing to an appropriate printer console. If a suitable console is not active at the time of failure, hardcopy processing is suspended and you are notified through the master console.

You can dynamically switch the hardcopy medium by using the VARY HARDCPY command. If OPERLOG is not active, you can activate it in addition to your existing hardcopy medium. Once OPERLOG is active, you can deactivate the old hardcopy medium.

For example, if the current hardcopy medium is SYSLOG and you want to stop the SYSLOG task and resume OPERLOG as the hardcopy medium, you enter the following commands:

```
D C,HC
 IEE889I 16.04.27 CONSOLE DISPLAY 304
 MSG: CURR=4    LIM=1500 RPLY:CURR=4    LIM=999  SYS=SC43      PFK=00
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS ---------------
  SYSLOG                 COND=H      AUTH=CMDS         NBUF=0    UD=Y
                         ROUTCDE=ALL
```

```
V OPERLOG,HARDCPY
 IEA630I  OPERATOR *OPLOG03 NOW ACTIVE,   SYSTEM=SC43    , LU=NONE
 D C,HC,L=VAINI
 IEE889I 16.04.35 CONSOLE DISPLAY 308
 MSG: CURR=4    LIM=1500 RPLY:CURR=4    LIM=999  SYS=SC43      PFK=00
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS ---------------
  SYSLOG                 COND=H      AUTH=CMDS         NBUF=0    UD=Y
                         ROUTCDE=ALL
  OPERLOG                COND=H      AUTH=CMDS         NBUF=N/A  UD=Y
                         ROUTCDE=ALL
```

The operations log is operationally independent of the SYSLOG. 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 once the operations log is started

with the SYSLOG active, you can enter the VARY SYSLOG,HARDCPY,OFF and the WRITELOG CLOSE commands.

```
V SYSLOG,HARDCPY,OFF
 IEE338I SYSLOG   INACTIVE AS HARDCPY
WRITELOG CLOSE
 IEF196I IEF285I   +MASTER+.SYSLOG.JOB00112.D0000009.?          SYSOUT
 IEE043I A SYSTEM LOG DATA SET HAS BEEN QUEUED TO SYSOUT CLASS Y
 IEE037D LOG NOT ACTIVE
D C,HC
 IEE889I 16.05.47 CONSOLE DISPLAY 316
 MSG: CURR=4    LIM=1500 RPLY:CURR=4    LIM=999  SYS=SC43      PFK=00
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS ---------------
  OPERLOG                COND=H      AUTH=CMDS         NBUF=N/A  UD=Y
                         ROUTCDE=ALL
```

If you now want to reactivate SYSLOG as the hardcopy medium, enter the following commands:

```
WRITELOG START
  IAT6100 ( DEMSEL ) JOB SYSLOG   (JOB00113), PRTY=15, ID=+MASTER+
  IEF196I IEF237I JES3 ALLOCATED TO SYSLOG08
  IEE041I THE SYSTEM LOG IS NOW ACTIVE
V SYSLOG,HARDCPY
  IEE889I 16.06.31 CONSOLE DISPLAY 324
  MSG: CURR=5    LIM=1500 RPLY:CURR=4    LIM=999  SYS=SC43      PFK=00
   CONSOLE/ALT        ID  --------------- SPECIFICATIONS ---------------
   SYSLOG                 COND=H      AUTH=CMDS         NBUF=0    UD=Y
                          ROUTCDE=ALL
   OPERLOG                COND=H      AUTH=CMDS         NBUF=N/A  UD=Y
                          ROUTCDE=ALL
```

5.4.2.2 Operations Log

The OPERLOG as a hardcopy medium is maintained by the system logger. The OPERLOG contains merged hardcopy message sets from each system in a sysplex that has the OPERLOG active and provides more data than is recorded in the SYSLOG.

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 does 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.

The OPERLOG function internally activates an extended MCS console to receive messages for transcription to the log when activated. Each system that is using the OPERLOG has an active console whose name is *OPLOGxx, where xx is the XCF system slot number of that system:

```
D C,KEY=NONE
 IEE892I 09.36.23 CONSOLE DISPLAY 916
 MSG: CURR=7    LIM=1500  RPLY: CURR=6    LIM=999  KEY=NONE
   NAME      NAME      NAME      NAME      NAME      NAME      NAME
 *OPLOG04  *OPLOG03  *OPLOG02  *OPLOG01  *OPLOG06  *OPLOG05
```

5.5 Activating the LOGR Subsystem

Before you can access the data through the LOGR subsystem, you must activate it. You can start the LOGR subsystem using one of the following methods:

- In the IEFSSNxx parmlib member, add the following:

  SUBSYS SUBNAME(LOGR) INITRTN(IXGSSINT)

  The LOGR subsystem will be active after next IPL.

- Use the SETSSI command to add the subsystem dynamically:

  SETSSI ADD,SUBNAME=LOGR,INITRTN=IXGSSINT

5.5.1 Accessing Log Stream Data

You can use the LOGR subsystem to access log stream data as a sequential data set from any program that uses only BSAM or QSAM access and uses only the following macros to access data:

- DCB
- OPEN
- GET (QSAM)
- READ (BSAM)
- SYNAD
- SYNADAF
- CLOSE

Your program is *not* eligible for use with the LOGR subsystem if it uses the NOTE, POINT, and CNTRL macros.

Once the LOGR subsystem (and OPERLOG) is active, you could, for example, copy the MDB format hardcopy message set from the log stream to a sequential data set using the following JCL:

```
//VAINIJ JOB (999,POK),EXPERT,MSGLEVEL=1,MSGCLASS=X,
//  CLASS=A,NOTIFY=&SYSUID
//S1       EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=X
//SYSUT2   DD DSN=VAINI.MDB.SYSLOG,DISP=SHR
//SYSUT1   DD DSN=SYSPLEX.OPERLOG,
// SUBSYS=(LOGR,IXGSEXIT,'FROM=(1995/077)'),
// RECFM=VB,BLKSIZE=12292,LRECL=4096,DSORG=PS
//SYSIN    DD DUMMY
```

Figure 39. Sample JCL for Log Stream Copy

The log stream allocation, shown in Figure 39, in the SUBSYS JCL option is defined as follows:

```
//ddname   DD  DSNAME=log.stream.name,
//         SUBSYS=(LOGR[,exit_routine_name]
//         [,'SUBSYS-options1'][,'SUBSYS-options2'])
```

Where:

exit_routine_name Specifies the name of the exit routine to receive control from the LOGR subsystem. If the exit_routine_name is not specified, the default log stream subsystem exit routine, IXGSEXIT, will be used.

SUBSYS-options1

```
[FROM={({[yyyy/ddd][,hh:mm[:ss]]}) | OLDEST}]
[TO={({[yyyy/ddd][,hh:mm[:ss]]}) | YOUNGEST}]
[,DURATION=(nnnn,HOURS)]
[,GMT|LOCAL]
```

SUBSYS-options2 Defined by the log stream owner

OPERLOG does not provide its own LOGR subsystem exit. The default exit IXGSEXIT should be used to extract the hardcopy message set MDBs from the log stream. IBM provides an IEAMDGLG sample program that may be used to extract *SYSLOG format* data from the OPERLOG log stream. See 5.2.1, “IEAMBDLG Sample Program” on page 127.

Access to the log stream data can be protected by the RACF LOGSTRM class with a resource name RESOURCE(log_stream_name).

For more information about the LOGR subsystem, see *MVS/ESA Programming: Assembler Services Guide*.

Chapter 6. Program to Operator/System Communication

This chapter addresses three distinct types of communication which are:

- Program to operator communication through the use of the WTO, WTOR, DOM, LOADWAIT, and WTL macros
- Operator to program communication through the use of QEDIT and EXTRACT macros
- Program/system communication through the use of the MCSOPER, MCSOPMSG, MGCRE, CONVCON, and CPF macros

The following IBM publications provide additional information on subjects discussed in this chapter: *MVS/ESA Programming: Authorized Assembler Services Reference, Volume 1 (ALESERV-DYNALLOC), MVS/ESA Programming: Authorized Assembler Services Reference, Volume 2 (ENFREQ-ITTFMTB), MVS/ESA Programming: Authorized Assembler Services Reference, Volume 3 (LLACOPY-SDUMPX), MVS/ESA Programming: Authorized Assembler Services Reference, Volume 4 (SETFRR-WTOR), MVS/ESA Programming: Assembler Services Guide, MVS/ESA Programming: Assembler Services Reference, MVS/ESA Programming: Authorized Assembler Services Guide, MVS/ESA Initialization and Tuning Reference, and MVS/ESA Installation Exits*.

6.1 Program to Operator Communication

The WTO and WTOR macro (SVC 35) services allow you to write a message to a display device, a printer, a hardcopy log, or a program that receives WTO and WTOR messages (such as an EMCS console or a subsystem console). Besides writing a message, WTOR allows you to request a reply from the operator who receives the message. The DOM macro (SVC 87) service allows you to delete a message that has been previously issued.

6.1.1 WTO and WTOR Macro Services

WTO or WTOR messages are routed to consoles by specifying one or more of the following:

ROUTCDE ROUTCDE routes messages by the specified routing codes. The routing codes determine which console or consoles receive the message. The message is displayed on all consoles in the sysplex that are defined to receive one or more routing codes specified for the message, and providing that the console's MSCOPE permits it. WTO and WTOR allow routing codes from 1 to 128. Routing codes 29 through 41 are reserved, and are ignored if specified. Routing codes 42 through 128 are available to authorized programs only, although the ROUTCDE parameter itself is available to non-authorized as well as authorized users.

CONSID

CONSNAME CONSID or CONSNAME can be used to route messages to a specific console. These mutually exclusive parameters let you specify the ID or the name of the console that is to receive the message. When you issue a WTO or WTOR macro that uses either the CONSID or CONSNAME parameter with the ROUTCDE parameter, the message or messages goes to the console

specified by the CONSID or CONSNAME and all of the consoles receiving the specified routing codes.

MSGTYP MSGTYP can be used to route messages by message type. The MSGTYP parameter is typically used for messages related to the MONITOR command. It specifies how the message is to be routed to consoles on which the MONITOR command is active. Specifically,

JOBNAMES MSGTYP (JOBNAMES) routes messages to consoles that have activated MONITOR JOBNAMES.

SESS MSGTYP (SESS) routes messages to consoles that have activated MONITOR SESS.

STATUS MSGTYP (STATUS) routes messages to consoles that have activated MONITOR STATUS.

MCSFLAG MCSFLAG routes messages by special attribute.

The MCSFLAG parameter is used to specify various attributes of the message, such as whether the message is:

REG0/QREG0 For a particular console by a 1-byte console ID. MVS/ESA SP Version 4 introduced the 4-byte console ID. Existing WTOs may be using register zero interface (REG0 or QREG0). If these keywords are specified, the console ID is placed in the low-order byte of register zero before the WTO is issued. Only a 1-byte console ID can be specified this way, so this code must be converted to use CONSID=.

Note: The high-order 3 bytes of register zero are assumed to be a CONNECT ID if this is a multiline message.

BRDCST For all active consoles

RESP An immediate command response. You may also specify CART parameter to associate a command response with a command.

Note: Use of descriptor code 5, rather than MCSFLAG=RESP, is the recommended way to indicate a command response. MPF exits can modify the descriptor codes, but not the MCSFLAG=RESP setting.

Note: If you are using message suppression through the MPFLSTxx parmlib member, and the message is a command response (DESC=5 or MCSFLAG=RESP), then you need SUP(ALL) to suppress it.

BRDCPY For the hardcopy log

NOCPY Not for the hardcopy log

BUSYEXIT You may also specify MCSFLAG=BUSYEXIT which terminates the WTO if there are no console buffers. The return information for a terminated WTO indicates the severity of the console buffer

shortage. If you do not specify BUSYEXIT, WTO processing may place the task that is invoking WTO in a wait state until WTO buffers are again available.

6.1.2 Issuing Command Responses Using WTO

The WTO macro is used to issue responses to commands and several considerations or rules should be followed:

DESC=5 Specify the descriptor code 5 to identify the message as a command response. Additional descriptor codes can also be used if desired.

MCSFLAG=RESP can also be used instead of DESC=5 to indicate a command response. DESC=5 is preferred for maximum flexibility because descriptor codes can be changed by an installation's message exits, but MCSFLAGs cannot.

CONSID=

CONSNAME= Direct the message to the console that issued the command by specifying the 4-byte console ID or console name of the console that issued the command.

CART= Use the CART value passed with the input command. The CART data is eight bytes.

It is possible to respond with one message, which may be many lines of text. You can issue a multiple line message either with all lines in one WTO invocation, or by using a connect loop by using the CONNECT parameter. Always issue an end line. Connecting lines do not need any other parameters as they inherit all message delivery attributes from the first line.

Note: Be sure to clear register zero before issuing a WTO for a line (or block of lines) of a multiple-line message. If a non-zero value is found in register zero, it assumes you are trying to CONNECT to a message with that message ID.

Note: As a coding consideration, if you are issuing several messages (instead of one multiline) in response to a command, do not change this as you may cause an incompatability for any automation programs that were analyzing the messages. Any new command responses should be designed as a single multiline response.

6.1.2.1 Command Response Data Areas

The location of the CART and CONSID depends on where the processing program receives its input command. You can find the console ID and CART information in the same control block as the command text.

Command Input Buffer (CIB): A program must first obtain a pointer to the communications ECB and a pointer to the first command input buffer (CIB) on the CIB chain for the task. Issue the following macro:

```
EXTRACT answer_area,FIELDS=COMM
```

Returned is the address of the command scheduler communications list (mapped by IEZCOM). The list consists of a pointer to:

- The communications event control block
- A pointer to the command input buffer (CIB) - field COMCIBPT

- A token

  Note: If a token exists, the high-order bit of the token field is set to one. The token is used only with internal START commands.

The CIB (mapped by IEZCIB) contains the information specified on the STOP, START, or MODIFY command. If the job was started from the console, initially the CIB pointer points to the START CIB. If the job was not started from the console, the CIB pointer is zero. The CIB data area is followed by the CIB extension, which includes pointer to:

- The UTOKEN of the console
- The console′s command authority
- The console name
- The CART (CIBXCART)
- The 4-byte console ID (CIBXCNID)

The CIB extension data should be used to verify the authority of the console entering the command and to send command responses back to the originating console. The CIBX (CIB extension) is new with MVS/ESA SP Version 4 and can be found by adding the CIBXOFF offset value to the address of the CIB. For an example, see Figure 40 on page 139.

All programs that use the MGCRE macro to issue commands can specify a CART to identify which command this is a response to. If the command issuer does not specify a CART= on the WTO macro, the CART field in the command input control block retrieves a field of zeroes, which is acceptable.

Note: An example program in SYS1.SAMPLIB in member IEAEXMCS shows how to use an EMCS console programming interface to:

- Activate a console
- Deactivate a console
- Receive messages
- Process MDB objects
- Listen for console alerts

This example also illustrates the use of some other MVS operations facilities including the MODIFY/STOP interface (using EXTRACT and QEDIT), CONVCON to see if a console is active, MGCRE to issue system commands, WTO using the TEXT key, and DOM to delete a held message.

Command Exit Routines: The CMDX control block (mapped by IEZVX101) is the input to the command exit routines. The console ID and CART can be found as follows:

- 4-byte console ID (CMDXC4ID)
- CART (CMDXCART)

Commands in SSI Routines: The SSCM control block (mapped by IEFSSCM) is the input to the command subsystem interface (SSI) routines. The console ID and CART can be found as follows:

- 4 byte console ID (SSCMCNID)
- CART (SSCMCART)

6.1.2.2 WTO Command Response Example with CIB Control Block

This is a simple example of a WTO, where the text is not variable. It is also not reentrant code.

```
    Assume the address of the CIB is in R7
    --------------------------------------
       USING   CIB,R7             CIB based on R7
       LR      R8,R7              Set up R8 as address of CIBX
       AH      R8,CIBXOFF         CIBX=addr(CIB)+CIBXOFF
       USING   CIBX,R8            Addressability to CIBX
       WTO     'ABC123I XYZ COMMAND COMPLETED SUCCESSFULLY',          X
               DESC=5,                                                X
               CONSID=CIBXCNID,                                       X
               CART=CIBXCART
       DROP    R8                 done with CIBX
       .
CIB    DSECT
       IEZCIB
       .
+CIBXCART DS CL8 - COMMAND AND RESPONSE TOKEN
+CIBXCNID DS F   - CONSOLE ID
       .
```

Figure 40. WTO Example Using CIB to Obtain CART and CONSID

Assembler Example with CMDX Control Block (multi-line WTO): This is an example of a multiple-line WTO using CONNECT. The text is variable. This code is reentrant if the static data has been copied into a GETMAINed area.

```
      .
      .
* Access to CMDX parameter list
       L       R8,0(R1)           save address of CMDX
       USING   CMDX,R8            Access the CMDX
       .
       .
       .
* Put the first line of text into the TEXT field
* Set the length of the first line
        LA    R4,L'first line of text
        STH   R4,TEXTLEN
* Copy actual first line into text field
        MVC   TEXTTEXT(L'first line of text),first line of text
* Clear register 0 so it will not look like a CONNECT id
        XR    R0,R0
* Issue WTO for the first line
        WTO   TEXT=((TEXTADDR,,)),                                   X
              CONSID=CMDXC4ID,                                       X
              CART=CMDXCART,                                         X
              DESC=5,                                                X
              MF=(E,LINE1)
* Save the connect id returned from the first line WTO
        ST    R1,MSGID
*
* Repeat the following block until finished putting out all the lines
*
* Put the next line of text into the TEXT field
* Set the length of the next line
        LA    R4,L'next line of text
        STH   R4,TEXTLEN
* Copy actual next line into text field
        MVC   TEXTTEXT(L'next line of text),next line of text
        WTO   TEXT=((TEXTADDR,,)),                                   X
              CONNECT=MSGID,                                         X
              MF=(E,CONTINUE)
*
* Ship a "null end line" (no text) to indicate message is finished
        WTO   (''.E),                                                X
              CONNECT=MSGID,                                         X
              MF=(E,NULLEND)
```

```
              .
              .
              .
* Dynamic variables - declare as static and copy into getmained area
LINE1     WTO  TEXT=((,D)),                                          X
               CONSID=,                                              X
               CART=,                                                X
               DESC=5,                                               X
               MF=L
CONTINUE  WTO  TEXT=((,D)),                                          X
               CONNECT=,                                             X
               MF=L
NULLEND   WTO  (''.E),                                               X
               CONNECT=,                                             X
               MF=L
TEXTADDR DS OH
TEXTLEN  DC CL2' '
TEXTTEXT DC CL72' '
MSGID    DS F
              .
              .
              .
* DSECTs
* CMDX - Command exit parameter list                              -*
       IEZVX101                CMDX
```

6.1.2.3 Displaying Messages by Keyname

Messages can also be associated with individual *keynames* through the KEY parameter on the WTO or WTOR macro. A keyname is 1 to 8 alphanumeric characters, and it appears with the message on the console. The keyname can be used as an operand in the MVS DISPLAY R console command, which operators can issue at the console. Figure 41 shows an action message that is issued with a key, how it is displayed on a console, and the DISPLAY R responses by keyname.

Note: This KEY parameter is not to be confused with the KEY used to associate a group of EMCS consoles.

Assembler program issues:

```
  WTO   'XYZOOOA IGNORE THIS MESSAGE',ROUTCDE=(2),              C
        KEY==CL8'GARBAGE',DESC=(2)
```

Console message issued by program:

```
  @07.18.07 SC50 VAINI    @XYZOOOA IGNORE THIS MESSAGE
```

DISPLAY R Responses:

```
D R,KEY
 IEE112I 07.05.18 PENDING REQUESTS 344
 CNT KEY          CNT KEY          CNT KEY          CNT KEY
   1 GARBAGE

D R,KEY=GARBAGE
 IEE112I 07.06.43 PENDING REQUESTS 348
 RM=4    IM=14    CEM=1     EM=0     RU=0    IR=0    AMRF
 ID:R/K     T MESSAGE TEXT
      27045 I @XYZOOOA IGNORE THIS MESSAGE
```

Figure 41. Displaying Messages by Keyname

6.1.2.4 Message Descriptor Codes

The descriptor codes describe the significance of the messages and also determine how MCS displays and deletes the messages. Descriptor codes are specified in the DESC parameter of the WTO or WTOR macro, as shown in Figure 41 on page 140. See 2.1.1, “Message Type and Descriptor Codes” on page 14 for a description of descriptor codes.

If a problem program issues a message with descriptor code of 1 or 2, the message is deleted at address space termination or task termination.

Note: For authorized programs, this is not done automatically. You may have the messages deleted by using descriptor code 7 (DESC=7).

If you supply a descriptor code in the WTO macro, an indicator is inserted at the start of the message. The indicators are:

a blank ( )

@ The at sign (@)

***** The asterisk (*) ,

+ A blank followed by a plus sign (+)

The indicator inserted in the message depends on the descriptor code that the user supplies and whether the user is a privileged or APF-authorized program or a non-authorized problem program. Table 7 shows the indicator that is used for each descriptor code.

Table 7. Message Indicators for Descriptor Codes

Descriptor Codes	Non-authorized Problem Program	Authorized Program
1	@	*
2	@	*
3-10	′blank′ +	′blank′
11	@	*
12-13	′blank′ +	′blank′

Descriptor code restrictions: The following restrictions apply when specifying descriptor codes:

- Descriptor codes 1 through 6, 11, and 12 are mutually exclusive. Assign only one of these codes to a message. If you assign two mutually exclusive codes to one message, the system uses the most important code and ignores the other.
- Descriptor codes 7 through 10 and 13 can be assigned in combination with any of the mutually exclusive codes.
- Descriptor code 9 or 10 can be used only with descriptor code 8.

Descriptor code conditions: Under certain conditions, the system uses a descriptor code other than that specified in the macro as follows:

- The system assigns descriptor code 7 if all of the following are true when a problem program issues a WTO(R) macro:
  - The WTO(R) macro omits both DESC and ROUTCDE parameters, or specifies descriptor codes 1 or 2.

- The program is not authorized.

- The system assigns no descriptor code if all of the following are true:
  - An authorized program issued the WTO(R) macro.
  - The WTO(R) macro omits both DESC and ROUTCDE parameters.

6.1.2.5 Message Suppression

Action messages to the operator, which are identified by the @ or * indicator, can be individually suppressed by the installation. When a program invokes a WTO or WTOR macro to send a message, the system determines if the message is to be suppressed. If the message is to be suppressed, the system writes the message to the hardcopy log and the operator does not receive it on the screen.

Note: If AMRF(Y) is specified on the INIT statement, an action message stays in the action queue and can be displayed with the `D R` command. To prevent the message from being retained in AMRF, you can use RETAIN(NO) in MPF.

6.1.2.6 Single-line Messages

The actual text in a single line operator message can be up to 125 characters long. The message text may be coded “inline” on the WTO macro or the TEXT parameter may be used to point to the message text. If you need to issue the same message with variable text, the use of TEXT parameter is recommended rather than modifying the WTO parameter list. Note that when a program issues a WTO macro, the system translates the text; only standard printable EBCDIC characters are passed to MCS-managed display devices.

Note: No translation is done for an EMCS console.

6.1.2.7 Multi-line messages

To write a multiple-line operator message, either issue WTO with all lines of text, or issue each line of text separately using the CONNECT parameter on the WTO macro. The CONNECT parameter connects a subsequent message to a previous message. For example, if your program develops a large, multiple-line message of unknown length, it can issue several WTOs for the different parts of the message at different times. The CONNECT parameter forces all these WTOs to use the same message ID, and physically unites the different parts of the message at the display console as a single message. CONNECT is mutually exclusive with CONSID and CONSNAME, and it is not available with WTOR.

You can create with one WTO macro request a message that consists of up to 255 lines. For more than 255 lines, issue more than one WTO macro. The additional lines appear at the end of the message and continue until you specify an “end” line. For the first request, you must ensure that the left-most three bytes of register 0 are zero. After processing the first request, the system places a message identifier in register 1. For each additional request, you must pass this identifier to the subsequent lines through the CONNECT parameter of WTO.

Figure 42 shows a multiple-line WTO assembler program and the resulting console display:

```
WTOML    CSECT
         YREGS
         BAKR  R14,0
         USING WTOML,12
         LR    R12,R15
* ------ Issue Multiple-Line WTO
         SLR   R0,R0              Zero = Zero
         WTO   TEXT=((CL,C),(LL,L),(DL,D),(DEL,DE)),ROUTCDE=(7)
         PR
* ------ Multiple-Line WTO Text
CL       DC    AL2(CE-C)          Length of the following text
C        DC    C'CONTROL LINE'    C          34 char   1 C type
CE       DS    0C
LL       DC    AL2(LE-L)          Length of the following text
L        DC    C'LABEL LINE'      L          70 char   2 L type
LE       DS    0C
DL       DC    AL2(DE-D)          Length of the following text
D        DC    C'INFORMATION'     D          70 char   10 D type
DE       DS    0C
DEL      DC    AL2(DEE-DED)       Length of the following text
DED      DC    C'LAST LINE'       DE         70 char   1 DE type
DEE      DS    0C                            or
*                                 E          None      1 E type
*                                 line type  text      maximum number
         END   WTOML
```

Figure 42. Multi-line Message by Assembler Program Example

Resulting Console Display:

```
- 10.52.45 SC50 VAINI     +CONTROL LINE
-  LABEL LINE
-  INFORMATION
-  LAST LINE

  02.34.00 SC50           IEE923I K S,DEL=R,SEG=28,CON=N,RNUM=28,RTME=1/4,
   MFORM=(T,S,J,X)
```

When a multiple-line WTO message is displayed on a console, as shown in the example, the control line is prefixed with the data requested through the console's MFORM definition. Label lines, information lines, and the end line of the multiple-line WTO message are left justified with no prefix data.

Multiple-line message label lines provide column headings in tabular displays. You can change the column headings used to describe different sections of a tabular display by embedding label lines in the existing multiple-line WTO message for a tabular display.

Authorized assembler programs (supervisor state, or PSW key 0-7, or APF-authorized) may add lines to an existing multiple-line WTO message using the CONNECT parameter. They are also permitted to embed label lines within that existing multiple-line WTO message. The label line does not have to appear immediately following the control line and before the data lines. At most two label lines can appear consecutively without an intervening data line.

Appendix D, “Display System Groups Command Exit” on page 187 shows an authorized assembler program that uses the WTO CONNECT parameter while building a multiple-line WTO message.

6.1.2.8 WTOR Messages with Reply

The WTOR macro causes a message requiring a reply to be written from a program to the operator and the hardcopy log. The WTOR message processing is asynchronous; once the message is issued the program continues to execute and an ECB is posted when an operator replies to the message. All WTOR messages are action messages.

The WTOR macro specifies, in addition to the message, the information required by the system to return the reply to the issuing program. This includes:

Reply address Specifies the address of the area into which the system is to place the reply. The reply is converted to uppercase except for any part in single quotes and is left-justified at this address.

Reply length Specifies the maximum length, in bytes, of the reply text and should be as long as the expected reply.

Note: If the reply issuer gives a reply longer than the expected reply, an error message is issued.

ECB address Specifies the address of the event control block (ECB) to be posted when an operator enters the reply. After the program receives the reply, the ECB appears as follows:

Offset	Length(bytes)	Contents
0	1	Completion code
1	2	Not an intended programming interface
3	1	Console ID in hexadecimal

Note: The 1-byte console ID is no longer useful.

Note: Use RPLYISUR to obtain the 4-byte console ID and console name of the console issuing the reply.

A multiple line WTOR can be used only when SYNCH=YES is also specified. SYNCH=YES indicates the request is to be processed synchronously. This type of WTOR is used in error and recovery environments, when normal message processing cannot be used. The message is sent to the console, and the reply is obtained immediately, before control is returned to the caller. Before return, the reply and reply length are moved to the areas specified by the caller, and the ECB marked “complete.” Copies of the message and reply are *queued* for hardcopy logging.

6.1.3 DOM Macro Service

The DOM macro service deletes the messages that have been previously issued through the WTO or WTOR macros. When a message is DOMed and it is retained in AMRF, it is removed from AMRF. Depending on the timing of a DOM macro relative to the WTO or WTOR, the message may or may not appear on the operator′s consoles.

When an action message already exists on the operator screen, it is displayed in a format that indicates to the operator whether the message still requires some action to be taken. When the operator responds to the action requested in the message, the program responsible for the action message should delete the message. When an action message is deleted, the message format changes to

remind the operator that the requested action has been taken. Note that when DOM deletes a message, it does not actually erase the message. It only changes the message's format, displaying it like a non-action message. (WTOR messages are automatically deleted by MCS when an operator issues the message reply.)

The DOM processing does not affect the logging action. That is, if the message is supposed to be logged, it is, regardless of when or if a DOM is issued. The message is logged in the format of a message that is waiting for operator action.

When a WTO(R) macro is issued, the system assigns an identification number to the message and returns this number to the issuing program. When the message is no longer needed to be displayed, the DOM macro identifying the message should be issued. The messages to be deleted are identified by the identification number returned by WTO(R) macro service or by the TOKEN parameter that was used on the WTO(R) macro. If you specified the same TOKEN parameter on one or more WTO(R)s, a DOM with the same value on the TOKEN parameter deletes all of those messages.

6.1.4 WTO and LOADWAIT Macro Services

There are situations in which system operations cannot continue until the system operator takes some external action. An example might be an authorized application detecting a critical problem that warrants stopping the entire system to correct. Using the LOADWAIT macro and the WTO macro with the WSPARM parameter stops the system so the operator may attempt to correct the problem. By using LOADWAIT and WTO, you issue a message to the operator and place the system into a wait state. By placing the system into a wait state, all processor activity is stopped.

To place the system into a wait state:

- Issue the LOADWAIT macro to store wait state information into the WSPARM parameter list of the WTO macro. The WTO with options LINKAGE=BRANCH, SYNCH=YES, and WSPARM=wait_state_parm_addr uses the parameter list from LOADWAIT to put the system into the wait state and issues one message to the operator.
- Issue the WTO macro and specify the parameter WSPARM along with LINEAGE=BRANCH and SYNCH=YES to issue a message and load the wait state. The WSPARM parameter contains the address of the parameter list that you previously built using the LOADWAIT macro. WTO issues one message to the operator and uses the parameter list from LOADWAIT to put the system into the wait state. The wait state code and operator message explain what action the operator is to take.

Note: All input storage for WTO and LOADWAIT must be page-fixed as it runs disabled. The use of these options as well as the SYNCH=YES WTOR has a serious impact on the whole system and should be used as a last resort.

The LOADWAIT macro specifies the following wait state information:

Wait state type This must be either restartable or non-restartable.

A non-restartable wait state allows the calling program to stop the system. This type of wait state is used only when MVS cannot be allowed to continue operating. To continue processing, the operator must re-IPL the system.

A restartable wait state allows the calling program to communicate with the operator (PSAPARM - a fullword field that you can use for additional information at the time you issue the LOADWAIT macro), allows the operator to communicate with the calling program (ACTCODE - a 1-byte field that the system updates with the contents of storage location X′30E′ after the system is restarted), and prevents any other activity in the system during operator communications. This type of wait state is used when:

- MVS must be stopped until the operator corrects some external condition, but can continue after the condition is corrected.
- MVS should be stopped to preserve storage contents for problem determination, but can continue afterward. An example of this type of wait state is when a SLIP trap (ACTION=WAIT) has been matched.

To continue processing, the operator initiates a restart action from the system console.

Wait state code This is the wait state code you must assign for the error.

Reason code This is the reason code you can assign for further information about the wait state.

Action code location This is a 1-byte area to receive information that the operator can supply to the calling program.

Additional information Additional information is made available to the operator. This could be a system ID or a pointer to a data area for further information about the wait state.

6.1.5 WTL Macro Service

The write-to-log (WTL - SVC 36) macro causes a message to be written to the hardcopy media. The message can include up to 126 characters. If the message text exceeds 126 characters, truncation occurs.

Note: The WTL macro should no longer be used. In its place, use a WTO with MCSFLAG=HRDCPY.

Note: When JES3 5.2.1 is installed, all hardcopy media records have the same format. The JES3 5.2.1 DLOG, when activated, continues to record the old JES3 SYSLOG format records into the SYSLOG data set. MCS SYSLOG hardcopy media is not required to be active. However, the SYSLOG task (activated through WRITELOG START command) must be active. The SYSLOG task is activated automatically when the JES3 DLOG is activated.

6.1.6 Writing to Hardcopy Media

There are other ways to request messages to be written to the hardcopy media besides the WTO macro:

- You may specify, through the HARDCOPY statement in the CONSOLxx parmlib member, the message routing codes MCS is to include in the hardcopy message set. The minimum set of routing codes is 1, 2, 3, 4, 7, 8, 10, and 42.

- You can use the HRDCPY option on the MCSFLAG parameter on the WTO macro.

Note: IBM recommends you use the WTO macro with the MCSFLAG=HRDCPY parameter instead of WTL, because WTO supplies more data in the hardcopy media than WTL.

The text of your WTL macro is written on the master console instead of on the system log if the system log is not active.

6.2 Operator to Program Communication

Operators can pass information to a program by issuing a STOP or a MODIFY command or a REPLY to a WTOR. This form of operator to program communication is based on QEDIT (command input buffer manipulation - SVC 34) and EXTRACT (extract TCB information) macro services.

6.2.1 QEDIT Macro Service

The QEDIT macro generates the required entry parameters and processes the command input buffer for the following uses:

- Dechaining and freeing of a command input buffer from the command input buffer chain for a task
- Setting a limit for the number of command input buffers that may be simultaneously chained for a task

The command input buffer contains the MODIFY command operand text among many other things.

In order to accept MODIFY and STOP commands, the program must be set up properly:

- A program must first obtain a pointer to the communications ECB and a pointer to the first command input buffer (CIB) on the CIB chain for the task. Issue the following macro:

  ```
  EXTRACT answer_area,FIELDS=COMM
  ```

  Returned is the address of the command scheduler communications list (mapped by IEZCOM). The list consists of a pointer to:

  - The communications event control block
  - A pointer to the command input buffer (CIB) - field COMCIBPT
  - A token

    Note: If a token exists, the high-order bit of the token field is set to one. The token is used only with internal START commands.

  The ECB is posted whenever a STOP or a MODIFY command is issued.

  The CIB (mapped by IEZCIB) contains the information specified on the STOP, START, or MODIFY command. If the job was started from the console, initially the CIB pointer points to the START CIB. If the job was not started from the console, the CIB pointer is zero. The CIB data area is followed by the CIB extension, which includes a pointer to:

  - The UTOKEN of the console
  - The console's command authority
  - The console name

- The CART (CIBXCART)
- The 4-byte console ID (CIBXCNID)

The CIB extension data should be used to verify the authority of the console entering the command and to send command responses back to the originating console. The CIBX (CIB extension) is new with MVS/ESA SP Version 4 and can be found by adding the CIBXOFF offset value to the address of the CIB. For an example, see Figure 40 on page 139.

- If the address of the START CIB is present, once you have processed the start command parameters, use the QEDIT macro to free this CIB:

  ```
  QEDIT ORIGIN=address_of_pointer_to_CIB,BLOCK=address_of_CIB
  ```

- The CIB counter must then be set to a non-zero positive value to allow CIBs to be chained for the job:

  ```
  QEDIT ORIGIN=address of pointer to CIB,CIBCTR=n
  ```

  The CIBCTR value n is any integer from 0 to 255. If n is set to zero, no MODIFY commands are accepted for the job and message IEE342I MODIFY REJECTED-TASK BUSY is issued. This message is also issued when the number of queued CIBs reaches the CIBCTR value before the job releases previously queued CIBs. However, STOP commands are accepted for the job regardless of the value set for CIBCTR. After a STOP command is issued, the system sets the value of CIBCTR to zero to prevent more modify CIB′s from being chained. To continue processing MODIFY commands after a STOP command, you must use the QEDIT macro to set the CIBCTR to non-zero.

- For the duration of the job, your program can wait on or check the communications ECB at any time to see if a command has been entered for the program. Check the verb code in the CIB to determine whether a STOP or a MODIFY command has been entered. After processing the data in the CIB, issue a QEDIT macro to free the CIB.

  Note: The communications ECB is cleared by QEDIT when no more CIBs remain. Care should be taken if multiple subtasks are examining these fields. Any CIBs not freed by the task are unchained by the system when the task is terminated. The area addressed by the pointer obtained by the EXTRACT macro, the communications ECB, and all CIBs are in protected storage and may not be altered.

The sample program in Appendix H, “Extended MCS Console Sample” on page 203 includes an example of how you can code the EXTRACT and QEDIT macros to accept MODIFY and STOP commands.

IBM recommends you use the MODIFY and STOP operator communication for programs that require continuous operator communication rather than using “always” outstanding WTORs.

6.3 Program to System Communication

In the following text, program to system communication means an authorized assembler program′s capability to enter operator commands and receive command responses. The MGCR and MGCRE macro (SVC 34) services enable a program to issue commands. The essential difference between the MCGRE and MGCR is that the MGCRE macro provides an easier to use interface and provides up-to-date parameters and services. IBM recommends you use

MGCRE rather than MGCR. See 4.3, “MGCRE Macro Service” on page 111 for details.

The strategic way for a program to retrieve command responses (and other console traffic) is to activate an extended MCS console. The MCSOPER macro service for REQUEST=ACTIVATE defines and activates an extended MCS console to the system. The MCSOPMSG macro service allows programs to receive messages directed to an extended console by using different message selection criteria. See 4.1, “MCSOPER Macro Service” on page 98 and 4.2, “MCSOPMSG Macro Service” on page 108 for the details of these macros.

6.3.1 MGCR Macro Service

IBM recommends you use the MGCR macro only for entering internal START or REPLY commands using the pseudo master console ID zero. The authorized program issuing MGCR can also pass a user security token to the system. The system uses the user security token for command authorization checking. The user′s security token can be extracted with RACROUTE REQUEST=TOKENXTR macro and modified with the RACROUTE REQUEST=TOKENBLD macro.

In the case of the START command, the program that issues MGCR can pass 31 bits of information (in a field called a program token) to the program being started.

Before issuing the MGCR macro, set general register 0 to zero, and the command buffer (mapped by IEZMGCR) must be initialized. Note that command responses are not returned back to the pseudo master console ID zero; they come to the master console if active.

Note: For details on the use of the MGCRE macro, see 4.3, “MGCRE Macro Service” on page 111.

6.3.2 CONVCON Macro Service

Applications that act as operators (process commands or issue messages) can extract information about MCS or extended MCS consoles using the CONVCON macro service. The CONVCON macro service can:

- Determine the name of a console behind a console ID
- Determine the ID of a console for a console name
- Validate a console name or console ID
- Validate a console area ID
- Check if a console is active

Before invoking the CONVCON macro, the application must set up the CONV parameter area (mapped by IEZVG200).

In a sysplex environment, operators should use console names rather than console IDs to avoid confusion. Console IDs may change during reinitialization of sysplex systems; console names do not. A console ID is not guaranteed to be associated with one console for the life of a sysplex.

6.3.3 CPF Macro Service

Through CPF, applications can assign unique installation-defined prefixes so that operators can direct a prefixed command to the system in the sysplex which is defined as the “target” system for the prefix (normally the system where the application is running). The operators do not have to keep track of which system in the sysplex an application is currently active (which may change, for example, as a result of ARM restarts). All they have to remember is the command prefix for the application and CPF automatically routes the prefixed commands to the right system in the sysplex. For example, if your installation is running multiple JES3 complexes in the sysplex, you define a unique command prefix for each JES3 complex, each of which directs the JES3 commands to the specific JES3 global (even if you go through a JES3 DSI). When an application issues the CPF REQUEST=DEFINE macro during its initialization, CPF adds to the CPF table the prefix for the system to which the application wants the commands directed. The CPF table is known by all systems in the sysplex. Also, by issuing the CPF macro, an application can redefine or delete entries in the table.

CPF processing also ensures that two or more systems do not have the same or overlapping prefixes, which helps prevent confusion. To define a valid prefix that does not conflict with existing prefixes, adhere to the following guidelines:

- Define a prefix as a 1- to 8-character string.
- Do not use a prefix that is a command or an abbreviation of a command.
- Do not define a prefix that is either a subset or a superset of an existing prefix with the same first character.

You can see which prefixes already exist by issuing the DISPLAY OPDATA command.

The SCOPE parameter of the CPF macro specifies the range of systems to which a command with this prefix can be routed for execution:

SCOPE=SYSPLEX Specifies that the command issued can be routed to another system in the sysplex for execution.

SCOPE=SYSTEM Specifies that the command issued is to be executed in the system on which the command is entered.

The FAILDISP parameter of the CPF macro defines what the system does on behalf of the prefix owner when the defining address space or system on which the command is processed terminates:

RETAIN If you specify FAILDISP=RETAIN, the system keeps the prefix in the CPF table at all times, even if the address space owning the prefix terminates.

SYSPURGE If you specify FAILDISP=SYSPURGE, the system deletes the command prefix from the CPF table when the receiving (target) system is removed from the sysplex, but not when the defining address space terminates.

PURGE If you specify FAILDISP=PURGE, the command prefix is automatically deleted when the receiving system is removed from the sysplex, or the defining address space terminates.

The REMOVE parameter of the CPF macro specifies whether the command prefix is removed from the command text prior to being executed on the receiving system.

REMOVE=NO Indicates the command prefix and the command are presented to the receiving system.

REMOVE=YES Indicates the command prefix is removed from the command before it is presented to the receiving system.

See Appendix K, “Define, Redefine, and Delete CPF Entry” on page 217 for an example on an MPF command installation exit that lets you manipulate command prefixes.

Note: For other definitions of command prefixes, see 2.7.2.1, “IEECMDPF Sample Program” on page 41.

Chapter 7. JES2 Sysplex Operations

With the changes made in MCS support to operate in a sysplex, for JES2 operations, consider the following for the multisystem environment:

- Defining consoles in a sysplex
- Command routing and message flow
- SDSF support for a sysplex

7.1 Defining Consoles in a Sysplex

With the ability to operate a sysplex from a single console, you may want to consider defining the MCS consoles in a different way. Consider all the various options that are now available with the MCS multisystem support:

- A single CONSOLxx member for the sysplex.

  Consoles receive messages by the routing code assignments. A single console could now operate or control each system in the MAS or sysplex. See 3.3.3, “Shared CONSOLxx Member Considerations” on page 83.

- Message routing is a sysplex

  In a sysplex, a message issued on one system is routed to other systems for display on a console matching the routing attributes of the message.

- Use of the ROUTE command.

  With the ROUTE command, commands may be directed from a single console to any system or systems in the sysplex. See 2.7.3, “MVS ROUTE Command” on page 43.

  Also, some JES2 commands have a single system aspect in that they can be issued on any member in the JES2 MAS and affect jobs in execution on another member of the MAS. See 7.2, “Command Routing and Message Display” on page 155.

- SDSF support for the sysplex environment.

  Beginning with SDSF 1.4, a ULOG capability is provided to allow the user or operator to have an extended MCS console from which the sysplex can be controlled. See 7.3.2, “User Session Log” on page 157.

- Defining the JES2 command prefix for each system. See 7.1.1, “Defining the JES2 Command Prefix.”

7.1.1 Defining the JES2 Command Prefix

Since it is possible for an operator to enter JES2 commands for all systems in the sysplex from a single console, the JES2 prefix character determines the target system. You can specify whether a single system or all systems recognize a JES2 command prefix by specifying its scope. The scope is specified on the CONDEF initialization statement, as shown in Figure 43 on page 154.

The CONCHAR keyword defines the command prefixes.

```
CONDEF CONCHAR=$,SCOPE=SYSTEM|SYSPLEX
```

Figure 43. Command Prefix Definition on CONDEF Statement

SYSPLEX JES2 sysplex-scoped prefixes can communicate with any system from any console.

Note: This is not a recommended option for a JES2 MAS unless you feel the operators can easily remember which prefix goes with each system.

SYSTEM JES2 system-scoped prefixes communicate with JES2 on the issuing system.

Note: This is the recommended option.

The SYSTEM scoped prefixes are registered on *each* system in the JES2 complex. Commands that begin with SYSTEM scoped prefixes are processed on the processor on which the command is entered. Explicitly routed commands with SYSTEM-wide prefixes are processed on the target processor.

7.1.1.1 Prefix Scope Processing

JES2 registers with CPF the command prefix that is defined with the CONCHAR keyword on the JES2 CONDEF initialization statement. One JES2 prefix can be specified. The prefix is always registered with CPF on each processor. In general, commands routed explicitly with the MVS ROUTE command and with a JES2 prefix are rerouted by MCS as required according to the JES2 prefix definition after the ROUTE command routing completes. If a command is routed using a JES2 prefix, MCS routes it only *once*, even if the command has multiple JES2 prefixes.

JES2 defines both SYSTEM and SYSPLEX scope prefixes with the option FAILDISP=SYSPURGE. This means that a command prefix is deleted when the defining system is removed from the sysplex but not when the defining address space terminates. The CPF REMOVE=NO option is used by JES2 for the prefix removal. This implies that both the command prefix and the command are presented on the receiving system.

7.1.1.2 Prefix Definition Considerations

Specifying the same CONCHAR prefix character on every system in the MAS or sysplex requires a scope of SYSTEM. This option allows for the use of the ROUTE command to send the same JES2 command to one or multiple members. See 2.7.3, “MVS ROUTE Command” on page 43. Also, see 2.7.2.1, “IEECMDPF Sample Program” on page 41 for the option of defining the system name as a command prefix as a way of sending JES2 commands to the appropriate member.

The DISPLAY OPDATA command lists all currently registered command prefixes and their attributes.

```
D OPDATA
IEE603I 14.42.58 OPDATA DISPLAY 840
 PREFIX     OWNER      SYSTEM     SCOPE     REMOVE   FAILDSP
 $          JES2       SC50       SYSTEM    NO       SYSPURGE
 $          JES2       SC52       SYSTEM    NO       SYSPURGE
 $          JES2       SC54       SYSTEM    NO       SYSPURGE
 $          JES2       SC53       SYSTEM    NO       SYSPURGE
 $          JES2       SC49       SYSTEM    NO       SYSPURGE
 $          JES2       SC43       SYSTEM    NO       SYSPURGE
 $          JES2       SC42       SYSTEM    NO       SYSPURGE
 $          JES2       SC47       SYSTEM    NO       SYSPURGE
 $          JES2       SC55       SYSTEM    NO       SYSPURGE
 SC42       IEECMDPF   SC42       SYSPLEX   YES      SYSPURGE
 SC43       IEECMDPF   SC43       SYSPLEX   YES      SYSPURGE
 SC47       IEECMDPF   SC47       SYSPLEX   YES      SYSPURGE
 SC49       IEECMDPF   SC49       SYSPLEX   YES      SYSPURGE
 SC50       IEECMDPF   SC50       SYSPLEX   YES      SYSPURGE
 SC52       IEECMDPF   SC52       SYSPLEX   YES      SYSPURGE
 SC53       IEECMDPF   SC53       SYSPLEX   YES      SYSPURGE
 SC54       IEECMDPF   SC54       SYSPLEX   YES      SYSPURGE
 SC55       IEECMDPF   SC55       SYSPLEX   YES      SYSPURGE
```

For the definitions of the command prefixes for the following owners:

JES2 See 7.1.1, “Defining the JES2 Command Prefix” on page 153.

IEECMDPF See 2.7.2.1, “IEECMDPF Sample Program” on page 41.

7.2 Command Routing and Message Display

The ability to route commands in a sysplex from a single console provides a new way of operating a JES2 MAS. The MVS ROUTE command allows commands to be sent to other systems or groups of systems and allows the retrieval of the messages back at the issuing console.

7.2.1 JES2 Command Enhancements for Sysplex

Several command changes have been made for JES2 commands to improve operations in a multisystem environment. The following commands allow a single system console to control jobs on any system in the sysplex:

$CJ Cancel a job on any system

$EJ Restart an executing job on any system

$DMJ Send a message to jobs on any system

$DMEMBER Display the status of MAS members

$TMEMBER Change a member’s operational mode (5.2)

$CT Cancel a TSU job on any system

$ECKPTLOCK Reset the checkpoint lock (5.2)

These commands became operational with JES2 5.1 except for the two commands that are shown as JES2 (5.2).

7.3 SDSF Enhancements for a Sysplex

SDSF 1.4 and SDSF 1.5 have the following enhancements to support the sysplex environment:

- MAS display panel
- User session log (ULOG)
- Sysplex-wide DA support
- Operations log (OPERLOG)

The SDSF I, O, and ST panels now have a sysplex-wide MAS scope and they display the job input and output queues identically for all members.

7.3.1 MAS Display Option

The new options introduced with SDSF 1.4 on the primary option menu panel are MAS and ULOG, as shown in Figure 44.

```
V1R4M0 ------------------  SDSF PRIMARY OPTION MENU  -------------------
COMMAND INPUT ===>                                          SCROLL ===> PAGE

 TYPE AN OPTION OR COMMAND AND PRESS ENTER.

    LOG       - DISPLAY THE SYSTEM LOG
    DA        - DISPLAY ACTIVE USERS OF THE SYSTEM
    I         - DISPLAY JOBS IN THE JES2 INPUT QUEUE
    O         - DISPLAY JOBS IN THE JES2 OUTPUT QUEUE
    H         - DISPLAY JOBS IN THE JES2 HELD OUTPUT QUEUE
    ST        - DISPLAY STATUS OF JOBS IN THE JES2 QUEUES
    PR        - DISPLAY JES2 PRINTERS ON THIS SYSTEM
    INIT      - DISPLAY JES2 INITIATORS ON THIS SYSTEM
    MAS       - DISPLAY JES2 MEMBERS IN THE MAS
    ULOG      - DISPLAY USER SESSION LOG

    TUTOR     - SHORT COURSE ON SDSF (ISPF ONLY)
    END       - EXIT SDSF

    LICENSED MATERIALS - PROPERTY OF IBM

    5665-488 (C) COPYRIGHT IBM CORP. 1981, 1993. ALL RIGHTS RESERVED.
    US GOVERNMENT USERS RESTRICTED RIGHTS - USE, DUPLICATION OR
    DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
```

Figure 44. SDSF Primary Option Menu

The MAS display shown in Figure 45 is a new tabular display that can be used to display and control all members of a JES2 MAS complex of up to 32 systems. For very large environments, the SDSF MAS display provides a full-screen means of simplifying the task of displaying system status. With the use of action characters, the user can start, stop, and restart systems. The HOLD, DORMANCY, and SYNCTOL parameters displayed can be overtyped to allow changing of these checkpoint parameters.

Like other SDSF tabular displays, the rows containing a system with a status of ACTIVE are highlighted. All the fields are not shown in Figure 45.

The MAS display supports action characters to display, start, restart, reset, and stop a member. Many of these actions require the command to be sent to the

target system. Since the system may not be connected to the MAS, the MVS ROUTE command is used to send the command to the system in place of the $M command. The command recognition character for the target JES2 (rather than the issuing JES2) is used when the ROUTE command is generated.

Specifying MAS ALL displays 32 members with the undefined members or systems in the display. When undefined systems are being displayed (as a result of the MAS ALL command), residual data is not shown. Only the NAME, STATUS, and SYSID columns contain valid data. The remaining fields contain either zeros or blanks as appropriate to the column data type.

Note: The MAS option appears on the SDSF primary option panel only when the user is authorized to use the display. The MAS display is only available with a JES2 Version 5 environment.

```
SDSF MAS DISPLAY SC53  DEFINED SYSTEMS     23% SPOOL   LINE 1-14 (14)
COMMAND INPUT ===>                                      SCROLL ===> HALF
NP NAME STATUS       SID PREVCKPT     HOLD  ACTHOLD DORMANCY    ACTDORM SYNCTOL
   SC52 ACTIVE         9     2.68       20       21 (20,200)        100     120
   SC53 ACTIVE        10     1.96       20       28 (20,200)        280     120
   SC54 ACTIVE        11     3.44       20       21 (20,200)         64     120
   SC47 ACTIVE        12     3.12       20       21 (20,200)        229     120
   SC49 DRAINED       13 27070.24       20       10 (20,200)         65     120
   SC50 ACTIVE        14     2.32       20       21 (20,200)        211     120
```

Figure 45. MAS Display Panel

7.3.2 User Session Log

A new option on the primary option menu panel, ULOG, is introduced with SDSF 1.4, and is shown in Figure 46. The user session log (ULOG) is a new panel that displays all MVS commands, JES2 commands, and the responses issued during a user's session. This includes commands generated by SDSF from the SYSLOG panel.

```
V1R4M0 ------------------  SDSF PRIMARY OPTION MENU  --------------------
COMMAND INPUT ===>                                            SCROLL ===> PAGE

 TYPE AN OPTION OR COMMAND AND PRESS ENTER.

    LOG       - DISPLAY THE SYSTEM LOG
    DA        - DISPLAY ACTIVE USERS OF THE SYSTEM
    I         - DISPLAY JOBS IN THE JES2 INPUT QUEUE
    O         - DISPLAY JOBS IN THE JES2 OUTPUT QUEUE
    H         - DISPLAY JOBS IN THE JES2 HELD OUTPUT QUEUE
    ST        - DISPLAY STATUS OF JOBS IN THE JES2 QUEUES
    PR        - DISPLAY JES2 PRINTERS ON THIS SYSTEM
    INIT      - DISPLAY JES2 INITIATORS ON THIS SYSTEM
    MAS       - DISPLAY JES2 MEMBERS IN THE MAS
    ULOG      - DISPLAY USER SESSION LOG

    TUTOR     - SHORT COURSE ON SDSF (ISPF ONLY)
    END       - EXIT SDSF

    LICENSED MATERIALS - PROPERTY OF IBM

    5665-488 (C) COPYRIGHT IBM CORP. 1981, 1993. ALL RIGHTS RESERVED.
    US GOVERNMENT USERS RESTRICTED RIGHTS - USE, DUPLICATION OR
    DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
```

Figure 46. SDSF Primary Option Menu

7.3.2.1 SDSF Extended MCS Console

The first time the SDSF user enters a command from his session or when the ULOG command is entered from any SDSF panel, SDSF uses MVS console services to acquire an extended console for the user and an extended console is created by SDSF. SDSF then uses the extended MCS console to return responses directly to the screen being viewed, without the need to access the LOG display. All commands are issued using the user's console identifier, which defaults to the user ID of the user. The ULOG display is shown in Figure 47, and the user ID shown is ROGERS. SDSF supplies an OPERPARM when activating the extended console that gives the console master level authority. Any OPERPARM segment for the user that is defined through RACF is ignored.

```
 SDSF ULOG  CONSOLE ROGERS                          LINE 0       COLUMNS
 COMMAND INPUT ===>                                            SCROLL ===>PAGE
******************************** TOP OF DATA ****************************
SC47      94168  15:44:46.98            ISF031I CONSOLE ROGERS ACTIVATED









******************************** BOTTOM OF DATA *************************
```

Figure 47. ULOG Panel Display

The ULOG eliminates the need to look in SYSLOG or the master console for command responses. All responses associated with the user are placed into

the user session log. Users with sufficient command authority perform MVS operator activities from a SDSF session, and view only the appropriate response directly at their terminal. This increases the productivity of operations and system support personnel by reducing the number of messages that must be read to determine the command response.

When a command is issued from the ULOG display, **/d c,cn=(rogers)** (shown in Figure 48), the response messages are displayed at the top of the screen. This is also true if the command is entered from the SYSLOG display screen. This is a major improvement over previous systems where the operator had to view the command responses through the SYSLOG or the master console, where the message rates can be extremely high.

```
SDSF ULOG  CONSOLE ROGERS                          LINE  COMMAND ISSUED
COMMAND INPUT ===> /d c,cn=(rogers)                          SCROLL ===>
RESPONSE=SC47
 IEE889I 15.48.13 CONSOLE DISPLAY 062
 MSG: CURR=2    LIM=1500 RPLY:CURR=2    LIM=999  SYS=SC47      PFK=00
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS ---------------
  ROGERS             ---  COND=A      AUTH=MASTER       UD=N
   SDSF                   MFORM=S         LEVEL=ALL
   SC47                   ROUTCDE=NONE
                          CMDSYS=SC47
                          MSCOPE=*ALL
                                                          CMDSYS=
                                                          MSCOPE=
SC47      94168  15:46:33.61            DSI020I OPERATOR 00000002 LOGGE
SC47      94168  15:47:53.13           -D A,L
SC47      94168  15:47:53.18            IEE114I 15.47.53 94.168 ACTIVITY
                                         JOBS     M/S    TS USERS    SYSAS
                                        00001    00018    00008      00019
                                         LLA      LLA      LLA      NSW  S
                                         AOFAPPL  AOFAPPL  NETVIEW  NSW  S
                                         RMF      RMF      IEFPROC  NSW  S
                                         RACF     RACF     RACF     NSW  S
                                         NET      NET      NET      NSW  S
                                         NETVSSI  NETVSSI  NETVIEW  NSW  S
                                         EKGXRODM EKGXRODM START    NSW  S
                                         GMFHS    GMFHS    STEP1    IN   S
                                         MERONIZ  STEP001           NSW  J
```

Figure 48. ULOG Panel Display with Messages

The message responses are accumulated by SDSF in an in-storage table (the user session log) that is displayed in a manner similar to the output data set panel. Each command and message response is recorded in the table for subsequent browsing through the SDSF browse facilities. When the ULOG command is entered, the user is placed directly into SDSF browse for the user session log.

7.3.3 Delay Interval for ULOG

A user option is provided to control the delay interval used for waiting for command responses. The default interval is one second. To change the delay interval to five seconds, enter:

```
SET DELAY 5
```

Normally most command responses are returned within one second. A ROUTE command however could take much longer. When the delay interval expires

and no responses are available, a message appears at the top of the display, as follows:

```
NO RESPONSE RECEIVED
```

7.3.4 Sysplex Wide DA Panel

The Display Active users panel can now display all address spaces in a sysplex and is introduced with SDSF 1.5. Three new columns are added to the display to reflect sysplex data:

SYSNAME The MVS system name on which the job is executing.

SPAG The system demand paging rate for the system the job is executing on.

SCPU% The system CPU utilization for the system the job is executing on.

If you enter an action character or overtypeable field on a sysplex-wide DA panel, the MVS commands generated and targeted for other systems are prefixed with the MVS ROUTE command.

Note: Sysplex-wide DA support requires JES2 5.2.0, and RMF must be active in each system for which data is to be gathered.

7.3.4.1 SYSNAME Command

The sysplex-wide DA panel is controlled by using the SYSNAME command. The SYSNAME command must be issued before the DA display displays other systems in the sysplex. The SYSNAME command options are:

SYSNAME * Allows displaying the address spaces for all systems in the sysplex.

SYSNAME system-name Allows displaying the address spaces running on the system specified by *system-name*.

The * and % wildcards may be used in specifying the system-name.

SYSNAME ? Displays a pop-up window that displays a current SYSNAME setting and allows a new SYSNAME setting to be set.

Note: The current SYSNAME setting is returned to the command line when SDSF is initialized under TSO.

SYSNAME With no parameters, the command displays the address spaces running on the system the user is logged on to.

Note: The SYSNAME setting is saved across sessions when SDSF is running under ISPF.

SYSNAME Command Authorization: The SYSNAME command can be authorized using ISFPARMS with the ISFGRP macro and AUTH=SYSNAME parameter or using SAF and the SDSF class profile:

```
ISFCMD.FILTER.SYSNAME
```

You can also set a default SYSNAME via the ISYS parameter in ISFPARMS. The ISYS setting can be overridden via authorized SYSNAME command users.

The ISFGRP macro now supports an optional ISYS=LOCAL|NONE parameter, which defaults to LOCAL

ISYS=LOCAL Only address spaces running on the system the user is logged on to are displayed.

ISYS=NONE Address spaces are not limited by system.

Figure 49 shows an example of the sysplex-wide DA panel. As you can see, the display has a lot of duplicate jobname entries but they vary in the SYSNAME column.

Note: The SYSNAME column has been moved from the alternate screen to the primary screen using the ARRANGE command.

By issuing the command SYSNAME *, the DA display shown is a sysplex-wide display.

```
  Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF DA SC55  (ALL)    PAG    0 SIO     5 CPU  29/ 18  LINE 1-37 (251)
COMMAND INPUT ===>                                        SCROLL ===> HALF
PREFIX=*  DEST=(ALL)  OWNER=*
NP   JOBNAME  STEPNAME PROCSTEP SYSNAME    CPU%  EXCP-CNT   CPU-TIME    SIO JOB
     *MASTER*                   SC47       0.44     4,793     613.59   0.01 STC
     XCFAS    XCFAS    IEFPROC  SC47       2.72         3     658.81   0.00
     JESXCF   JESXCF   IEFPROC  SC47       0.26       330     372.66   0.00
     IXGLOGR  IXGLOGR  IEFPROC  SC47       0.01       121       2.18   0.00
     JES2     JES2     IEFPROC  SC47       0.98   118,109    1532.70   0.44
     RACF     RACF     RACF     SC47       0.00        11       0.02   0.00
     *MASTER*                   SC49       0.61     6,733    5694.75   0.01
     XCFAS    XCFAS    IEFPROC  SC49       4.63         3    5404.93   0.00
     JESXCF   JESXCF   IEFPROC  SC49       0.31       335    2993.20   0.00
     IXGLOGR  IXGLOGR  IEFPROC  SC49       0.02        25       9.98   0.00
     JES2     JES2     IEFPROC  SC49       1.19   143,950   11379.31   0.42
     RACF     RACF     RACF     SC49       0.00         9       0.08   0.00
     JES3CI   CI7      JES3CI   SC49       0.00       293       0.97   0.00
     *MASTER*                   SC50       0.59    10,167    5650.12   0.02
     XCFAS    XCFAS    IEFPROC  SC50       7.56         3    5936.12   0.00
     JESXCF   JESXCF   IEFPROC  SC50       0.39       343    3199.65   0.00
     IXGLOGR  IXGLOGR  IEFPROC  SC50       0.02        24      10.50   0.00
     JES2     JES2     IEFPROC  SC50       1.31     2,952     242.68   0.41
     RACF     RACF     RACF     SC50       0.00         9       0.08   0.00
     *MASTER*                   SC52       0.54     6,591    5092.52   0.01 STC
     XCFAS    XCFAS    IEFPROC  SC52       4.55       167    5369.52   0.00
     JESXCF   JESXCF   IEFPROC  SC52       0.30       329    2879.62   0.00
     IXGLOGR  IXGLOGR  IEFPROC  SC52       0.02        57       9.47   0.00
     JES2     JES2     IEFPROC  SC52       1.13   145,048   11456.79   0.42
     RACF     RACF     RACF     SC52       0.00         9       0.08   0.00
     *MASTER*                   SC55       0.90    11,158    7021.82   0.02 STC
     XCFAS    XCFAS    IEFPROC  SC55       8.40         3    6335.19   0.00
     JESXCF   JESXCF   IEFPROC  SC55       0.39       327    3020.48   0.00
     IXGLOGR  IXGLOGR  IEFPROC  SC55       0.03        70     207.13   0.00
     JES2     JES2     IEFPROC  SC55       1.42   149,980   12154.45   0.41
     RACF     RACF     RACF     SC55       0.00         9       0.08   0.00
     TRAUNER  IKJACCNT SCT3055D SC55       0.02     1,769      34.70   0.00 TSU
     JEST     JEST     IEFPROC  SC55       1.56    22,754   27838.49   0.41
```

Figure 49. Sysplex-Wide Display Active Panel

The new information displayed on the sysplex-Wide DA panel title line shows the following:

- **SC55** is the SYSNAME of the users session.
- **PAG**, **SIO**, and **CPU** are rates for the local system of the users session.
- In you are running in LPAR mode, the local system CPU rate, shown on the top line in Figure 49, as an MVS view and as in this example an LPAR view (29/ 18).

Note: The SYSNAME command can be issued using the Filter action bar and using option 5, which is the SYSNAME setting to get the pop-up window.

7.3.5 OPERLOG Panel

The OPERLOG panel allows users to browse a merged, sysplex-wide system message log stream and is introduced with SDSF 1.5. There are new components in MVS/ESA Version 5.2, the MVS System Logger and the OPERLOG log stream. The OPERLOG log stream provides an alternative to the single-system DASD data sets used for SYSLOG.

There are new parameters on the LOG command to allow a user to browse the OPERLOG or the SYSLOG. The OPERLOG panel is accessed with the existing LOG command:

LOG Specifying LOG with no parameters now displays the OPERLOG panel if the MVS system logger has been activated on the system the user is logged on to.

The SYSLOG panel is displayed if the MVS system logger is not active on the system the user is logged on to.

LOG O Displays the OPERLOG panel.

LOG S Displays the SYSLOG panel.

To have access to the OPERLOG panel, panel access has to be authorized through:

```
ISFPARMS (ISFGRP macro; AUTH=LOG parameter)

   or

SAF (SDSF class)
ISFCMD.ODSP.LOG.jesx
```

Connection to the log stream is protected by:

```
SAF (LOGSTRM class)

SYSPLEX.OPERLOG
```

Note: The sysplex-wide OPERLOG requires at least MVS 5.2.0.

7.3.5.1 Using OPERLOG

The OPERLOG panel functions are generally the same as the SYSLOG panel except for the following command enhancements due to the absence of absolute line numbers in the OPERLOG:

PRINT * number Prints the number of lines relative to the top line of the display. This form of PRINT is available on all panels.

LOCATE line-number Line-number locates relative to the top line of the display for the OPERLOG.

NEXT number H|D The NEXT command allows quick positioning forward in the OPERLOG by hours or days. Positioning begins from the top of the display.

Specifying the command without any parameters positions forward by one hour.

PREV number H|D The PREV command positions backward by the number of hours or days relative to the top line of the display in the OPERLOG.

Specifying the command without any parameters positions backward by one hour.

7.3.5.2 Viewing OPERLOG

Figure 50 shows the OPERLOG panel. The log stream data is formatted to look like the SYSLOG panel. On the OPERLOG panel title line, there is different information than on the SYSLOG:

- Four WTORS is the number of outstanding WTORs. These WTORs appear at the top of the screen when you first enter the OPERLOG panel.
- There is no job number and data set number for the OPERLOG.
- There are no absolute line numbers since the log stream is essentially very large in size.

```
  Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------------------------------------------------
SDSF OPERLOG  DATE 04/17/95         4 WTORS                                                          COLUMNS 1- 120
COMMAND INPUT ===>                                                                                  SCROLL ===> HALF
NC0000000 SC55     95107 11:54:41.60 TRAUNER  00000290 RO *ALL,$D MASDEF,AUTOEMEM
NC0000000 SC55     95107 11:54:41.65 *ROUT089 00000290 RO T=005,SC55,$D MASDEF,AUTOEMEM
NC0000000 SC47     95107 11:54:41.67 *ROUT089 00000290 RO T=005,SC47,$D MASDEF,AUTOEMEM
NC0000000 SC47     95107 11:54:41.68 *ROUT089 00000290 $D MASDEF,AUTOEMEM
NR0000000 SC47     95107 11:54:41.69 *ROUT089 00000090 $HASP843 MASDEF  AUTOEMEM=ON
NC0000000 SC52     95107 11:54:41.68 *ROUT089 00000290 RO T=005,SC52,$D MASDEF,AUTOEMEM
NC0000000 SC54     95107 11:54:41.68 *ROUT089 00000290 RO T=005,SC54,$D MASDEF,AUTOEMEM
NC0000000 SC49     95107 11:54:41.68 *ROUT089 00000290 RO T=005,SC49,$D MASDEF,AUTOEMEM
NC0000000 SC53     95107 11:54:41.69 *ROUT089 00000290 RO T=005,SC53,$D MASDEF,AUTOEMEM
NC0000000 SC50     95107 11:54:41.67 *ROUT089 00000290 RO T=005,SC50,$D MASDEF,AUTOEMEM
NC0000000 SC54     95107 11:54:41.72 *ROUT089 00000290 $D MASDEF,AUTOEMEM
NC0000000 SC55     95107 11:54:41.71 *ROUT089 00000290 $D MASDEF,AUTOEMEM
NC0000000 SC52     95107 11:54:41.72 *ROUT089 00000290 $D MASDEF,AUTOEMEM
NC0000000 SC53     95107 11:54:41.73 *ROUT089 00000290 $D MASDEF,AUTOEMEM
NC0000000 SC49     95107 11:54:41.72 *ROUT089 00000290 $D MASDEF,AUTOEMEM
NR0000000 SC55     95107 11:54:41.76 *ROUT089 00000090 $HASP843 MASDEF  AUTOEMEM=ON
NR0000000 SC54     95107 11:54:41.77 *ROUT089 00000090 $HASP843 MASDEF  AUTOEMEM=ON
NR0000000 SC53     95107 11:54:41.76 *ROUT089 00000090 $HASP843 MASDEF  AUTOEMEM=ON
NR0000000 SC49     95107 11:54:41.76 *ROUT089 00000090 $HASP843 MASDEF  AUTOEMEM=ON
NR0000000 SC52     95107 11:54:41.77 *ROUT089 00000090 $HASP843 MASDEF  AUTOEMEM=ON
NC0000000 SC50     95107 11:54:41.89 *ROUT089 00000290 $D MASDEF,AUTOEMEM
NR0000000 SC50     95107 11:54:42.00 *ROUT089 00000090 $HASP843 MASDEF  AUTOEMEM=ON
MR0000000 SC55     95107 11:54:43.08 TRAUNER  00000090 IEE421I RO *ALL,$D MASDEF,AUTOEM 992
LR                                      992 00000090 SYSNAME  RESPONSES ---------------------------------------------
DR                                      992 00000090 SC47     $HASP843 MASDEF  AUTOEMEM=ON
DR                                      992 00000090 SC49     $HASP843 MASDEF  AUTOEMEM=ON
DR                                      992 00000090 SC50     $HASP843 MASDEF  AUTOEMEM=ON
DR                                      992 00000090 SC52     $HASP843 MASDEF  AUTOEMEM=ON
DR                                      992 00000090 SC53     $HASP843 MASDEF  AUTOEMEM=ON
DR                                      992 00000090 SC54     $HASP843 MASDEF  AUTOEMEM=ON
ER                                      992 00000090 SC55     $HASP843 MASDEF  AUTOEMEM=ON
8000000 SC47     08.41.36 STC04672 *157 DSI802A SCN47   REPLY WITH VALID NCCF SYSTEM OPERATOR COMMAND
8000000 SC47     08.36.25          *154 DSI802A SCI47   REPLY WITH VALID NCCF SYSTEM OPERATOR COMMAND
8000000 SC49     12.52.13          *075 DSI802A SCI49   REPLY WITH VALID NCCF SYSTEM OPERATOR COMMAND
8000000 SC55     10.32.15          *977 DSI802A SCI55   REPLY WITH VALID NCCF SYSTEM OPERATOR COMMAND
****************************************** BOTTOM OF DATA ********************************************************
```

Figure 50. An OPERLOG Display Example

Chapter 8. JES3 5.2.1 Sysplex Operations

MVS/ESA JES3 Version 5 Release 2.2 improves systems management and automated operations by enabling the sysplex operations features of MVS/ESA. System and subsystem code can now build on these features without the need to import function or provide alternate implementations which were required in the pre-JES3 5.2.1 environment.

Sysplex-wide operation control is enabled and available from consoles on any system rather than a subset of the consoles specified to JES3 on the global processor. JES3 5.2.1 command processors are also enhanced to preserve the command and response token (CART) and appropriately identify command responses, enabling improved automated operations.

Optionally, the use of the MVS operations log (OPERLOG) and the system logger provides a sysplex-wide log containing more information than the current JES3 DLOG (JES3 log format written to SYSLOG).

This chapter describes briefly the changes made to JES3 5.2.1 in support of the MCS multisystem operations.

8.1 JES3 5.2.1 Console Enhancements

JES3 operations support is significantly enhanced to exploit the MCS multisystem operations capabilities. The JES3 managed locally attached consoles are not required any longer to manage a JES3 complex and therefore their support has been removed. Note that the JES3 unique operations features, such as functional message routing, are preserved in cases where there is no equivalent MCS support.

In the JES3 5.2.1 environment, MCS sysplex is enabled and provides sysplex-wide control from any console on any system in a sysplex:

- Command and message transportation between systems is controlled by MCS. All console types; system console, master console, MCS and EMCS consoles, can send MVS and JES3 commands to any system and receive messages from any system in a sysplex.
- The JES3 message retention facility (JMRF) is removed. MCS maintains a sysplex-wide view of retained messages through its action message retention facility (AMRF).
- The MCS command prefixing facility (CPF) is enabled sysplex-wide in a JES3 5.2.1 complex. By using the appropriate prefix, commands can be routed to all subsystems that exploit this facility. JES3 registers command synonyms with CPF such that JES3 global commands can be entered from anywhere in the sysplex and they are routed to the global for processing. CPF is also used to register local prefixes for JES3 commands, such as *RETURN, that must execute on a local processor.
- Subsystems and applications can exploit without restrictions all MCS facilities (for example, CART and 4-byte console ID). The JES3 command processors support 4-byte console IDs and the MCS command-and-response token (CART).

The *INQUIRY and *MODIFY command processors identify command responses by using the WTO command response descriptor code. Some JES3 inquiry command responses are changed to include a summary message, so that the end of the command response can be easily identified by automation applications. These commands are: *I A, *I G, *I P, and *I Q.

JES3 5.2.1 has also begun to exploit MCS facilities. Extended MCS consoles are used to implement JES3 DLOG, RJP, and NJE consoles, thus making MCS responsible for queuing messages to all consoles.

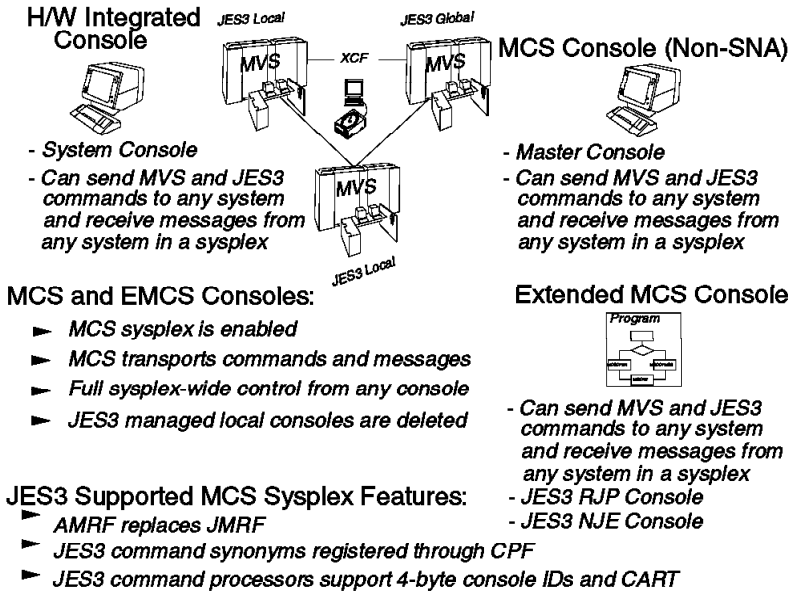


Figure 51. JES3 5.2.1 and MCS Sysplex

8.2 JES3 5.2.1 Operational Changes

The JES3 sysplex operations enhancements allow customers to realize the benefits of other MVS functions and products that depend on an MCS sysplex environment. Figure 51 summarizes the JES3 5.2.1 console enhancement changes.

The following is a list of the major functional changes introduced in the JES3 5.2.1 operations enhancements:

- JES3 operator consoles no longer supported.
- JES3 no longer requires the suppression of the MCS multisystem processing, but instead, relies on MCS to transport command and message traffic between systems in a sysplex. Operators can control the sysplex operations from any console regardless of where it is attached. This includes extended MCS consoles such as TSO consoles, or NetView consoles.

  Because MCS takes over the command and message transportation, JES3 is no longer involved with the transportation of messages for display and does not reissue WTOs from local processors on the global to display the messages on MCS consoles attached to the global.

- JES3 continues to log messages in a job's JESMSGLG data set.
- JES3 continues to provide functional message routing through the following existing interfaces:
  - MSGROUTE initialization statement
  - Device-related message routing through JES3 destination class specification on the DEVICE initialization statement
  - Writer FSS message routing through destination class specification on the FSSDEF initialization statement
- JES3 5.2.1 IATXMLWO and MESSAGE macro services allow JES3 functions to issue true multi-line WTOs. The IATXMLWO macro creates one line of a multi-line message. Each line of a multi-line message is stored in its own copy (IATYMLWO token). These lines (tokens) are chained together and sent to the MESSAGE macro to issue the multi-line WTO message. A multi-line WTO message is limited to a maximum of 999 lines.
- The JES3 SVC 34 SSI interface module has been changed to allow JES3 commands to be up to 126 characters (instead of 80 characters). In JES3 5.2.1, output service command processing (*I U and *F U) and the *CALL JMF command accept 126 character input. Other command processors are still restricted to the maximum of 80 character command length. In addition, commands may be stacked (multiple commands on a single line) up to 126 characters, provided each command in the stack adheres to its length restriction.

In JES3 releases prior to JES3 5.2.1, a JES3 or non-JES3 (MVS or some other subsystem) command is sent to a targeted processor using the JSERV or SSISERV macro service. The modifier codes in the MOD= parameter on the JSERV or SSISERV macro specifies the function (destination queue modifier) to be invoked to process the command at its destination. JES3 5.2.1 deletes the following modifier codes for the SVC 34 destination queue:

MODINTJS Since JES3 does not manage command transportation and it does not support the *SEND command to transport a JES3 command to another system, the modifier code MODINTJS is deleted.

MODS34J Since JES3 does not manage the command transportation and it does not use the *SEND command to transport a system command to another system, the modifier code MODS34J is deleted.

Note: All installation programs that are using the JSERV or SSISERV macros to communicate with JES3 should be changed to use extended MCS consoles and MGCRE macro service.

8.3 JES3 5.2.1 Command Processing

Figure 52 illustrates the command processing flow in JES3 5.2.1. All commands go through the MCS MGCRE service, which serves as the single point of command entry. The MCS MGCRE processing invokes the MPF command exits prior to entering the SSI loop for the command. JES3 5.2.1 SSI processing (IATSI34) continues to examine input commands to see if they are valid JES3 commands. Valid JES3 command are sent to JES3 through the SSISERV service and further MGCRE processing is terminated.

JES3 commands are accepted from the same sources as in prior JES3 releases except for the JES3 locally attached consoles, which are deleted. Also, support

for JES3 command entry through a BDT session is deleted. The JES3 command authorization exit (IATUX56) for commands entered through BDT is also deleted.

JES3 DSPs can internally issue JES3 commands through the INTERCOM macro (simulate input of operator command). The INTERCOM macro is converted into an MCS MGCRE.

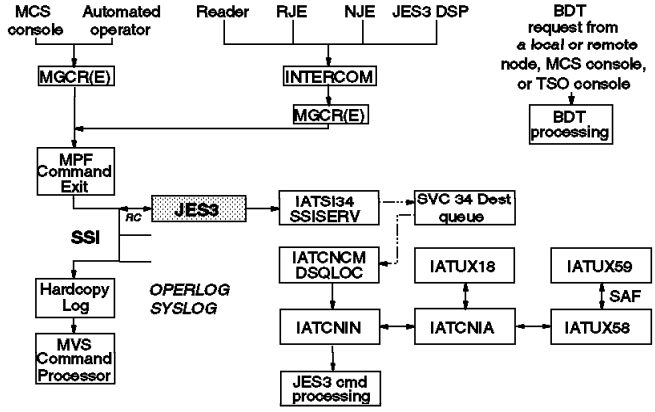


Figure 52. JES3 5.2.1 Command Processing

The JES3 5.2.1 processing for commands entered from an MCS console or an automated operator interface remains the same. When multiple commands are entered through a single input from an MCS console, MCS unstacks the commands and sends one command at a time to the SSI. When multiple commands are entered through a single MGCRE by an automated operator, MCS sends the whole stack of commands to the SSI. If the first command is an MVS command, the JES3 SSI routine returns the whole stack of commands back to MCS for processing. If the first command is a JES3 command, the JES3 SSI routine sends the whole stack of commands to the JES3 address space for processing, where the JES3 5.2.1 command processing continues to provide the unstacking service as in prior releases of JES3. Note that JES3 5.2.1 uses the MCS command delimiter defined in the CONSOLxx parmlib member instead of the one defined on the EDIT parameter in the CONSTD initialization statement.

8.4 JES3 5.2.1 Command Authorization and Exits

The JES3 command authorization process includes invocation of the security authorization facility (SAF). Before JES3 calls SAF to perform security processing, exit IATUX58 is given control. This exit allows you to modify security checks or to make your own security decisions for JES3. After the SAF call, JES3 calls exit IATUX59. Also this exit gives you the opportunity to modify security checks or to make security decisions for JES3.

Note: IBM recommends that you use a security product such as RACF for command authorization instead of exit code whenever possible. When exit code is required, MPF command exits should be considered first. Use IATUX18 only for code that must run in the JES3 address space.

After successful completion of the command authorization checks, the command is passed to the appropriate command executor.

In JES3 5.2.1, exit IATUX18 is *not* entered for MVS commands issued from a JES3 source; it is entered only for JES3 commands.

During the JES3 5.2.1 migration, current IATUX18 installation exit code related to non-JES3 commands entered from JES3 sources must be moved to an MPF command installation exit. These exits are still running in the JES3 address space, and JES3 address space data, such as device group, continues to be available for the exits. In summary, it is recommended to continue to have all installation JES3 command processing in MPF command installation exits or in the JES3 command exit, and move all installation non-JES3 command processing into MPF command installation exits (or other facilities).

8.5 JES3 5.2.1 Message Processing

As JES3 5.2.1 enables the MCS sysplex, MCS controls the flow of messages between systems in a JES3 sysplex. JES3 no longer transports messages to the global processor for display on MCS consoles. Support for the JES3 managed locally attached consoles is also removed. Message processing is determined by the following functions:

- JES3 issues all DSP messages through the WTO and WTOR macro service.
- MCS provides MPF processing on the originating system that issued the message.
- Prior to JES3 5.2.1, JES3 transported messages originating on a local JES3 processor to the global and reissued the messages (WTO) for display on MCS consoles attached to the global. Some JES3 installations rely on the JES3 global processor to be a “focal point” for MPF processing. To accommodate the “global-oriented” MPF processing, JES3 5.2.1 provides an option (GLOBMPF=YES on CONSTD initialization statement) to pass all messages routed to the global processor to MPF processing on the global. This is in addition to MPF processing on the originating system.
- Exits 69 and 70 allow you to implement JES3 environment specific message processing in the global JES3 address space.
- Installation exit IATUX31 is deleted.
- The JES3 functional message routing is preserved.
- MCS ARMF for message retention is required, as JMRF is deleted.

8.5.1 MPF Message Processing

Some JES3 installations have come to rely on the JES3 global processor to be a “focal point” for message processing and have implemented extensive MPF processing on the global for messages that originate on local processors. To accommodate “global-oriented” MPF processing that an installation may have, an option (GLOBMPF=) is provided on the CONSTD initialization statement.

Note that the GLOBMPF option does not influence the routing of messages to the global processor. Installations wishing to use the GLOBMPF option must ensure that routing mechanisms are in place to guarantee the global is presented the proper set of messages. This could include the activation of DLOG, which results in the hardcopy message set being presented to the global, or the definition of a physical console or extended MCS console on the global that receives the proper set of routing codes (or all routing codes).

Note: Because JES3 5.2.1 does not transport messages to the JES3 global address space for display and logging purposes, two JES3 installation exits, exits 69 and 70, are provided to accommodate special message processing that is dependent on running in the JES3 address space and having access to JES3-maintained information.

8.5.2 Functional Message Routing

JES3 functional message routing is currently a way to specify routing information for messages that pertain to some functional area or that are associated with a specific device. The routing information is provided directly on the JES3 MESSAGE macro for messages issued by JES3 DSPs. The message routing is provided during the JES3 WTO SSI processing based on MSGROUTE initialization statement specifications for messages that are not directly issued by JES3 functions.

The JES3 functional message routing information is specified on the initialization stream parameters. The routing information on the various JES3 initialization statements may be specified either as a JES3 console destination class or as a single MCS routing code. The continued support of destination classes ensures that existing initialization statements are accepted during migration to JES3 5.2.1.

8.5.3 JES3 Action Message Retention Facility

JES3 5.2.1 deletes the JES3 action message retention facility (JMRF) and the *I R commands used to display the JMRF messages. JMRF was used to maintain a queue of operator action messages. Messages retained by JMRF included messages issued by JES3 DSPs, messages issued by jobs, and messages issued by other subsystems and MVS components.

The MCS action message retention facility (AMRF) takes over the function provided by JMRF. The MCS DISPLAY R command must be used to display outstanding messages requiring operator action. AMRF retains action messages issued when JES3 is not initialized, and action messages issued from programs running under the MASTER subsystem which the JMRF could not do.

JES3 in previous versions retained messages with descriptor codes 1 or 2. You can tailor the AMRF set of retained action messages with the

```
.DEFAULT
.NO_ENTRY
```

statements and the RETAIN(YES|I,E,CE|NO) parameters in the MPFLSTxx parmlib member to correspond closely the JES3 JMRF function.

The default action messages that are retained by AMRF are messages with the following descriptor codes:

1 System failure

2 Immediate action

3 Eventual action

11 Critical eventual action

JES3 in previous versions retained messages with descriptor codes 1 or 2.

The MCS ARMF is activated through the ARMF(Y) setting on the INIT statement of the CONSOLxx parmlib member or through the CONTROL M,ARMF=Y operator command.

```
INIT  AMRF{(Y|N)
```

The message types retained by AMRF can be tailored using the MPFLSTxx RETAIN keyword. To be consistent with the messages retained in previous JES3 releases and have AMRF retain messages with descriptor codes 1 and 2, include the following statement in your MPFLSTxx member of SYS1.PARMLIB:

```
.NO_ENTRY RETAIN(I)
```

Note: The sequence numbering for action messages changes when the MCS sysplex is active. Therefore, consider specifying a larger range on the K C command when deleting messages from an MCS console, for example:

```
K C,A,1-999999
```

8.6 JES3 5.2.1 DLOG

JES3 5.2.1 provides a migration accommodation (DLOG) for customers who are unable to activate the MVS OPERLOG across the JES3 complex. A sysplex-wide log is written from the global processor in the JES3 DLOG format:

```
743584 -SO08= VAINI    D T
743587  SO08= VAINI    IEE136I LOCAL  TIME=17.43.58 DATE=1994.271  GMT  TIME=21.43.58 DATE=
743587  SO08= VAINI    1994.271
```

as opposed to the MVS format:

```
NC0000000 SC47     94271 17:39:05.03 VAINI    00000290  D T
NR0000000 SC47     94271 17:39:05.04 VAINI    00000090  IEE136I LOCAL: TIME=17.39.05 DATE..
SR                                                      DATE=1994.271
```

There are some content differences in the JES3 5.2.1 DLOG format compared with the pre-JES3 5.2.1 DLOG format as follows:

- When a message is to be issued to multiple routing codes, the single JES3 message destination class shown in the DLOG record may differ from that shown in prior releases. The destination class may not be the JES3 global routing determined on the issuing processor.
- For messages suppressed by MPF processing, the MPF suppression character in the log record will be the global processor's suppression character, rather than the character for the originating system. MPF suppression characters are defined in parmlib member MPFLSTxx using the MPFHCF= keyword. The default is an ampersand (&).
- The minimal/marginal spool space and JSAM buffer shortage flags will no longer appear with each message logged while the condition exists. These conditions can be identified using existing messages IAT1017, IAT1018, IAT1101, IAT1102, and IAT1103.

Note: JES3 MLOG (hardcopy printer) support is removed in JES3 5.2.1. Also, JES3 exit IATUX31 (Examine/Modify Destination or Message Text) is removed and thus no longer affects logging decisions.

The DLOG message traffic is managed by MCS. On the global, JES3 activates an extended MCS console with the HARDCOPY=YES attribute to receive the hardcopy message set, as shown in Figure 53. The messages received by the DLOG EMCS console are in the MDB format and are converted to the JES3

DLOG format and then written using the WTL macro service to the global JES3 system's SYSLOG (alias DLOG).

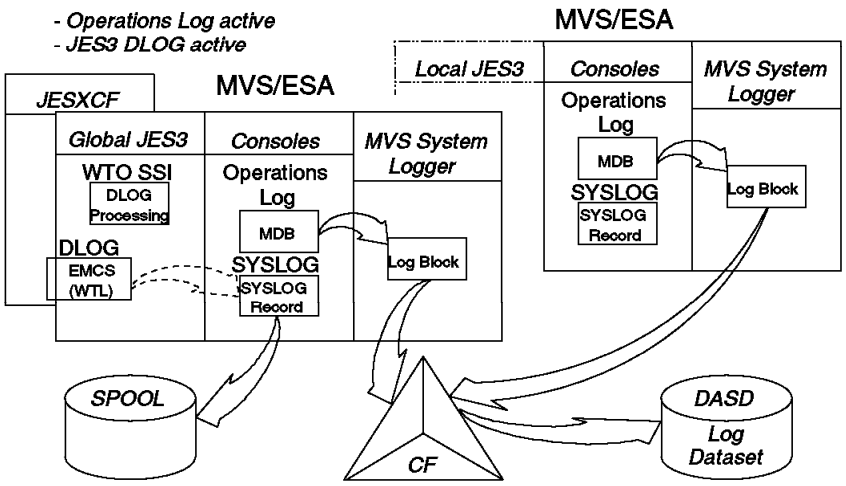


Figure 53. JES3 5.2.1 and MVS OPERLOG

The JES3 WTO subsystem interface routines receive control as part of the MCS WTO or WTL processing. If the DLOG is active, the JES3 WTO SSI processing suppresses the MCS logging of operator commands and WTO messages into the SYSLOG. JES3 never suppresses the MCS logging into the OPERLOG log stream.

Prior to JES3 5.2.1, JES3 initialization assigned SYSLOG unconditionally to be the MVS hardcopy medium (VARY SYSLOG,HARDCPY) on all systems. If a hardcopy log to a device was required (MLOG was requested in the initialization stream), it was written by JES3 to a JES3 managed device. In addition, if DLOG was specified in the initialization stream, and the JES3 global was restarting without IPL, JES3 ensured that the log was active on the global processor (WRITELOG START). During local connect processing, JES3 caused the accumulated log data on each local to be written to spool (WRITELOG).

When JES3 5.2.1 is installed, customers have the option to use the OPERLOG as the only hardcopy media. Because JES3 can no longer unconditionally assign SYSLOG to the MVS hardcopy media (VARY SYSLOG,HARDCPY) on all systems, this processing is removed. However, if DLOG is requested in the initialization stream, JES3 global initialization attempts to ensure that SYSLOG is assigned the hardcopy media and that the SYSLOG task in the console address space is active.

The *I O,DLOG command displays the current DLOG status, and *F O,DLOG=ON or OFF activates or deactivates the DLOG function.

The JES3 DLOG runs in its own address space. This address space is started under the MSTR subsystem through an internally issued start command. Once initialized, the JES3 DLOG address space activates an extended MCS console (NAME=SYSJ3Dxx KEY=xcfgroupname) that is set to receive the hardcopy message set. The xx is a generated number that starts with 01.

8.6.1 JES3 5.2.1 NJE Console

NJE consoles provide a remote node the capability to inquire on and control work that has arrived from the NJE network. There are no physical JES3 NJE consoles, but instead the NJE console support provides a way for performing command association between a requestor on a remote node and console operations on the JES3 node. JES3 5.2.1 uses an internally created extended MCS console for NJE console processing.

The NJE extended MCS console name is SYSJ3Nxx with a KEY=xcfgroupname. The xx is a generated number that starts with 01.

DSI processing switches the console name from SYSJ3N01 to the next available console name (for example, SYSJ3N02).

8.6.2 JES3 5.2.1 RJP Consoles

SNARJP and BSCRJP consoles that are attached to remote job entry workstations can be either real console devices, or logical devices that are being simulated by an application. These consoles are usually used to control work that originates from or is associated with a location where a group of print devices reside.

JES3 5.2.1 internally creates an extended MCS console for RJP console processing. The RJP extended MCS console name is SYSJ3Rxx with a KEY=xcfgroupname. The xx is a generated number that starts with 01.

Note: Information for RJP workstation commands to identify the actual workstation from which the command was entered is passed to the JES3 command authorization exit (IATUX18) and is supplied by JES3 on the SAF authorization calls. Information passed by MCS to MPF command and message exits identify only the SYSJ3Rxx console, not the actual remote work station.

DSI processing switches the console name from SYSJ3R01 to the next available console name (for example, SYSJ3R02).

8.7 JES3 5.2.1 and MCS CPF

The MVS command prefix facility (CPF) allows a subsystem (like JES3 or JES2) to register system- or sysplex-wide command prefixes. An operator command that is entered with a registered CPF prefix is recognized by MCS and routed to the system where the prefix was defined. The CPF macro service allows you to:

- Define a command prefix by specifying the command prefix. It may specify a name that identifies the subsystem owning the command prefix, the scope (SYSPLEX or SYSTEM), the failure disposition, and whether the command prefix is to be removed from the command text prior to the command being executed on the receiving system.
- Delete an existing command prefix.
- Redefine an existing command prefix for a system or owner name.

CPF processing is enabled sysplex-wide in a JES3 5.2.1 complex. The CPF scope is no longer limited to a single system as it is in a sysplex running JES3 releases prior to JES3 5.2.1. JES3 registers the SYSPLEX (PLEXSYN=) and SYSTEM (SYN=) scope command prefixes that are defined on the JES3 CONSTD initialization statement, as shown in Figure 54.

Up to six sysplex-wide JES3 prefixes (also known as synonyms) can be specified. The sysplex-wide prefixes are always registered with CPF on the global processor and are re-registered during the dynamic system interchange.

Note: To avoid conflicts with short-form WTOR replies, the JES3 default SYSTEM scope prefix (8) is not registered with CPF and therefore is not displayed via the operator command *D OPDATA*.

8.7.1 Defining the JES3 Command Prefix

A new keyword, PLEXSYN, defines the sysplex command prefixes.

```
CONSTD,EDIT=({escape,bkspace,,linedel}),
       GLOBMPF={YES|NO},
       SYN={(syn1,...syn6)|8}
       PLEXSYN={(syn1,...syn6)|*},
       CIFSS={FSSDEF|MSGROUTE},
       DLOG={ON|OFF}
```

Figure 54. Command Prefix Definition on CONSTD Statement

PLEXSYN PLEXSYN={(syn1,...syn6)|*} - Specifies a sysplex scope for the command prefix. The sysplex scope means that any command issued with this prefix from any system in the sysplex executes on the global processor. The default is *. This keyword is used together with the SYN keyword to determine the prefix to be used. If a prefix is defined on both keywords, the prefix is used as a system scoped prefix.

8.7.2 SYSPLEX Scope Processing

JES3 5.2.1 registers with CPF sysplex-wide command prefixes that are defined with the PLEXSYN= parameter on the JES3 CONSTD initialization statement. Up to six sysplex-wide JES3 prefixes (also known as synonyms) can be specified. The sysplex-wide prefixes are always registered with CPF on the global processor and are re-registered during the dynamic system interchange. In general, commands routed explicitly with the MVS ROUTE command and with sysplex-wide prefixes are rerouted by MCS as required according to the sysplex-wide prefix definition after the ROUTE command routing completes. If a command is routed using a sysplex-wide prefix, MCS routes it only *once*, even if the command has multiple sysplex-wide prefixes.

JES3 defines both SYSTEM and SYSPLEX scope prefixes with the the option FAILDISP=PURGE. This means that a command prefix is deleted when the defining system is removed from the sysplex or the defining address space terminates. The CPF REMOVE=NO option is used by JES3 for the prefix removal. This implies that both the command prefix and the command are presented on the receiving system. The length of both SYSTEM and SYSPLEX scope prefixes can be one to eight characters.

If a prefix is specified on both the SYN= and PLEXSYN= parameter, JES3 defines it as a SYSTEM-wide (SYN=) prefix. If the JES3 global fails to define one or more of the prefixes specified on the PLEXSYN= parameter, a warning message is issued and the prefix is ignored. When every PLEXSYN= prefix registration fails, JES3 uses the default SYSPLEX-scope prefix * and issues a warning message stating that the default sysplex-wide prefix is being used.

Note: Even if * is the default SYSPLEX-wide prefix, it is recognized as a *valid JES3 command prefix on a local* when it is passed to command processing on a local. This happens, for example, if a * JES3 command is routed to a local using double prefixing. The * has preserved the *guaranteed-to-work* JES3 command prefix role for compatibility reasons. This way, the * can be used as the fixed JES3 command prefix by automation products. In this case, the command transportation should be implemented using double command prefixing.

If JES3 fails to register the * prefix, there will *not* be any JES3 sysplex-wide prefixes, and all JES3 commands from locals must be explicitly routed to the global using either the MVS ROUTE command or double command prefixes. The first of the double prefixes must be defined as “REMOVE=YES,” to route the command to the global. The second prefix should be the JES3 “*” or any of the SYSTEM-scope prefixes registered by the global.

Note: Double prefixes are allowed for commands. MCS uses the first prefix to transport the command. Once the command is transported, MCS then presents the command with the remaining prefix to the command processors on that system.

8.7.3 SYSTEM Scope Processing

The existing SYN= parameter on the CONSTD initialization statement continues to define up to six SYSTEM scope JES3 prefixes. The SYSTEM scope prefixes are registered on *each* system in the JES3 complex. Commands that begin with SYSTEM scope prefixes are processed on the processor on which the command is entered. Explicitly routed commands with SYSTEM-wide prefixes are processed on the target processor.

When JES3 is unable to register one or more of the SYSTEM-scope prefixes specified on the SYN= parameter on any processors, a warning message is issued. If all prefixes specified on the SYN= parameter fail to be defined to CPF on a processor, JES3 uses the default prefix value 8 on that processor and issues a warning message stating the fact.

Note: The 8 prefix is implemented as a SYSTEM-scope prefix but is not registered with CPF to avoid conflicts with short-form WTOR replies.

8.8 Multiple Globals in the Same Sysplex

IBM recommends that a sysplex include only one JES3 global processor. If you choose to run more than one global in the same sysplex, the considerations described in the following sections apply.

8.8.1 JES3 XCF Group Name Considerations

The XCF groupnames must be unique. This name can be specified on the OPTIONS initialization statement (XCFGRPNM= keyword) or can default to the home nodename. The XCF groupnames and CONSTD prefix specifications for the sample sysplex configuration are shown in Figure 55 on page 176.

8.8.2 Multiple Globals in the Same Sysplex

The * command prefix must be explicitly defined in each initialization stream as SYN=*, a SYSTEM-scoped prefix, as shown in Figure 55. Otherwise, any JES3 commands issued internally by JES3 may be processed by the wrong global since JES3 internally issues commands using the * prefix.

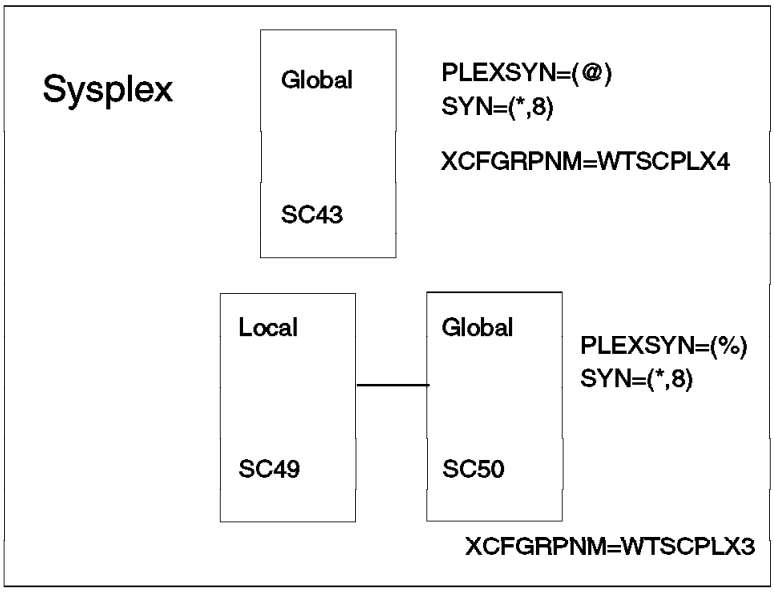


Figure 55. Multiple Globals in the Same Sysplex

If use of the character 8 as a prefix is also required, then you must explicitly define it on the SYN= parameter in addition to the * prefix. Figure 55 shows the prefixes for a sysplex with more than one global.

You should also specify a SYSPLEX-scoped prefix for each global. This allows any console in the sysplex to send commands to the global and receive the responses. Automation routines that may issue JES3 commands on other than the target global should use the appropriate SYSPLEX-scoped prefix.

The sample program IEECMDPF may also be used to define each system's name as a SYSPLEX-scoped command prefix. This allows the * prefix, in conjunction with the appropriate global system name, to be used to direct commands to any JES3 global in the sysplex (e.g. SC43*I Q).

Figure 56 shows the registered prefixes for the sysplex with more than one global.

```
D OPDATA
 IEE603I 22.08.03 OPDATA DISPLAY 444
          PREFIX     OWNER      SYSTEM     SCOPE     REMOVE   FAILDSP
          %          JES3       SC50       SYSPLEX   NO       PURGE
          *          JES3       SC50       SYSTEM    NO       PURGE
          *          JES3       SC49       SYSTEM    NO       PURGE
          @          JES3       SC43       SYSPLEX   NO       PURGE
          *          JES3       SC43       SYSTEM    NO       PURGE
```

Figure 56. Display of Prefixes for the Sysplex

8.8.2.1 Using Multiple Prefixes

The following command is issued on a MCS console attached to the local processor SC49:

SC508I Q

The SC508I Q command is an example of double prefixing. The first prefix SC50 transports the command to the global where it is removed. The rest of the command is passed to the SSI, where JES3 recognizes its SYSTEM-scope default prefix 8 and executes the 8I Q command.

```
SC508I Q
 IAT8674 JOB NJERDR   (JOB00093) P=15 NJERDR(ACTIVE)
 IAT8674 JOB INTRDR   (JOB05875) P=15 INTRDR(ACTIVE)
```

The following command is issued on a MCS console attached to the global processor SC50:

SC49*I Q

The SC49*I Q command is another example of double prefixing. The first prefix SC49 transports the command to a local where it is removed. The rest of the command is passed to the SSI, where JES3 recognizes its “guaranteed-to-work” prefix * and tries to execute the *I Q command.

```
SC49*I Q
 IAT7130 '*I Q      ' REJECTED, NOT VALID ON LOCAL OR GLOBAL IS NOT AVAILABLE
```

The command is rejected because this command type is not valid on a local processor.

Appendix A. IHAHCLOG Macro

The following macro maps the hardcopy log message prefix and can be used in this redbook to understand the messages shown in 2.1.3.1, “Hardcopy Log Messages” on page 19.

```
         MACRO
         IHAHCLOG
*/***START OF SPECIFICATIONS*****************************************/
*/*                                                                  */
*/* MACRO NAME = IHAHCLOG                                            */
*/*                                                                  */
*/* DESCRIPTIVE NAME = HARDCOPY LOG FORMAT (HCL)                     */
*/*   ACRONYM = HCL                                               @P1A*/
*/*                                                               @P1A*/
*/* EXTERNAL CLASSIFICATION:  GUPI                                @P2A*/
*/* END OF EXTERNAL CLASSIFICATION:                               @P2A*/
*/*                                                                  */
*/*                                                                  */
*/* FUNCTION = MAPS HARDCOPY LOG RECORDS                             */
*/*                                                                  */
*/*                                                                  */
*/********************************************************************/
         SPACE
HCL      DSECT                  HARDCOPY LOG RECORD
HCLHEAD  EQU   *                HEADER INFORMATION
HCLRECID DS    OCL2             RECORD ID
HCLRECTP DS    C                RECORD TYPE
HCLWTO   EQU   C'N'             SINGLE-LINE MESSAGE
HCLWTOR  EQU   C'W'             SINGLE-LINE MESSAGE WITH REPLY
HCLMLWTO EQU   C'M'             FIRST LINE OF A MULTI-LINE MESSAGE
HCLLOG   EQU   C'O'             LOG COMMAND INPUT
HCLOTHER EQU   C'X'             ENTRY FROM A SOURCE OTHER THAN       *
                                HARDCOPY OR LOG COMMAND
HCLSPLIT EQU   C'S'             CONTINUATION OF PREVIOUS LINE
HCLLABEL EQU   C'L'             LABEL LINE OF A MULTI-LINE MESSAGE
HCLDATA  EQU   C'D'             DATA LINE OF A MULTI-LINE MESSAGE
HCLDTEND EQU   C'E'             DATA/END LINE OF A MULTI-LINE MESSAGE
         SPACE
HCLREQTP DS    C                REQUEST TYPE
HCLCMD   EQU   C'C'             COMMAND ISSUED BY OPERATOR
HCLRESP  EQU   C'R'             COMMAND RESPONSE
HCLINTNL EQU   C'I'             INTERNAL ISSUED COMMAND      @ZA79240
         EJECT
HCLROUTC DS    CL7              ROUTING CODES                    @L1C
         DS    C                BLANK
HCLSYSID DS    CL8              SYSTEM ID
         DS    C                BLANK
HCLDATE  DS    OCL5             JULIAN DATE OF MESSAGE  - YYDDD
HCLYEAR  DS    CL2              YEAR                      YY
HCLDAY   DS    CL3              DAY OF YEAR                 DDD
         DS    C                BLANK
HCLTIME  DS    OCL11            TIME MESSAGE WAS ISSUED - HH.MM.SS.TH
HCLHR    DS    CL2              HOURS                     HH
HCLCOLN1 DS    C                COLON                       :
HCLMIN   DS    CL2              MINUTES                      MM
HCLCOLN2 DS    C                COLON                          :
HCLSEC   DS    CL2              SECONDS                         SS
HCLDOT1  DS    C                DECIMAL POINT                     .
HCLTHSEC DS    CL2              .01 SECONDS                        TH
         DS    C                BLANK
HCLCONID DS    OCL8             ID OF CONSOLE THAT ISSUED COMMAND -  X
                                APPEARS ON FIRST/ONLY LINE OF        X
                                COMMANDS AND COMMAND RESPONSES
HCLJOBID DS    OCL8             ID OF JOB THAT ISSUED MESSAGE -      X
                                APPEARS ON FIRST/ONLY LINE OF OTHER  X
                                MESSAGES
         DS    CL5              RESERVED
HCLMLID  DS    CL3              MULTI-LINE MESSAGE ID - APPEARS ON   X
```

```
                                ADDITIONAL LINES OF MULTI-LINE       X
                                MESSAGES
         DS    C                BLANK
HCLREQFL DS    CL8              USER EXIT/MPF REQUEST FLAGS
         SPACE
*********************************************************************
*                                                                   *
*        WHEN THIS PRINTABLE HEX VALUE IS CONVERTED TO       @ZA79240*
*        BINARY, THE RESULTING BITS HAVE THE FOLLOWING       @ZA79240*
*        DEFINITIONS ASSOCIATED WITH THEM.                   @ZA79240*
*                                                                   *
*                                                                   *
*        REQUEST FLAGS BYTE 1                                @ZA79240*
*           BIT POSITION X'80' - MESSAGE TEXT WAS CHANGED    @ZA79240*
*           BIT POSITION X'40' - ROUTING CODES WERE CHANGED  @ZA79240*
*           BIT POSITION X'20' - DESCRIPTOR CODES WERE       @ZA79240*
*                                CHANGED                     @ZA79240*
*           BIT POSITION X'10' - MESSAGE WAS QUEUED TO A     @ZA79240*
*                                PARTICULAR ACTIVE CONSOLE   @ZA79240*
*           BIT POSITION X'08' - MESSAGE WAS QUEUED TO A     @ZA79240*
*                                PARTICULAR CONSOLE          @ZA79240*
*                                UNCONDITIONALLY             @ZA79240*
*           BIT POSITION X'04' - MESSAGE WAS QUEUED BY       @ZA79240*
*                                ROUTING CODES ONLY          @ZA79240*
*           BIT POSITION X'02' - THE CONSOLE ID TO WHICH THE @ZA79240*
*                                MESSAGE WAS QUEUED, WAS     @ZA79240*
*                                CHANGED                     @ZA79240*
*           BIT POSITION X'01' - MINOR LINES WERE PROCESSED  @ZA79240*
*                                                                   *
*        REQUEST FLAGS BYTE 2                                @ZA79240*
*           BIT POSITION X'80' - MESSAGE WAS DELETED         @ZA79240*
*           BIT POSITION X'40' - MPF SUPPRESSION OVERRIDDEN  @ZA79240*
*           BIT POSITION X'20' - MESSAGE WAS FORCED TO       @ZA79240*
*                                HARDCOPY                    @ZA79240*
*           BIT POSITION X'10' - MESSAGE BYPASSED HARDCOPY   @ZA79240*
*           BIT POSITION X'08' - MESSAGE WAS FORCED TO       @ZA79240*
*                                HARDCOPY ONLY               @ZA79240*
*           BIT POSITION X'04' - MESSAGE WAS BROADCASTED TO  @ZA79240*
*                                ACTIVE CONSOLES             @ZA79240*
*           BIT POSITION X'02' - BROADCASTING OF MESSAGE WAS @ZA79240*
*                                TURNED OFF                  @ZA79240*
*           BIT POSITION X'01' - MESSAGE WAS NOT RETAINED BY @ZA79240*
*                                AMRF DUE TO A USER EXIT'S   @ZA79240*
*                                REQUEST                     @ZA79240*
*                                                                   *
*        REQUEST FLAGS BYTE 3                                @ZA79240*
*           BIT POSITION X'80' - MESSAGE WAS RETAINED BY     @ZA79240*
*                                AMRF DUE TO A USER EXIT'S   @ZA79240*
*                                REQUEST                     @ZA79240*
*           BIT POSITION X'40' - CHANGE THE RETRIEVAL KEY        @L2A*
*           BIT POSITION X'20' - CHANGE THE 4-BYTE CONSOLE ID    @L2A*
*           BIT POSITION X'10' - CHANGE THE MESSAGE TYPE FLAGS   @L2A*
*           BIT POSITION X'08' - AUTOMATION IS NOT REQUIRED      @L2A*
*           BIT POSITION X'04' - AUTOMATION IS REQUIRED          @L2A*
*                                AND/OR AUTOMATION TOKEN         @L2A*
*           BIT POSITION X'02' - MESSAGE WAS ISSUED AS       @YA22959*
*                                HARDCOPY ONLY               @YA22959*
*           BIT POSITION X'01' - MESSAGE WAS A UD MESSAGE        @L2A*
*                                                                   *
*        SUPPRESSION FLAGS BYTE 4                                   *
*           BIT POSITION X'80' - MESSAGE NOT SERVICED BY ANY @ZA79240*
*                                WTO USER EXIT               @ZA79240*
*           BIT POSITION X'40' - ESTAE ERROR IN IEAVX600     @ZA79240*
*           BIT POSITION X'20' - MESSAGE NOT SERVICED BECAUSE @ZA79240*
*                                OF AN INCOMPATIBLE REQUEST  @ZA79240*
*           BIT POSITION X'10' - AUTOMATION REQUESTED        @YA06920*
*           BIT POSITION X'08' - MESSAGE NOT QUEUED TO ANY   @ZA79240*
*                                CONSOLE                     @ZA79240*
*           BIT POSITION X'04' - MESSAGE SUPPRESSED BY A     @ZA79240*
*                                SUBSYSTEM                   @ZA79240*
*           BIT POSITION X'02' - MESSAGE SUPPRESSED BY A     @ZA79240*
*                                WTO USER EXIT ROUTINE       @ZA79240*
*           BIT POSITION X'01' - MESSAGE SUPPRESSED BY MPF   @ZA79240*
*                                                                   *
*********************************************************************
```

```
         SPACE
         DS    C                 BLANK
HCLHEADL EQU   *-HCLHEAD         LENGTH OF HEADER
HCLTEXT  EQU   *                 MESSAGE OR COMMAND TEXT
         SPACE
```

Appendix B. Defining Command Resource Names to RACF

The RACF OPERCMDS class controls who can issue operator commands (for example, JES and MVS, and RACF operator commands). The OPERCMDS class is defined as a RACLIST class to optimize performance. To define command resource names to the RACF OPERCMDS class, the following steps can be followed:

1. To create a RACF profile for a *command resource name* into the OPERCMDS class, issue the RDEFINE RACF command:

   ```
   RDEFINE OPERCMDS command_resource_name UACC(NONE) [WARNING]
   ```

   The RACF resource name profile can be generic or specific. A generic profile can protect several resources that have a similar naming structure and security requirements. Specify generic characters in the profile name if you want to protect more than one resource with the same security requirements.

   Before you define generic profiles, issue the following command:

   ```
   SETROPTS GENERIC(OPERCMDS)
   ```

   You might want to include the WARNING option in the beginning. The WARNING option allows access to the resource even if access authority is insufficient. RACF issues a warning message and also records the access attempt in the SMF record if logging is specified in the profile.

   Later, when you have verified that all intended users have proper access only to the set of commands they should be able to use, you should remove the WARNING option.

2. To permit operators or groups of operators (users logged on to MCS consoles or extended MCS console users) to successfully execute commands controlled by the command_resource-name profile in the OPERCMDS class, issue the RACF command:

   ```
   PERMIT command_resource_name CLASS(OPERCMDS)
   ID(user_group) ACCESS(UPDATE)
   ```

   Note: User_group must be the name of a RACF-defined user or group profile.

3. If the OPERCMDS class is not already active, issue the SETROPTS command as follows:

   ```
   SETROPTS CLASSACT(OPERCMDS)
   ```

   To verify that the OPERCMDS class is active, you can issue the SETROPTS LIST command.

4. To have changes take effect after defining a generic profile or updating existing profiles, issue SETROPTS RACLIST:

   ```
   SETROPTS RACLIST(OPERCMDS) REFRESH
   ```

See also *RACF V2 Security Administrator's Guide*.

Appendix C. Controlling MCS Console LOGONs through RACF

Your MVS system may have been set up to require operators to log on to and log off from MCS-managed consoles through the LOGON option specification on the DEFAULT statement in CONSOLxx parmlib member.

When the RACF CONSOLE class is active and a console being used is protected by a profile in the CONSOLE class, RACF ensures that the person attempting to log on to an MCS console has the proper authority to do so.

Note: The following RACF options do not apply to MCS consoles:

- The SETROPTS TERMINAL command does not set universal access for consoles. You can specify this option with the TERMINAL operand of the SETROPTS command to set the universal access authority (UACC) that RACF uses when users attempt to log on to TSO from RACF-protected terminals.
- The group terminal option, which controls LOGON to TSO from RACF-protected terminals for users of the group, is not used by MCS console LOGON processing.
- You can require that a user be logged onto a particular terminal by specifying WHEN(TERMINAL(...)). You cannot specify the WHEN operand on the RDEFINE and RALTER commands for profiles in the CONSOLE class.

  However, you can require that a user be logged onto a particular console by specifying WHEN(CONSOLE(...)) on the PERMIT command.

To control the use of MCS consoles, take the following steps:

1. Create a profile for each console using the RACF RDEFINE command:

   ```
   RDEFINE CONSOLE conname UACC(NONE) [WARNING]
   ```

   The *conname* must match with the NAME(conname) specified on the CONSOLE statement in the CONSOLxx parmlib member. The NAME specifies the symbolic console name that uniquely identifies the console.

   It is strongly recommended that you specify names for all your consoles defined in CONSOLxx. If you do not name MCS consoles, MCS assigns a default name (EBCDIC console ID) to it. This name may change from IPL to IPL.

   You might want to include the WARNING option in the beginning. The WARNING option allows access to the resource even if access authority is insufficient. RACF issues a warning message and also records the access attempt in the SMF record if logging is specified in the profile.

2. If console LOGON is required or automatic as specified in the CONSOLxx parmlib member, use the PERMIT command to allow users and groups to use the console. You must give users at least READ access authority to the console. Otherwise, the users are not authorized to log on to the console.

   ```
   PERMIT conname CLASS(CONSOLE) ID(user_group) ACCESS(READ)
   ```

   Note: After you define a console with UACC(NONE) and no WARNING option, and activate the CONSOLE class, no one can log on to the console until you grant users access authority to the console profile.

3. When you are ready to start console protection, activate the CONSOLE class. The CONSOLE class allows, but does not require the class be RACLISTed. You may activate RACLIST processing for the class, which helps ensure high

performance when access authorities are checked. You can do these two actions in one command:

```
SETROPTS CLASSACT(CONSOLE) [RACLIST(CONSOLE)]
```

Appendix D. Display System Groups Command Exit

The following sample MCS command exit displays on request (GSYS operator command) the system groups defined through the IEEGSYS sample program. It also demonstrates how to use the CMDAUTH service to verify an operator's authority to use the command.

Internet access to code

Internet access to source code found in GG24-4626 can be accessed via the Internet.

http://www.redbooks.ibm.com/redbooks

Download sample code from our redbooks.

or you can:

You can access the server by anonymous FTP:

ftp ftp.almaden.ibm.com
user: anonymous
password: your E-mail address
cd redbooks/SG244626
get gsys.code

```
        TITLE 'MPF COMMAND EXIT GSYS - DISPLAY SYSTEM GROUPS'
CMDGSYS  CSECT
CMDGSYS  AMODE 31                      31 BIT ADDRESSING MODE
CMDGSYS  RMODE ANY                     31 BIT RESIDENCE
*
*  Function:
*    Display SYSTEM GROUPS defined through IEEGSYS program
*  Invocation:
*    GSYS command on an operator console.
*    - Note: The MPF .CMD statement should be updated to include
*      the CMDGSYS exit before the GSYS command can be
*      used.
*  Warning:
*    This sample program uses non-GUPI data areas!
*  Disclaimer:
*    This sample source is provided for tutorial purposes only. A
*    complete handling of error conditions has not been shown or
*    attempted, and this source has not been submitted to formal
*    IBM testing. This source is distributed on an 'as is' basis
*    without any warranties either expressed or implied.
*  Operation:
*   - Use CMDAUTH macro to verify RACF authorization of the user:
*     - The entity name is 'CMDX.GSYS'
*     - UTOKEN is passed through CMDXUTOK input parameter field
*     - READ access is required
*   - Locate system groups (defined by IEEGSYS sample program)
*   - For each system group add a line in to a multi-line WTO
*   - After all groups are processed, issue the multi-line WTO
*
*  LE ATTR: RENT REUS REFR
*
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
CMDXPTR  EQU   5
R5       EQU   5
BUFFPTR  EQU   6
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
RBASE    EQU   12
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
* ------
         BAKR  R14,0                   SAVE CALLER'S REGISTERS
         LR    RBASE,R15               ESTABLISH MODULE BASE
         USING CMDGSYS,RBASE
         L     CMDXPTR,0(R1)           GET CMDX ADDRESS
         USING CMDX,CMDXPTR            ACCESS THE CMDX
         L     BUFFPTR,CMDXCLIP        GET THE COMMAND BUFFER ADDRESS
         USING CMDXCLIB,BUFFPTR        ACCESS THE BUFFER
         LA    R2,CMDXCMDI             ACCESS START OF TEXT
         SLR   R8,R8                   DITTO
         ICM   R8,3,CMDXCMDL           GET CMD LTH
         BZ    RC0                     NOTHING - SPLIT
         CH    R8,=H'4'                COMMAND LENGTH < 4?
         BL    RC0                     YES - EXIT
         BE    *+4+6+4                 EQUAL - CHECK MORE
         CLC   =CL5'GSYS',0(R2)        GSYS COMMAND?
         BNE   RC0                     NO - EXIT
         CLC   0(4,R2),=CL5'GSYS'      GSYS COMMAND?
         BNE   RC0                     NO - NOTHING DO
* ------ PROCESS GSYS COMMAND
         LA    R0,DYNE-DYN
         GETMAIN  RU,LV=(0),SP=230     OBTAIN DYNAMIC STORAGE
         LR    R11,R1                  SAVE DYNAMIC AREA ADDRESS
         USING DYN,R11
         XC    DYN(DYNWL-DYN),DYN      CLEAR
         LA    R13,DYNSA               POINT AT SAVE AREA
         MVC   DYNSA+4(4),=CL4'F1SA'   *CHAIN*
* ------ VERIFY ISSUER'S AYTHORITY
*
* To verify the user's command authorization, the CMDAUTH service
```

```
* is invoked.  The service provides a return code that indicates
* the user's authorization status.  Under certain conditions, the
* command authorization service is unable to make a decision.
* In those cases the user's console authority is used to verify
* the command authority.
*
* Return codes from CMDAUTH service:
*     00  -  Command issuer is authorized to issue the command
*     04  -  No authorization decision was made.
*     08  -  Command issuer is not authorized to issue the command.
*
         ICM   R9,15,CMDXUTOK          POINT AT UTOKEN
         CMDAUTH ENTITY=ENTITY,ATTR=READ,UTOKEN=(R9),LOGSTR=LOGSTR,
               MF=(E,DYNCAP)
         CH    R15,=H'4'               RC = 0 - OK, 4 - DON'T KNOW,
                                            8 - FAIL
         BL    OK
         BH    FAIL
* ------ NO RACF DECISION - REQUIRE CONS COMMAND AUTHORITY
         TM    CMDXAFLA,CMDXACON       CONS command authority?
         BZ    FAIL                    NO - FAIL
         B     OK                      YES - OK
* ------ NOT AUTHORIZED
FAIL     DS    0H
         LA    R15,2                   ISSUER NOT AUTHORIZED
         B     FREE
* ------ LOCATE SYSTEM GROUPS AND SYSTEMS IN THE GROUPS
* /* Comments in REXX syntax */
OK       DS    0H
* cvt     = c2x( storage(10,4)       )
         L     R7,X'010'(,0)
* cvt_8c  = d2x( x2d(cvt) + x2d(8c)  )
* ecvt    = c2x( storage(cvt_8c,4)   )
         L     R7,X'08C'(,R7)
* ecvt_8c = d2x( x2d(ecvt) + x2d(8c) )
* ntth    = c2x( storage(ecvt_8c,4)  )
         L     R7,X'08C'(,R7)
* ntth_40 = d2x( x2d(ntth) + x2d(40) )
* ntte    = c2x( storage(ntth_40,4)  )
         ICM   R8,15,X'040'(R7)
* if x2d(ntte) = 0 then do;go  = 0;say "nothing";end
         BNZ   GSYSL
         BAL   R3,NOTHING
         B     RC4
* else go = 1
* do while ¬= 0
GSYSL    DS    0H
*   ntte_08 = d2x( x2d(ntte) + x2d(08) )
*   name    =      storage(ntte_08,16)
*   if substr(name,1,8) = "GSYS    " then do
         CLC   =CL8'GSYS',X'008'(R8)
         BNE    GSYSLC
*     ntte_1c = d2x( x2d(ntte) + x2d(1c) )
*     nttecsa = c2x( storage(ntte_1c,4)  )
         L      R9,X'01C'(,R8)
*     ntte_1b = d2x( x2d(ntte) + x2d(1b) )
*     nttecs# = x2d(c2x( storage(ntte_1b,1) ) )
         BAL    R3,CONTROL
         XR     R10,R10
         IC     R10,X'01B'(,R8)
*     say "** Systemgroup" substr(name,9) "("nttecs# "system(s))"
         MVC    DYNWA+2(L'GSYSNM),GSYSNM
         MVC    DYNWA+2+6(8),X'010'(R8)
         LA     R0,L'GSYSNM
         STH    R0,DYNWA
         BAL    R3,DATA
*     systems = "  "
         LTR    R10,R10
         BZ     GSYSLC
GSYSNL   DS    0H
         MVI    DYNWA+2,C' '
         MVC    DYNWA+3(80),DYNWA+2
*     do i = 0 to nttecs#
*      if i ¬= 0 then systems = systems storage(nttecsa,8)
*      nttecsa = d2x( x2d(nttecsa) + x2d(08) )
         LA     R2,DYNWA+2+6
         LA     R0,6+44
         STH    R0,DYNWA
         LA     R0,5
GSYSPN   DS    0H
         LA     R9,X'008'(,R9)
         MVC    0(8,R2),0(R9)
         LA     R2,X'009'(,R2)
         BCT    R10,GSYSDM
         BAL    R3,DATA
         B      GSYSLC
GSYSDM   DS    0H
         BCT    R0,GSYSPN
         BAL    R3,DATA
         B      GSYSNL
*     end
*     say systems
*   end
*   ntte_40 = d2x( x2d(ntte) + x2d(40) )
*   ntte    = c2x( storage(ntte_40,4)  )
**  if x2d(ntte) = 0 then go = 0
GSYSLC   DS    0H
 *       ICM   R8,15,X'040'(R8)
         BNZ   GSYSL
         LA    R3,RC4
         TM    DYNFLG,DYNFSG           HAVE SYSTEM GROUPS?
         BZ    NOTHING                 NO - SAY 'NOTHING'
         B     END                     END MLWTO
* end
* ------ CLEAN UP
RC4      DS    0H
         LA    R15,4                   EXIT PROCESSED THE COMMAND
FREE     DS    0H
         LR    R9,R15                  SAVE RC
         LA    R0,DYNE-DYN
         FREEMAIN RU,LV=(0),A=(R11),SP=230 FREE THE STORAGE
         LR    R15,R9                  RESTORE RC
         B     RET
* ------ RETURN ETC.....................
RC0      DS    0H
         XR    R15,R15                 SYSTEM TO PROCESS COMMAND
         B     RET
RET      DS    0H
         PR                            RETURN TO CALLER
* ------ ISSUE MESSAGES.....................
NOTHING  DS    0H
         LA    R6,MSG2                 POINT AT MESSAGE
         MVC   DYNWL(SWTOE-SWTO),SWTO  INIT WTO MF=L PLIST
         WTO   TEXT=(R6),CONSID=CMDXC4ID,MF=(E,DYNWL)
         BR    R3                      YES - RETURN
* ------
CONTROL  DS    0H
         TM    DYNFLG,DYNFSG           HAVE SYSTEM GROUPS?
         BNZR  R3
         OI    DYNFLG,DYNFSG           SET HAVE SYSTEM GROUPS
         LA    R6,MSGC                 POINT AT MESSAGE
         MVC   DYNWL(CWTOE-CWTO),CWTO  INIT WTO MF=L PLIST
         XR    R0,R0
         WTO   TEXT=(((R6),)),CONSID=CMDXC4ID,MF=(E,DYNWL)
         ST    R1,DYNCONN
         BR    R3                      YES - RETURN
* ------
DATA     DS    0H
         LA    R6,DYNWA                POINT AT MESSAGE
         MVC   DYNWL(DWTOE-DWTO),DWTO  INIT WTO MF=L PLIST
         XR    R0,R0
         WTO   TEXT=(((R6),)),                                         C
               CONNECT=DYNCONN,MF=(E,DYNWL)
         ST    R1,DYNCONN
         BR    R3                      YES - RETURN
* ------
END      DS    0H
         LA    R6,MSGDE                POINT AT MESSAGE
         MVC   DYNWL(DEWTOE-DEWTO),DEWTO  INIT WTO MF=L PLIST
         XR    R0,R0
         WTO   TEXT=(((R6),)),                                         C
               CONNECT=DYNCONN,MF=(E,DYNWL)
         BR    R3                      YES - RETURN
* ------
         DROP  R11 ,                   DYN
         LTORG
* ------ Data Areas
SWTO     WTO   TEXT=0,MF=L
SWTOE    DS    0C
CWTO     WTO   TEXT=((0,C)),MF=L
CWTOE    DS    0C
DWTO     WTO   TEXT=((0,D)),MF=L
DWTOE    DS    0C
DEWTO    WTO   TEXT=((0,DE)),MF=L
DEWTOE   DS    0C
* ------
```

```
MSGC     DC    AL2(34),CL34'UGSYS01I DEFINED SYSTEM GROUPS'
MSGDE    DC    AL2(34),CL34'END OF DEFINED SYSTEM GROUPS'
* ------
MSG2     DC    AL2(28),CL28'UGSYS02I NO SYSTEM GROUPS DEFINED'
* ------
GSYSNM   DC    CL28'GROUP XXXXXXXX - SYSTEM(S):'
* ------ CMDAUTH Related
ENTITY   DC    CL39'CMDX.GSYS'    Entity = OPERCMDS class profile
LOGSTR   DC    AL1(4),C'GSYS'     Command
* ------
DYN      DSECT
DYNSA    DS    18F
DYNCONN  DS    F
DYNWL    DS    XL(DEWTOE-DEWTO)
DYNWA    DS    CL128
DYNFLG   DS    XL1
DYNFSG   EQU   X'80'          HAVE SYSTEM GROUPS
         DS    0F
DYNCAP   DS    XL(CMDAULN)    CMDAUTH P-LIST
DYNE     DS    0D
* ------
         CMDAUTH MF=(L,CMDAU)
         PRINT NOGEN
         IEZVX101
         END   CMDGSYS
```

Appendix E. CONSOL00 Parmlib Member

The following CONSOL00 parmlib member is for the sample configuration shown in Figure 9 on page 41.

```
INIT      MLIM(1500)
          RLIM(999)
          LOGLIM(6000)
          AMRF(Y)
          UEXIT(N)
          MPF(00)
          PFK(00)
          CNGRP(00)
          MONITOR(DSNAME)
          ROUTTIME(5)
          CMDDELIM(<)
HARDCOPY  DEVNUM(SYSLOG,OPERLOG)
          ROUTCODE(ALL)
          CMDLEVEL(CMDS)
          UD(Y)
DEFAULT   ROUTCODE(1-16)
          RMAX(999)
          HOLDMODE(YES)
          SYNCHDEST(DCCF)
CONSOLE   DEVNUM(8B0)
          UNIT(3270-X)
          NAME(M01)
          AUTH(MASTER)
          USE(FC)
          ROUTCODE(ALL)
          ALTGRP(MASTER)
          MSCOPE(*ALL)
          CMDSYS(*)
          LEVEL(ALL)
          CON(N)
          DEL(R)
          RNUM(28)
          SEG(28)
          AREA(NONE)
          RTME(1/4)
          MFORM(T,S,J,X)
          PFKTAB(MCONPFK0)
CONSOLE   DEVNUM(8A0)
          UNIT(3270-X)
          NAME(M02)
          AUTH(MASTER)
          USE(FC)
          ROUTCODE(ALL)
          ALTGRP(MASTER)
          MSCOPE(*ALL)
          CMDSYS(*)
          LEVEL(ALL)
          CON(N)
          DEL(R)
          RNUM(28)
          SEG(28)
          AREA(NONE)
          RTME(1/4)
          MFORM(T,S,J,X)
          PFKTAB(MCONPFK0)
CONSOLE   DEVNUM(8B1)
          UNIT(3270-X)
          NAME(X05)
          AUTH(MASTER)
          USE(FC)
          ROUTCODE(ALL)
          ALTGRP(MASTER)
          MSCOPE(*ALL)
          CMDSYS(*)
          LEVEL(ALL)
          CON(N)
          DEL(R)
          RNUM(28)
          SEG(28)
          AREA(NONE)
          RTME(1/4)
          MFORM(T,S,J,X)
          PFKTAB(MCONPFK0)
CONSOLE   DEVNUM(8B2)
          UNIT(3270-X)
          NAME(X06)
          AUTH(MASTER)
          USE(FC)
          ROUTCODE(ALL)
          ALTGRP(MASTER)
          MSCOPE(*ALL)
          CMDSYS(*)
          LEVEL(ALL)
          CON(N)
          DEL(R)
          RNUM(28)
          SEG(28)
          AREA(NONE)
          RTME(1/4)
          MFORM(T,S,J,X)
          PFKTAB(MCONPFK0)
CONSOLE   DEVNUM(8B3)
          UNIT(3270-X)
          NAME(X07)
          AUTH(MASTER)
          USE(FC)
          ROUTCODE(ALL)
          ALTGRP(MASTER)
          MSCOPE(*ALL)
          CMDSYS(*)
          LEVEL(ALL)
          CON(N)
          DEL(R)
          RNUM(28)
          SEG(28)
          AREA(NONE)
          RTME(1/4)
          MFORM(T,S,J,X)
          PFKTAB(MCONPFK0)
CONSOLE   DEVNUM(8B4)
          UNIT(3270-X)
          NAME(X08)
          AUTH(MASTER)
          USE(FC)
          ROUTCODE(ALL)
          ALTGRP(MASTER)
          MSCOPE(*ALL)
          CMDSYS(*)
          LEVEL(ALL)
          CON(N)
          DEL(R)
          RNUM(28)
          SEG(28)
          AREA(NONE)
          RTME(1/4)
          MFORM(T,S,J,X)
          PFKTAB(MCONPFK0)
CONSOLE   DEVNUM(860)
          UNIT(3270-X)
          NAME(M04)
```

```
          AUTH(MASTER)
          USE(FC)
          ROUTCODE(ALL)
          ALTGRP(MASTER)
          MSCOPE(*ALL)
          CMDSYS(*)
          LEVEL(ALL)
          CON(N)
          DEL(R)
          RNUM(28)
          SEG(28)
          AREA(NONE)
          RTME(1/4)
          MFORM(T,S,J,X)
          PFKTAB(MCONPFK0)
CONSOLE   DEVNUM(861)
          UNIT(3270-X)
          NAME(X09)
          AUTH(MASTER)
          USE(FC)
          ROUTCODE(ALL)
          ALTGRP(MASTER)
          MSCOPE(*ALL)
          CMDSYS(*)
          LEVEL(ALL)
          CON(N)
          DEL(R)
          RNUM(28)
          SEG(28)
          AREA(NONE)
          RTME(1/4)
          MFORM(T,S,J,X)
          PFKTAB(MCONPFK0)
CONSOLE   DEVNUM(9E0)
          UNIT(3270-X)
          NAME(M03)
          AUTH(MASTER)
          USE(FC)
          ROUTCODE(ALL)
          ALTGRP(MASTER)
          MSCOPE(*ALL)
          CMDSYS(*)
          LEVEL(ALL)
          CON(N)
          DEL(R)
          RNUM(28)
          SEG(28)
          AREA(NONE)
          RTME(1/4)
          MFORM(T,S,J,X)
          PFKTAB(MCONPFK0)
CONSOLE   DEVNUM(9E1)
          UNIT(3270-X)
          NAME(X10)
          AUTH(MASTER)
          USE(FC)
          ROUTCODE(ALL)
          ALTGRP(MASTER)
          MSCOPE(*ALL)
          CMDSYS(*)
          LEVEL(ALL)
          CON(N)
          DEL(R)
          RNUM(28)
          SEG(28)
          AREA(NONE)
          RTME(1/4)
          MFORM(T,S,J,X)
          PFKTAB(MCONPFK0)
CONSOLE   DEVNUM(SYSCONS)
          AUTH(MASTER)
          ROUTCODE(ALL)
          LEVEL(ALL)
          UD(Y)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S01)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S02)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S03)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S04)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S05)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S06)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S07)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S08)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S09)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S10)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S11)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S12)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S13)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S14)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S15)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S16)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S17)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S18)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S19)
CONSOLE   DEVNUM(SUBSYSTEM)
          AUTH(ALL)
          NAME(S20)
```

Appendix F. MPF Exit - Add Routing Codes

The following sample MPF message exit allows you customize message routing codes and thus, helps you implement functional message routing. When you assign the customized routing codes to the set of consoles in a functional area, for example a tape pool, the operators working on that area do not see other messages routed to the consoles but those directly related to their work.

Internet access to code

Internet access to source code found in GG24-4626 can be accessed via the Internet.

http://www.redbooks.ibm.com/redbooks

Download sample code from our redbooks.

or you can:

You can access the server by anonymous FTP:

ftp ftp.almaden.ibm.com
user: anonymous
password: your E-mail address
cd redbooks/SG244626
get mpf.code

```
        TITLE 'MPF EXIT - IEF233A M devnum - ADD ROUTCODES'
MPFARCD  CSECT
MPFARCD  AMODE 31
MPFARCD  RMODE ANY
*
*  Function:
*    Add routing codes 13 and 97 to message IEF233A
*  Disclaimer:
*    This sample source is provided for tutorial purposes only. A
*    complete handling of error conditions has not been shown or
*    attempted, and this source has not been submitted to formal IBM
*    testing. This source is distributed on an 'as is' basis
*    without any warranties either expressed or implied.
*  Invocation:
*    Through MPFLSTxx definition:
*      IEF233A,USEREXIT(MPFARCD),SUP(NO),RETAIN(YES)
*  Function:
*   - The device number in the IEF233A message is checked against
*     the DEVRO table contained in this program.  When a match is
*     found, the routing codes from the DEVRO table are added to
*     the IEF233A message.
*
*  LE ATTR: RENT REUS REFR
*
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*
         SAVE (14,12),,MPFARCD.&SYSDATE..&SYSTIME.JPV
         USING MPFARCD,R12
         LR    R12,R15
         L     R5,0(,R1)                    EXIT PARMS (CTXT) PTR.
         USING CTXT,R5
*===>    GET STORAGE FOR SAVE AREA
         GETMAIN RU,LV=DASIZE,SP=230
         LR    R11,R1                       PGM WORK AREA PTR.
         USING DAREA,R11
         ST    R13,DASAVE+4
         LA    R15,DASAVE
         ST    R15,8(,R13)
         LR    R13,R15
*===>    COUNTER-CHECK MSG ID.
         L     R2,CTXTTXPJ                  MAJOR LINE MSG ATTR PTR
         USING CTXTATTR,R2
         LA    R4,CTXTTMSG                  MSG TXT PTR
         USING MSG,R4
         CLC   MSGT,=CL8'IEF233A'
         BNE   RETU
*------- FIND DEVNUM IN IEF233A M OB31,NOSER ,,VAINID,E1,X.Y.Z
         LA    R1,DEVRO                     POINT AT DEVICE TABLE
DVL      DS    0H
         CLI   0(R1),X'FF'                  END OF LIST?
         BE    RETU
         CLC   MSGDV,0(R1)                  DEVICE IN TABLE?
         BE    DVP                          YES - ADD NEW ROUTING CODES
         LA    R1,L'DEVRO(,R1)              POINT AT NEXT TABLE ENTRY
         B     DVL                          LOOP ....
DVP      DS    0H
         L     R3,CTXTRCP                   POINT AT ROUTING CODES
         USING CTXTROUT,R3
         LH    R6,CTXTRCLN                  L'ROUTING CODES
         BCTR  R6,0
         OC    CTXTROUT(*-*),4(R1)          ADD NEW ROUTING CODES
```

```
         EX    R6,*-6                      COMPLETE ADD
         DROP  R3
         OI    CTXTRFLG,CTXTRCRC           CHANGE ROUTING CODE
* ------
RETU     DS    0H
         L     R13,4(,R13)
         FREEMAIN RU,LV=DASIZE,SP=230,A=(R11)
         ST    R15,8(,R13)
         RETURN (14,12)
* ------
         LTORG
* ------ DEVNUM - ROUTING CODE TABLE
DEVRO    DS    0CL(4+16)
*              DEVNO   ROUTCODE  V - 13              V - 97
         DC    CL4'0B30',XL16'00080000000000000000000080000000'
         DC    CL4'0B31',XL16'00080000000000000000000080000000'
         DC    CL4'0B32',XL16'00080000000000000000000080000000'
         DC    CL4'0B33',XL16'00080000000000000000000080000000'
         DC    CL4'0B34',XL16'00080000000000000000000080000000'
         DC    CL4'0B35',XL16'00080000000000000000000080000000'
         DC    CL4'0B36',XL16'00080000000000000000000080000000'
         DC    CL4'0B37',XL16'00080000000000000000000080000000'
         DC    X'FF'                        END OF LIST
DAREA    DSECT
DASAVE   DS    18F
DAEND    DS    0D
DASIZE   EQU   DAEND-DAREA
*
MSG      DSECT
MSGT     DS    CL8                          MSG ID
MSGA     DS    CL1                          ACTION
         DS    CL1
MSGDV    DS    CL4                          DEVICE NUMBER
*
         IEZVX100
*
         END   MPFARCD
```

Appendix G. REXX EXEC for Extended MCS Console

This appendix contains a sample program for a REXX EXEC for an MVS operator.

Internet access to code

Internet access to source code found in GG24-4626 can be accessed via the Internet.

http://www.redbooks.ibm.com/redbooks

Download sample code from our redbooks.

or you can:

You can access the server by anonymous FTP:

```
ftp ftp.almaden.ibm.com
user: anonymous
password: your E-mail address
cd redbooks/SG244626
get emcs.code
```

The CLIST in Figure 57 sets up the necessary ISPF libraries and invokes the actual REXX EXEC for an MVS operator extended MCS console. It assumes that required panels, REXX EXECs, and the ISPF command table definitions are installed in the 'userid.C.CLIST' data set.

```
PROC 0 DB HLQ(userid) NAME()
IF .&DB = .DB THEN +
  CONTROL LIST SYMLIST CONLIST MSG PROMPT
ISPEXEC CONTROL ERRORS RETURN
ISPEXEC LIBDEF ISPPLIB DATASET ID('&HLQ..EMCS.CLIST')
ISPEXEC LIBDEF ISPTLIB DATASET ID('&HLQ..EMCS.CLIST')
ALTLIB  ACT    APPL(CLIST)     DA('&HLQ..EMCS.CLIST')
IF &NAME ¬= &STR() THEN SET A = NAME(&NAME.)
ELSE SET A = &STR()
ISPEXEC SELECT CMD(CNDS &A &DB) NOCHECK NEWAPPL(CNDS) PASSLIB NEWPOOL
ALTLIB  DEACT  APPL(CLIST)
ISPEXEC LIBDEF ISPPLIB
ISPEXEC LIBDEF ISPTLIB
```

Figure 57. REXX EXEC for MVS Operator

Figure 58 is the ISPF 3.9 selection table display for the CNDSCMDS command used by the REXX EXEC. Before invoking the application, define the command verbs.

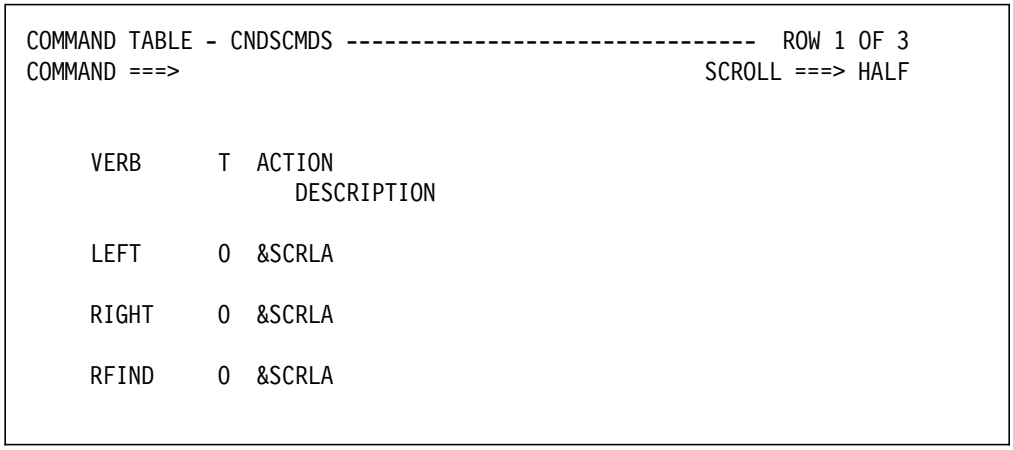


Figure 58. ISPF 3.9 Selection Table Display

The following sample code is the REXX EXEC for the MVS operator extended MCS console.

```
/* REXX             ##### UNCLASSIFIED #####                     */
/* ------------------------------------------------------------------*/
/* Sample REXX EXEC that demonstrates the power of CONSOLE and      */
/* CONSPROF commands when combined with the ISPF Dialog Management  */
/* Services. You get a full function console that does not wrap or  */
/* roll and retains all messages as long you keep this EXEC active. */
/* Furthermore a PDF command is provided which invokes ISPF PDF in  */
/* case you need to keep the console active and do some other work  */
/* on this (split) screen.                                          */
/*                                                                  */
/* This EXEC provides two modes of CONSOLE message displays:        */
/*                                                                  */
/* - SOL (for solicited). Only the output of the CONSOLE command    */
/*   that was issued on the SOL mode is displayed. The MCS or sub-  */
/*   system commands are issued with a CART and the REXX GETMSG     */
/*   function with the same CART is used to retrieve the output.    */
/*   Meanwhile unsolicited messages are captured, but are not       */
/*   displayed unless you enter BOTH mode.                          */
/*                                                                  */
/*   The console command output is stored into an ISPF table which  */
/*   is displayed with TBDISPL DM service. You can scroll the       */
/*   message output table UP and DOWN.                              */
/*   If you enter a command that does not generate output, for      */
/*   example S DEALLOC, you will be locked-out for 15 seconds.      */
/*   You should not use this mode to issue JES3 commands, prior     */
/*   to JES3 5.2.1.  They do not support CART and you will be       */
/*   locked out.                                                    */
/*                                                                  */
/* - BOTH (for both solicited and unsolicited).  All message traffic*/
/*   directed to your console is displayed.  The MCS or subsystem   */
/*   system commands are issued without a CART and the REXX GETMSG  */
/*   function without any CART is used to retrieve the output.      */
/*                                                                  */
/*   The console command output is stored into an ISPF table which  */
/*   is displayed with TBDISPL DM service. You can scroll the       */
/*   message output table UP, DOWN, RIGHT, and LEFT.                */
/*   A FIND and RFIND command helps you to find a specific string of*/
/*   data in the output message table.                              */
/*   On the display panel you can select time stamp, system name,   */
/*   and job name to be included as part of the display data. You   */
/*   can also lock the display. When locked, the display does not   */
/*   move unless you use the scroll commands (PFKs). Normally when  */
/*   the display is unlocked, the display moves to the first new    */
/*   line of data that has not yet been displayed.                  */
/*                                                                  */
/*   The BOTH mode should be used to issue JES3 commands, prior      */
/*   to JES3 5.2.1.                                                 */
/* NOTE:                                                            */
/*   There is no programmed upper limit for the size of the BOTH    */
/*   mode output message table (no TBDELETEs).                      */
/*                                                                  */
/*   You can have this EXEC running on both sides of split screens. */
/*   To have them both in BOTH mode is not recommended.             */
/*                                                                  */
/* ------------------------------------------------------------------ */
parse upper arg a
if wordpos("DB",a) <> 0 then trace "R"
else trace "o"
parse var a with "NAME(" b ")" .
if b = "" then cname = ""
else cname = "NAME("||b||")"
/* ------------------------------------------------------------------ */
/*   This EXEC must run under ISPF. Let's check it. If not, the     */
/*   EXEC issues a message and exits.                               */
/* ------------------------------------------------------------------ */
if sysvar(sysispf) ¬= 'ACTIVE' then do
  say "Sorry - this REXX EXEC works only under ISPF!"
  exit
end
else nop
/* - Get CVT SYSNAME ------------------------------------------------ */
cvt = c2x( storage(10,4) )
cvt_154 = d2x( x2d(cvt) + x2d(154) )
SID = storage(cvt_154,8)
/* ------------------------------------------------------------------ */
/*   Set console display mode and activate console.                 */
/*   If you are running in split screens, you will see the message: */
/*   IKJ55302I THE CONSOLE SESSION IS ALREADY ACTIVE.               */
/*   It's just a reminds you that there is already one.             */
/* ------------------------------------------------------------------ */
x = MSG("OFF")
address "TSO"
"CONSPROF SOLDISP(NO) UNSOLDISP(NO) SOLNUM(4500) UNSOLNUM(4500)"
"CONSOLE ACTIVATE" cname
If rc > 4 then do
  zedsmsg = ""
  zedlmsg = "CONSOLE activation failed - RC="rc
  address "ISPEXEC" "SETMSG MSG(ISRZ001)"
  exit 16
end
x = MSG(x)
/* ------------------------------------------------------------------ */
/*   The CONSOLE command must be kept active as long as there is    */
/*   one copy of this EXEC active. A count of active copies is      */
/*   maintained.                                                    */
/* ------------------------------------------------------------------ */
address "ISPEXEC" "CONTROL ERRORS RETURN"
address "ISPEXEC" "CONTROL DISPLAY SAVE"
address "ISPEXEC" "VGET (ZSPLIT ZSCREEN)"
"ALLOC DD(@#$@#$"||ZSCREEN") NEW SPA(0) TRA"
if RC ¬= 0 then do
  call SETMP "Sorry - there is already one on this side of (split) screen!"
  exit
end
address "ISPEXEC" "VGET (CNDSCT) PROFILE"
if ZSPLIT = "NO" then CNDSCT = 1
else do
  if datatype(CNDSCT,"N") then CNDSCT = CNDSCT + 1
  else  CNDSCT = 1
end
address "ISPEXEC" "VPUT (CNDSCT) PROFILE"
address "ISPEXEC" "VGET (CDTS CDSN CDJN CDHD) PROFILE"
/* ------------------------------------------------------------------ */
/*   Initialize some panel field variables and parameters.          */
/* ------------------------------------------------------------------ */
CER = "Enter '?' for HELP"
```

```
SA = ""
UND = "ORG"            /* Unsolicited message scroll LEFT/RIGHT valu*/
UST = "NOT"            /* Unsolicited message table not created     */
A   = "HALF"           /* ISPF table display scroll amount          */
ST = TIME('N')
SY = " "
                       /* Set time stamp to be displayed            */
if  (CDTS = "Y") | (CDTS = "N") then nop
else CDTS = "Y"
                       /* Set system name to be displayed           */
if  (CDSN = "Y") | (CDSN = "N") then nop
else CDSN = "Y"
                       /* Set job name to be displayed              */
if  (CDJN = "Y") | (CDJN = "N") then nop
else CDJN = "Y"
                       /* Set hold mode to NO                       */
if  (CDHD = "Y") | (CDHD = "N") then nop
else CDHD = "N"
ZCMD = "?"             /* Set initial command to HELP               */
TN = "MSCCMD"ZSCREEN   /* Solicited message table name              */
UN = "USLCMD"ZSCREEN   /* Unsolicited message table name            */
/* ---------------------------------------------------------------- */
/*   Create initial solicited message table and display it.         */
/* ---------------------------------------------------------------- */
address "ISPEXEC" "TBCREATE" TN "NAMES(CTXT) NOWRITE REPLACE"
/* ---------------------------------------------------------------- */
/*   Main line.                                                     */
/* ---------------------------------------------------------------- */
/* ---------------------------------------------------------------- */
/*   If this is the first of the kind, start with BOTH mode         */
/*   otherwise with the SOL mode.                                   */
/* ---------------------------------------------------------------- */
if CNDSCT > 1 then do
  MOD  = "SOL"
  IMOD = "SOL"
  call SOL
end
else do
  MOD  = "BOTH"
  IMOD = "BOTH"
  call BOTH
end
/* ---------------------------------------------------------------- */
/*   Clean-up and exit .....                                        */
/* ---------------------------------------------------------------- */
address "ISPEXEC" "TBCLOSE" TN
address "ISPEXEC" "TBCLOSE" UN
address "ISPEXEC" "VGET (CNDSCT) PROFILE"
CNDSCT = CNDSCT - 1
address "ISPEXEC" "VPUT (CNDSCT) PROFILE"
address "ISPEXEC" "VPUT (CDTS CDSN CDJN CDHD) PROFILE"
address "TSO"
if CNDSCT = 0 then do
  "CONSPROF SOLDISP(YES) UNSOLDISP(YES) SOLNUM(4500) UNSOLNUM(4500)"
  "CONSOLE DEACTIVATE"
end
else nop
address "ISPEXEC" "CONTROL DISPLAY RESTORE"
"UNALLOC DD(@#$@#$"||ZSCREEN")"
EXIT
/* ---------------------------------------------------------------- */
/*   Subroutines ......                                             */
/* ---------------------------------------------------------------- */
/* ---------------------------------------------------------------- */
/*   SOL mode processing.                                           */
/* ---------------------------------------------------------------- */
SOL:
    SY = " "
    SCRLA = " "        /* ISPF command variable value               */
    do forever
      /* ------------------------------------------------------- */
      /*   Select action based on the contents of the ZCMD.      */
      /*    - PDF   select ISR@PRIM panel                        */
      /*    - BOTH  invoke solicited and unsolicited msg processing */
      /*    - ?     display HELP information                     */
      /*    - other issue MCS console command with CART          */
      /*    - empty NOP                                          */
      /* ------------------------------------------------------- */
      select
        when (ZCMD = "PDF") then ,
          address "ISPEXEC" "SELECT PANEL(ISR@PRIM) NEWAPPL(ISP)"
        when ZCMD = "BOTH" then do
          if IMOD = "BOTH" then do
            FR = 8
            leave
          end
          else do
            call BOTH
            MOD = "SOL"
            SCRLA = " "        /* ISPF command variable value       */
          end
        end
        when ZCMD ¬= " " then do
          MOD = "SOL"
          address "ISPEXEC" "TBCREATE" TN "NAMES(CTXT) NOWRITE REPLACE"
          select
            when ZCMD = "?" then call INFO
            otherwise do
                /* ------------------------------------------------ */
                /*   Issue MCS command with CART                    */
                /* ------------------------------------------------ */
                "CONSOLE SYSCMD("ZCMD") CART('MY100000')"
                /* ------------------------------------------------ */
                /*   Retrieve response messages for the CART        */
                /* ------------------------------------------------ */
                GC = 0
                MGC = 999
                TME = 10
                do while GC = 0
                  GC = GETMSG('R.','SOL','MY100000',,TME)
                  IF GC = 0 THEN DO
                   do i = 1 to r.0
                     MGC = min(GC,MGC)
                     TME = 2
                     CTXT =  r.i
                     SY =    r.MDBGOSNM
                     ADDRESS "ISPEXEC" "TBADD" TN
                   end
                  END
                  ELSE MGC = min(MGC,GC)
                end
                select
                  when MGC = 0 then nop
                  when MGC = 4 then it = '- did not retrieve messages'
                  when MGC = 8 then it = '- ATTENTION during processing'
                  otherwise it = '- see SC28-1883 for more info'
                end
                if MGC ¬= 0 then do
                  CTXT =  '          ===>> GETMSG rc =' MGC it
                  ADDRESS "ISPEXEC" "TBADD" TN
                end
              end
            end
          end
          otherwise nop
        end
        /* ------------------------------------------------------- */
        /*   Display solicited message table.                      */
        /* ------------------------------------------------------- */
        address "ISPEXEC" "TBTOP" TN
        address "ISPEXEC" "TBDISPL" TN "PANEL(USRCN)"
        FR = RC
        if FR > 4 then leave
        else nop
      end
return FR
/* ---------------------------------------------------------------- */
/*   BOTH mode processing.                                          */
/* ---------------------------------------------------------------- */
BOTH:
    SCRLA = "PASSTHRU" /* ISPF command variable value               */
    MOD = "BOTH"
    if UST = NOT then do
      address ISPEXEC "TBCREATE" UN "NAMES(TS SY JN CT CP XT) NOWRITE REPLACE"
      UST = YES
    end
    else nop
    UNP = 1
    do forever
      /* ------------------------------------------------------- */
      /*   Display solicited and unsolicited message table.      */
      /* ------------------------------------------------------- */
      address "ISPEXEC" "TBTOP" UN
      address "ISPEXEC" "TBSKIP" UN "NUMBER("||UNP||")"
      CVAR = ""
      /* ------------------------------------------------------- */
      /*   Test whether to display time stamp, system name, job name*/
      /*   and implement LEFT/RIGHT scrolling.                   */
      /* ------------------------------------------------------- */
      if CDTS = "Y" then CVAR = CVAR || "¢TS      "
      if CDSN = "Y" then CVAR = CVAR || "¢SY      "
      if CDJN = "Y" then CVAR = CVAR || "¢JN      "
      select
        when UND = "ORG" then CVAR = CVAR||"+CT"
        when UND = "LEFT" then CVAR = "+CT"
        when UND = "MID"  then CVAR = "+CP"
        otherwise  CVAR = "+XT"
      end
      address "ISPEXEC" "TBDISPL" UN "PANEL(USRCNS)"
      FR = RC
      CER = "Enter '?' for HELP"
      tasw = 0
      if FR > 4 then leave
      else nop
      /* ------------------------------------------------------- */
      /*   Select action based on the contents of the ZCMD.      */
      /*    - ?     display HELP information                     */
      /*    - SOL   invoke solicited message processing          */
      /*    - LETT/RIGHT scroll processing                       */
      /*    - PDF   select ISR@PRIM panel                        */
      /*    - PRT   copy message table into 'userid.CONSOLE.LOG' */
      /*    - BOTH  invoke solicited and unsolicited processing  */
      /*    - FIND  find a string in the table. Find starts always */
      /*            at the top of the table and proceeds to the  */
      /*            bottom.                                      */
      /*    - RFIND repeat a previous find starting with the line */
```

```
/*            on the top of the display. If the bottom is     */
/*            reached before the string is found, the search  */
/*            continues from the top of the table.            */
/*    - other issue MCS console command and retrieve messages */
/*    - empty try to retrieve new messages                    */
/* ------------------------------------------------------------ */
select
  when ZCMD = "?" then do
      SY = " "
      address "ISPEXEC" "TBCREATE" TN "NAMES(CTXT) NOWRITE REPLACE"
      call INFO
      MOD  = "INFO"
      address "ISPEXEC" "TBBOT" TN
      address "ISPEXEC" "TBDISPL" TN "PANEL(USRCN)"
      MOD  = "BOTH"
    end
  when ZCMD = "SOL" then do
    if IMOD = "SOL" then do
      FR = 8
      leave
    end
    else do
      ZCMD = "?"
      call SOL
      MOD = "BOTH"
      SCRLA = "PASSTHRU" /* ISPF command variable value       */
    end
  end
  when (ZCMD = "LEFT") | (ZCMD = "RIGHT") then do
    if (CDTS||CDSN||CDJN = "NNN") & (UND = "ORG") then UND= "LEFT"
    select
      when (UND = "ORG") & (ZCMD="RIGHT")  then UND = "LEFT"
      when (UND = "LEFT") & (ZCMD="RIGHT") then UND = "MID"
      when (UND = "MID") & (ZCMD="RIGHT")  then UND = "RIGHT"
      when (UND = "RIGHT") & (ZCMD="LEFT") then UND = "MID"
      when (UND = "MID") & (ZCMD="LEFT")   then UND = "LEFT"
      when (UND = "LEFT") & (ZCMD="LEFT")  then UND = "ORG"
      otherwise  nop
    end
    UNP = ZTDTOP
  end
  when (word(ZCMD,1) = "PDF") then do
      op = "" ; parse var ZCMD . op . ; op = "OPT("op")"
      address "ISPEXEC" "CONTROL DISPALY SAVE"
      address "ISPEXEC" "SELECT PANEL(ISR@PRIM) NEWAPPL(PDF)" op
      address "ISPEXEC" "CONTROL DISPALY RESTORE"
    end
  when (word(ZCMD,1) = "PRT") then call PRT
  when ZCMD ¬= " " then do
    ZC = subword(ZCMD,1,1)
    select
      when ZC = "FIND" then do
        UNP = ZTDTOP
        address "ISPEXEC" "TBTOP" UN
        SA = strip(overlay(" ",ZCMD,1,4),"B")
        if substr(SA,1,1) = "'" then SA = strip(SA,"B","'")
        else nop
        if substr(SA,1,1) = '"' then SA = strip(SA,"B",'"')
        else nop
        if SA = "" then  call SETMP "Srcharg missing"
        else do
          call FRF
          if ZR = 8 then call SETMP "Not found"
          else nop
        end
      end
      when ZC = "RFIND" then do
        if words(ZCMD) > 1 then call SETMP "Use FIND command"
        else do
          if strip(SA) ¬= "" then do
            UNP = ZTDTOP
            ZP = UNP + 1
            address "ISPEXEC" "TBTOP" UN
            address ISPEXEC "TBSKIP" UN "NUMBER("ZP") POSITION(ZP)"
            call FRF
            if ZR = 8 then do
              address "ISPEXEC" "TBTOP" UN
              call FRF
              if ZR = 8 then call SETMP "Not found"
              else nop
            end
            else nop
          end
          else call SETMP "Srcharg missing"
        end
      end
      otherwise do
        if (substr(ZCMD,1,1)="~") then  ZCMD=strip(ZCMD,"L","~")
        /* ------------------------------------------------- */
        /*    Issue MCS command                              */
        /* ------------------------------------------------- */
        "CONSOLE SYSCMD("ZCMD") "
        call BMGM      /* Retrieve messages                  */
      end
    end
  end
  otherwise call BMGM  /* Retrieve messages                  */
end
    end
return
/* ------------------------------------------------------------ */
/*   BOTH mode - retrieve messages.                             */
/* ------------------------------------------------------------ */
BMGM:
      HDP = ZTDTOP
      GC = 0
      ADDRESS "ISPEXEC" "TBBOTTOM" UN "NOREAD POSITION(UNP)"
      do while gc = 0
        GC = GETMSG('U.',,'EITHER',,,1)
        IF GC = 0 then do
          do i = 1 to u.0
            /* ------------------------------------------------- */
            /*   Pick up time stamp, system name and job name    */
            /*   from MDB.                                       */
            /* ------------------------------------------------- */
            JN =    u.MDBCOJBN
            SY =    u.MDBGOSNM
            TS =    u.MDBGTIMH
            /* ------------------------------------------------- */
            /*   Split message text into scroll LEFT/RIGHT variables*/
            /* ------------------------------------------------- */
            CLXT =  u.MDBTLEN.i - 50
            CLCP =  u.MDBTLEN.i - 25
            if CLCP > 79 then CLCP = 79
            CT =  substr(u.i,1,79)
            if CLCP > 0 then CP =  substr(u.i,26,CLCP)
            else CP = " "
            if CLXT > 0 then XT =  substr(u.i,51,CLXT)
            else XT = " "
            /* ------------------------------------------------- */
            /*   Add text to solicited/unsolicited message table */
            /* ------------------------------------------------- */
            ADDRESS "ISPEXEC" "TBADD" UN
            /* ------------------------------------------------- */
            /*   If in display is locked, don't move display     */
            /* ------------------------------------------------- */
            if CDHD = "N" then UND = "ORG"
            else UNP = HDP
            tasw = 1
          end
        end
        else do
          if tasw = 0 then UNP = HDP
          else do
            if ZCMD ¬= "" then UNP = max(ZTDROWS,1)
            else nop
          end
        end
      end
      if ZCMD ¬= "" then do
        JN = " "
        SY = " "
        TS = " "
        CT =  substr(ZCMD,1,79)
        CMDL =  length(strip(ZCMD))
        CLXT =  CMDL  - 50
        CLCP =  CMDL  - 25
        if CLCP > 0 then CP =  substr(ZCMD,26,CLCP)
        else CP = " "
        if CLXT > 0 then XT =  substr(ZCMD,51,CLXT)
        else XT = " "
        ADDRESS "ISPEXEC" "TBADD" UN
        if CDHD = "Y" then UNP = HDP
      end
return
/* ------------------------------------------------------------ */
/*   FIND/RFIND a string in the table.                          */
/* ------------------------------------------------------------ */
FRF:
      ZR = 0
      /* ------------------------------------------------- */
      /*   Scan table from top to bottom until found or e.o.t.     */
      /* ------------------------------------------------- */
      do forever
        address "ISPEXEC" "TBSKIP" UN "NUMBER(1) POSITION(ZP)"
        ZR = RC
        if ZR = 0 then do
          ZW = substr(ct,1,25)||SUBSTR(CP,1,25)||XT
          if index(ZW,SA,1) = 0 then nop
          else do
            UNP = ZP
            leave
          end
        end
        else leave
      end
return
/* ------------------------------------------------------------ */
/*   Copy message table into 'userid.CONSOLE.LOG' data set      */
/* ------------------------------------------------------------ */
PRT:
         Parse upper var ZCMD . d .
         ds  = "'"SYSVAR("sysuid")".CONSOLE.LOG'"
         Select
           when (d = "") then  d = "SHR"
           when (d = "SHR") | (d = "MOD") then  nop
```

```
          otherwise d = "SHR"
        end
        If SYSDSN(ds) ¬= "OK" then
          do
          "ALLOC DD($$CNLG##) DS("ds") SPA(10 20) TRA " ,
          "RECF(V B) LREC(255) BLKSI(25504) DSOR(PS) NEW REUSE"
          "UNALLOC DD($$CNLG##)"
          end
        else nop
        "ALLOC DD($$CNLG##) DS("ds") REUSE" d
        Address "ISPEXEC" "TBTOP" UN
        pc = 0
        "NEWSTACK"
        do while pc = 0
          Address "ISPEXEC" "TBSKIP" UN
          pc = RC
          if pc = 0 then do
            Address "ISPEXEC" "TBGET" UN
            /*  (TS SY JN CT CP XT) */
            mts = "        "; mts = overlay(ts,mts,1)
            msy = "        "; msy = overlay(sy,msy,1)
            mjn = "        "; mjn = overlay(jn,mjn,1)
            msg = ct; msg = overlay(cp,msg,26);msg = overlay(xt,msg,51)
            queue mts msy mjn msg
          end
        end
        queue ""
        "EXECIO * DISKW $$CNLG## (FINIS "
        "UNALLOC DD($$CNLG##)"
        "DELSTACK"
        call SETMS "Output in data set" ds
return
/* ---------------------------------------------------------------- */
/*   Build initial and help solicited table.                        */
/* ---------------------------------------------------------------- */
INFO:
address "ISPEXEC"
CTXT =  "    - Please, enter MCS command on the Command ===> line"
"TBADD" TN
CTXT =  "      or very long commands on the Long Command ===> line."
"TBADD" TN
CTXT =  "      The long commands can not be retrieved. Sorry!"
"TBADD" TN
CTXT =  "    - The initial 'SOL' mode displays only solicited messages. "
"TBADD" TN
CTXT =  "    - To view unsolicited messages - enter command 'BOTH'."
"TBADD" TN
CTXT =  "      In 'BOTH' mode you can scroll the display 'LEFT' and 'RIGHT'."
"TBADD" TN
CTXT =  "      The screen is updated every time you enter a command or when"
"TBADD" TN
CTXT =  "      you just hit the 'ENTER' key."
"TBADD" TN
CTXT =  "      You can enter a 'FIND' command to locate the first row where the"
"TBADD" TN
CTXT =  "      search argument string is found. 'RFIND' locates the next occu-"
"TBADD" TN
CTXT =  "      rance of the search argument."
"TBADD" TN
CTXT =  "    - To return to solicited mode enter command 'SOL' or 'END'."
"TBADD" TN
CTXT =  "      The unsolicited messages are retained and are displayed when "
"TBADD" TN
CTXT =  "      you re-enter the "BOTH" mode."
"TBADD" TN
CTXT =  "    - If you want to work with PDF on this screen - type 'PDF'."
"TBADD" TN
CTXT =  "      Console is left active and collects messages which are"
"TBADD" TN
CTXT =  "      displayed on "BOTH" mode when you exit from PDF."
"TBADD" TN
CTXT =  "    - You can copy the message table to 'useid.CONSOLE.LOG' data set"
"TBADD" TN
CTXT =  "      with 'PRT' command. If you want to append the message table to"
"TBADD" TN
CTXT =  "      the data set use 'PRT MOD' command."
"TBADD" TN
CTXT =  "    - To EXIT enter 'END' or hit PF3 - in 'BOTH' mode hit 'END' twice."
"TBADD" TN
CTXT =  " "
"TBADD" TN
CTXT =  "NOTE: After entering a command in solicited mode, you can be 'locked-"
"TBADD" TN
CTXT =  "      out' if there are no messages. To 'unlock' - hit 'ATTN' key."
"TBADD" TN
CTXT =  "      The 'lock out' time is 15 seconds."
"TBADD" TN
CTXT =  "    - To re-display this info enter '?'. (HELP is reserved for ISPF!)"
"TBADD" TN
"TBTOP" TN
address "TSO"
return
/* ---------------------------------------------------------------- */
/*   Set ISPF messages                                              */
/* ---------------------------------------------------------------- */
SETMP: /* with PEEP */
  parse arg ZEDLMSG
  ZEDSMSG = ""
  address "ISPEXEC" "SETMSG MSG(ISRZ001)"
return
SETMS: /* no PEEP   */
  parse arg ZEDLMSG
  ZEDSMSG = ""
  address "ISPEXEC" "SETMSG MSG(ISRZ000)"
return
/* --------------------------------- End of EXEC -------------------- */
```

Figure 59 shows the USRCN panel that is used to display solicited messages.

```
)ATTR
 _ TYPE(INPUT) INTENS(HIGH)  PADC(NULLS) JUST(LEFT) CAPS(ON)
 @ TYPE(INPUT) INTENS(LOW)   PAD(' ')  JUST(ASIS) CAPS(ON)
 % TYPE(OUTPUT) INTENS(HIGH) PAD(' ')  JUST(ASIS) CAPS(ON)
 ? TYPE(OUTPUT) INTENS(HIGH) PAD('-')  JUST(ASIS) CAPS(ON)
 ¢ TYPE(OUTPUT) INTENS(HIGH) PAD(' ')  JUST(RIGHT) CAPS(ON)
 + TYPE(OUTPUT) INTENS(LOW)  PAD(' ')  JUST(ASIS) CAPS(ON)
 | TYPE(TEXT) INTENS(HIGH)   PAD(' ')  JUST(ASIS) CAPS(ON)
 ! TYPE(TEXT) INTENS(LOW)    PAD(' ')  JUST(LEFT) CAPS(ON)
)BODY
|------------------------- LIST MCS COMMAND OUTPUT -----------------------------
|C =>_ZCMD                                                                    |S =>_A   |
|Mode:+MOD  |%SID     |
|Long C =>_OCMD
                                                        !&ZTIME &ZDATE            |
?ZX
)MODEL
+CTXT
)INIT
  &ZX = '----'
  &OCMD = ' '
)REINIT
)PROC
  IF (&ZCMD = ' ')
    IF (&OCMD ¬= ' ')
      &SCMD = &OCMD
      &ZCMD = &OCMD
  IF (&ZCMD = 'RETL')
    IF (&SCMD ¬= ' ')
      &ZCMD = ' '
      &OCMD = &SCMD
      REFRESH (OCMD ZCMD)
      &ZEDSMSG = 'O.K.'
      &ZEDLMSG = 'Enter / Modify retrieved command'
      .MSG = ISRZ000
    ELSE
      &ZCMD = ' '
      REFRESH ZCMD
      &ZEDSMSG = 'Nothing Saved'
      &ZEDLMSG = 'Nothing to retreve - Long commands have not been used'
      .MSG = ISRZ001
)END
```

Figure 59. USRCN Panel to Display Unsolicited Messages

Figure 60 on page 201 shows the USRCNS panel that is used to display both solicited and unsolicited messages.

```
)ATTR
 _ TYPE(INPUT) INTENS(HIGH)  PADC(NULLS) JUST(LEFT) CAPS(ON)
 % TYPE(OUTPUT) INTENS(HIGH) PAD(' ')  JUST(ASIS) CAPS(ON)
 ? TYPE(OUTPUT) INTENS(HIGH) PAD('-')  JUST(ASIS) CAPS(ON)
 ¢ TYPE(OUTPUT) INTENS(HIGH) PAD(' ')  JUST(LEFT) CAPS(ON)
 + TYPE(OUTPUT) INTENS(LOW)  PAD(' ')  JUST(ASIS) CAPS(ON)
 \ TYPE(OUTPUT) INTENS(LOW)  PAD(' ')  JUST(LEFT) CAPS(ON)
 | TYPE(TEXT) INTENS(HIGH)   PAD(' ')  JUST(ASIS) CAPS(ON)
 ! TYPE(TEXT) INTENS(LOW)    PAD(' ')  JUST(ASIS) CAPS(ON)
)BODY
|------------------------- LIST MCS COMMAND OUTPUT -----------------------------
|C =>_ZCMD                                                                |S =>_A   |
|Mode:\MOD  |%SID     |
|Long C =>_OCMD
                                                 !&ZTIME &ZDATE |&CER
|Display ===>  !Time Stamp _Z! System Name _Z! Job Name  _Z| Hold ===> _Z| (Y/N)
?ZX
)MODEL
&CVAR
)INIT
  .ZVARS = '(CDTS CDSN CDJN CDHD)'
  &ZX = '----'
  &OCMD = ' '
)REINIT
)PROC
  VER (&CDTS,LIST,Y,N)
  VER (&CDSN,LIST,Y,N)
  VER (&CDJN,LIST,Y,N)
  VER (&CDHD,LIST,Y,N)
  IF (&ZCMD = ' ')
    IF (&OCMD ¬= ' ')
      &SCMD = &OCMD
      &ZCMD = &OCMD
  IF (&ZCMD = 'RETL')
    IF (&SCMD ¬= ' ')
      &ZCMD = ' '
      &OCMD = &SCMD
      REFRESH (OCMD ZCMD)
      &ZEDSMSG = 'O.K.'
      &ZEDLMSG = 'Enter / Modify retrieved command'
      .MSG = ISRZ000
    ELSE
      &ZCMD = ' '
      REFRESH ZCMD
      &ZEDSMSG = 'Nothing Saved'
      &ZEDLMSG = 'Nothing to retreve - Long commands have not been used'
      .MSG = ISRZ001
)END
```

Figure 60. USRCNS Panel to Display both Solicited and Unsolicited Messages

Appendix H. Extended MCS Console Sample

The attached sample program activates an extended MCS console to receive the hardcopy message set (MDBs). The MDBs are “opened” and the message text is written with the TSO PUTLINE service. For the functional details of the program see the prologue text in beginning of the program source listing.

The MSCOPE of the extended EMCS console is defined as “*ALL,” thus providing a sysplex wide view of the hardcopy message set. The recommended way to run the program is under the TSO background session:

```
//SPXLOG JOB (999,POK),EXPERT,MSGLEVEL=1,MSGCLASS=A,
//       CLASS=A,NOTIFY=&SYSUID
//*MAIN  SYSTEM=SC50
//EMCS   EXEC PGM=IKJEFT01,DYNAMNBR=99
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  HCEMCS
```

You can view the log output through SDSF of a equivalent product.

Internet access to code

Internet access to source code found in GG24-4626 can be accessed via the Internet.

http://www.redbooks.ibm.com/redbooks

Download sample code from our redbooks.

or you can:

You can access the server by anonymous FTP:

ftp ftp.almaden.ibm.com
user: anonymous
password: your E-mail address
cd redbooks/SG244626
get emcsmdb.code

```
        TITLE 'ANOTHER SYSPLEX WIDE HARCOPY LOG?'
HCEMCS   CSECT
HCEMCS   AMODE 31
HCEMCS   RMODE ANY
* ------
* FUNCTION:
*   Receive the hardcopy message set through an EMCS console and
*   use TSO/E PUTLINE service to write it to the terminal.
* DISCLAIMER:
*   This sample source is provided for tutorial purposes only.  A
*   complete handling of error conditions has not been shown or
*   attempted, and this source has not been submitted to formal
*   IBM testing.  This source is distributed on an 'as is' basis
*   without any warranties either expressed or implied.
* INVOCATION:
*   As TSO command processor 'HCEMCS'.  No parameters required nor
*   processed.  Run this command in the background.  If you run it
*   in the foreground, it locks your terminal.  You can 'attention'
*   yourself out.
* FUNCTION:
*  - Open operator communication for MOFIFY/STOP command using
*    QEDIT macro service.
*  - Activate an attention exit
*  - Activate an extended MCS console to receive the hardcopy
*    message set (MDBs).
*  - Enter WAIT for either CIB processing, TSO attention, or MDB
*    queued for the console:
*     - For CIB post - check the command verb:
*        - STOP (P) command - close down the program
*        - MODIFY (F) command - if the modify text is either
*          STOP or END, close down the program.  For all other
*          modify texts, issue a response back to the console
*          that entered the command.  (The response is word
*          WHAT?)  Note the command response WTO does not use
*          the CART located in the CIB extension.  Finally,
*          the CIB is deleted such that new MODIFY/STOP
```

```
*          commands can be received and WAIT is entered.
*     - Attention interrupt:
*       - Exit program immediately
*     - For MDB queued post:
*       - Retrieve the queued MDB and move it into a work area
*       - Scan the MDB objects and issue PUTLINE for each text
*         object message text.  The message text is preceded
*         by a header consisting of system name, date, and time
*         stamp. The header information is extracted from the
*         MDB general object.
*
* LE ATTR: AC(1)
* NOTE:
*  To run this TSO/E command you must update IKJTSOxx member
*  to include HCEMCS name in the authorized command section.
* ------
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
* ------ ACCESS REGISTERS
AR0      EQU   0
AR1      EQU   1
AR2      EQU   2
AR3      EQU   3
AR4      EQU   4
AR5      EQU   5
AR6      EQU   6
AR7      EQU   7
AR8      EQU   8
AR9      EQU   9
AR10     EQU   10
AR11     EQU   11
AR12     EQU   12
AR13     EQU   13
AR14     EQU   14
AR15     EQU   15
* -----
         SAVE  (14,12),,&SYSDATE.-&SYSTIME./HCEMCS
         USING HCEMCS,R12
         LR    R12,R15
         ST    R13,SA+4
         LA    R3,SA
         ST    R3,8(,R13)
         LR    R13,R3
* ------ SAVE CPPL INFO
         TM    0(R1),X'80'        IS THIS A CPPL?
         BZ    GSTRTD             YES - CON'T
         WTO   'HCEMCS - MUST BE INVOKED AS A CP',ROUTCDE=(11)
         B     RETOUT             RETURN
GSTRTD   DS    OH
         LR    R2,R1              SAVE CPPL PTR
         USING CPPL,R2
         L     R3,CPPLUPT         PICK UP UPT PTR
         L     R4,CPPLECT         PICK UP ECT PTR
         STM   R3,R4,PGUE         SAVE UPT/ECT PTRS
         DROP  R2
* -----  ENABLE OPERATOR COMMUNICATION - QEDIT PROLOGUE
         EXTRACT COMM,'S',FIELDS=(COMM)
         L     R11,COMM
         USING COMLIST,R11
         L     R10,COMECBPT        ECB
         ST    R10,CIBECB          ECB INTO ECBLIST
         LA    R9,COMCIBPT         ORIGIN
         QEDIT ORIGIN=(R9),CIBCTR=1
         LTR   R15,R15
         BZ    *+4+2
         DC    H'1'                ABEND OC1-1  - RC CHECKING - TBD
* -----  DELETE START CIB, IF ANY
         ICM   R8,15,COMCIBPT
         BZ    CONT1
         BAL   R5,PCIBFR
         DROP  R11
CONT1    DS    OH
* ------ SET ATTENTION EXIT
         STAX  AXIT,USADDR=BS,REPLACE=NO,DEFER=NO,                     C
               TOPLEVL=YES,MF=(E,STAXL)
* ------ MANAGE EXTENDED MCS CONSOLE
         L     R1,X'21C'(,0)       BUILD..
         L     R1,X'00C'(,R1)      .EMCS CONSOLE..
         MVC   CN,0(R1)            ..NAME.
         BAL   R14,SETSM
* Note: The attrubutes of the EMCS console are hardcoded in the
*       OPERPARM data area 'OP' rather than set dynamically.
         MCSOPER REQUEST=ACTIVATE,NAME=CN,CONSID=ID,TERMNAME=TN,       C
               MCSCSA=CS,MCSCSAA=AL,MSGDLVRY=FIFO,MSGECB=MSGECB,       C
               OPERPARM=OP
         BAL   R14,SETPM
         LTR   R15,R15             O.K?
         BZ    *+4+2               YES - CONTINUE
         DC    H'11'               ABEND OC1-11  - RC CHECKING - TBD
* ------ ISSUE INITIAL COMMAND - USE PSEUDO MASTER ID (0)
         BAL   R14,SETSM
         MVC   CM+7(8),CN        MOVE CONSNAME INTO D C,CN=
         MGCRE MF=(E,ML),TEXT=CMA,CONSID=ZERO
         BAL   R14,SETPM
* -----  WAIT FOR SOMETHING TO HAPPEN - CIB OR MESSAGE POST
ACWT     DS    OH
         WAIT  1,ECBLIST=ECBL
         XR    R14,R14             SET BRANCH INDEX TO 0
         LA    R1,ECBL             POINT AT ECBLIST
* -----  CHECK ECBS
CECB     DS    OH
         L     R15,0(,R1)          POINT AT 1ST ECB
         TM    0(R15),X'40'        POSTED?
         BO    BRVE                YES - PROCESS
         TM    0(R1),X'80'         LAST ECB IN ECBLIST
         BO    ACWT                YES - BACK TO WAIT
         LA    R14,4(,R14)         INCREMENT BRANCH INDEX
         LA    R1,4(,R1)           POINT AT NEXT ECB IN THE LIST
         B     CECB                LOOP...
BRVE     B     *+4(R14)            BRANCH TO SERVICE..
         B     PCIB                .CIB PROCESSING
         B     RET                 ..TSO ATTN PROCESSING
         B     PMSG                ...MESSAGE ARRIVED
* ------ CIB PROCESSING
PCIB     DS    OH
         L     R11,COMM            GET..
         USING COMLIST,R11         .ADDRESSABILITY..
         L     R8,COMCIBPT         ..TO CIB.
         USING CIBNEXT,R8
         CLI   CIBVERB,CIBMODFY    MODIFY COMMAND ?
         BNE   PCIBST              NO  - CHECK FOR STOP COMMAND
         CLC   =C'END',CIBDATA     'END' REQUESTED ?
         BE    PCIBRE              YES
         CLC   =C'STOP',CIBDATA    'END' REQUESTED ?
         BE    PCIBRE              YES
* ------ F JOBNAME,TEXT - TEXT NOT CORRECT - ISSUE MESSAGE
         LH    R9,CIBXOFF          FIND NAME OF THE..
         LA    R9,CIBNEXT(R9)      .CONSOLE THAT ISSUED..
         LA    R9,CIBXCNNM-CIBX(,R9) THE MODIFY.
         WTO   'HCEMCS - WHAT?',CONSNAME=(9)
         B     PCIBDL              CHECK FOR WORK
PCIBRE   DS    OH
         BAL   R5,PCIBFR           FREE CIB
         B     RET                 CHECK FOR WORK
PCIBST   DS    OH
         CLI   CIBVERB,CIBSTOP     STOP COMMAND ?
         BE    PCIBRE              YES  - QUIT
* ------ DELETE PROCESSED CIB
PCIBDL   DS    OH
         BAL   R5,PCIBFR           FREE CIB
         B     ACWT
PCIBFR   DS    OH
         LA    R9,COMCIBPT         ORIGIN
         QEDIT ORIGIN=(R9),BLOCK=(R8)
         BR    R5
         DROP  R8,R11
* ------ RETRIEVE QUEUED MDB FROM THE EMCS CONSOLE
*    MDB structure for a message with more than one MDB.
*           ________________          _________________
*          |       MDBPNEXT |---+    |                 |
* MCSOPMSG |   MDB prefix   |   |    |   MDB prefix    |
*  GETMSG  |   (IEAVG132)   |   |    |   (IEAVG132)    |
```

```
* R1,AR1-->|----------------|   +-->|----------------|
*          |     MDB #1     |       |     MDB #2     |
*          |   (IEAVM105)   |       |   (IEAVM105)   |
*          |+--------------+|       |+--------------+|
*          ||General object||       ||General object||
*          |+--------------+|       |+--------------+|
*          |+--------------+|       |+--------------+|
*          ||Control object||       ||Text object N+||
*          |+--------------+|       |+--------------+|
*          |+--------------+|       |+--------------+|
*          ||Text object 1 ||       ||Text object   ||
*          |+--------------+|       |+--------------+|
*
PMSG     DS    OH
         MVI   MSGECB,0           DROP POSTING
PMSGR    DS    OH
         BAL   R14,SETSM
         SAC   512                SET AR MODE
         LAM   AR1,AR1,=F'0'      SET HOME
         MCSOPMSG REQUEST=GETMSG,CONSID=ID
* ON RETURN, WHEN YOUR PROGRAM IS IN AR MODE, GPR 1 CONTAINS
* THE ADDRESS OF AN MDB, AR 1 CONTAINS THE ALET FOR THE MDB
         CH    R15,=H'8'          MESSAGES?
         BE    PMSGB              NO - RETURN
         BL    *+4+2              YES - JUMP
         DC    H'13'              ABEND OC1-13  - RC CHECKING - TBD
* ------ OLD FASHION LOOP TO MOVE MDB INTO WORK ARE
         LA    R4,WA              POINT AT WORK AREA
PMSGD    DS    OH
         LA    R6,0(,R1)          SAVE MDB POINTER
         LR    R7,R4              SAVE MOVED MDB POINTER
         LH    R2,0(,R1)          MDB LENGTH
         LA    R14,0(R2,R4)       POINT PAST MESSAGE MDB
         MVC   0(4,R14),=F'-1'    END OF MDB MARKER
         LA    R3,256             MAX MVC LTH
PMSGA    DS    OH
         CR    R2,R3              LOOP..
         BH    *+4+2              .TO..
         LR    R3,R2              ..MOVE..
         LR    R5,R3              ...MDB..
         BCTR  R5,0               ....DATA..
         MVC   0(0,R4),0(R1)      .....INTO..
         EX    R5,*-6             ......WORK..
         AR    R1,R3              .......AREA..
         AR    R4,R3              ........FROM..
         SR    R2,R3              .........THE MESSAGE..
         BP    PMSGA              ..........DATA SPACE
* ------ CHECK FOR CHAINED MDB - MDBPNEXT
         SH    R6,=AL2(MDBPLNNO)  MDBPRFX ADDR
         LR    R1,R6
         USING MDBPRFX,R1
         ICM   R14,15,MDBPNEXT    PICK UP NEXT MDB POINTER..
         ST    R14,MDBN           .SAVE IT AND..
         BZ    PMSGC              ..PROCESS ONLY MDB.
         DROP  R1
* ------ PREPARE TO MOVE NEXT MDB
         AH    R7,MDBLEN-MDB(,R7) POINT TO NEXT MDB IN WORK
         LR    R4,R7             COPY POINTER
         L     R1,MDBN            POINT TO NEXT MDB IN DS
         B     PMSGD              TO MOVE MDB
* ------ MDB MOVE COMPLETE
PMSGC    DS    OH
         LAM   AR1,AR1,=F'0'      SET HOME
         SAC   0                  RESET AR MODE
         BAL   R14,SETPM
         B     PMDB               PROCESS MDB
PMSGB    DS    OH
         LAM   AR1,AR1,=F'0'      SET HOME
         SAC   0                  RESET AR MODE
         BAL   R14,SETPM
         B     ACWT
* ------ PROCESS MDB OBJECTS - OUTPUT MESSAGE TEXT
* Note: The output message header is not a one-to-one copy of
*       the MVS syslog message header (IHAHCLOG).  Only some
*       of the fields are included.  See IEAMDBLG for details
*       on how to create a syslog message header from the MDB
*       data.
PMDB     DS    OH
         LA    R5,WA               POINT AT MDB DATA
         USING MDB,R5
PMDBL    DS    OH
         CLC   MDBMID,=CL4'MDB'    MDB?
         BNE   ALLDO               NO
         MVI   PGTXT,C' '
         MVC   PGTXT+1(PMTX-PMSY-1),PGTXT CLEAR MSG HDR
         LH    R6,MDBLEN           GET TOTAL MDB LTH
         LA    R6,MDB(R6)          POINT PAST MDB
         LA    R7,MDB+MDBHLEN      POINT PAST HEADER OBJECT
* ------ LOOP FOR ALL MDB OBJECTS
NXTMO    DS    OH
         CR    R7,R6               ALL DONE?
         BNL   MAYBE               YES - PERHAPS
* ------ PROCESS MDB GENERAL OBJECT
         USING MDBG,R7
         CLC   MDBGTYPE,=AL2(MDBGOBJ) GENERAL OBJECT?
         BNE   NOTGO
         C     R5,=A(WA)           FIRST MDB?
         BNE   NOTGO               NO - SKIP
         MVC   PMSY,MDBGOSNM       SYSTEM NAME
         MVC   PMYD,MDBGDSTP       DATE (YYYYDDD)
         MVC   PMTS,MDBGTIMH       TIME (HH.MM.SS.TH)
         B     NOTTO               BUMP TO NEXT OBJECT
NOTGO    DS    OH
* ------ PROCESS MDB CONTROL PROGRAM OBJECT
         USING MDBSCP,R7
         CLC   MDBCTYPE,=AL2(MDBCOBJ) CONTROL PROGRAM OBJECT?
         BNE   NOTCO
         B     NOTTO               BUMP TO NEXT OBJECT
NOTCO    DS    OH
* ------ PROCESS MDB TEXT OBJECT
         USING MDBT,R7
         CLC   MDBTTYPE,=AL2(MDBTOBJ) MESSAGE TEXT OBJECT?
         BNE   NOTTO
         LA    R1,MDBTMSGT        POINT AT TEXT
         LH    R15,MDBTLEN        COMPUTE..
         SH    R15,=AL2(MDBTMBOB) .TEXT LENGTH..
         LA    R0,4+PMTX-PMSY(,R15) AND PUTLINE..
         STH   R0,PGMSG           ...LENGTH
         LTR   R15,R15            0 (ZERO) LENGTH MESSAGE?
         BZ    NOMOV              YES - NOTHING TO MOVE
         BCTR  R15,0              MOVE..
         MVC   PMTX(1),0(R1)      .MESSAGE INTO..
         EX    R15,*-6            ..PUTLINE AREA.
NOMOV    DS    OH
* ------ OUTPUT MESSAGE
         BAL   R14,TPUT           ISSUE MESSAGE
         MVI   PGTXT,C' '
         MVC   PGTXT+1(PMTX-PMSY-1),PGTXT CLEAR MSG HDR
NOTTO    DS    OH
* ------ ADVANCE TO NEXT MDB OBJECT
         OC    MDBTLEN,MDBTLEN    OUT OF SYNCH?
         BNZ   *+4+2              NO - KEEP GOING
         DC    H'14'              ABEND OC1-14  - RC CHECKIN - TBD
         AH    R7,MDBTLEN         POINT AT NEXT OBJECT
         B     NXTMO
* ------ MDB PROCESSED - TRY TO OBTAIN NEXT MDB
MAYBE    DS    OH
         CLC   0(4,R6),=F'-1'     END OF MDB MARKER?
         BE    ALLDO              YES - DONE
         LR    R5,R6              POINT AT NEXT MDB
         B     PMDBL              TO PROCESS NEXT MDB
* ------ MDB PROCESSED - TRY TO OBTAIN NEXT MESSAGE
ALLDO    DS    OH
         DROP  R5,R7              MDB,MDBT
         B     PMSGR              GET NEXT MDB
* ------ ALL DONE - CLEAN UP - RETURN ETC.
RET      DS    OH
         L     R11,COMM
         USING COMLIST,R11
         LA    R9,COMCIBPT        ORIGIN
         QEDIT ORIGIN=(R9),CIBCTR=0
         DROP  R11
* ------ DEACTIVATE EMCS CONSOLE
         BAL   R14,SETSM
         MCSOPER REQUEST=DEACTIVATE,CONSID=ID
         BAL   R14,SETPM
         STAX  ,                  CANCEL ATTENTION
RETOUT   DS    OH
         L     R13,4(,R13)
         RETURN (14,12),RC=0
*------- PUTLINE SUBROUTINE - OUTPUT MESSAGES
TPUT     DS    OH
         ST    R14,0(,R13)
         MVI   PGECB,0
         LM    R3,R4,PGUE         GET UPT/ECT PTRS
         PUTLINE PARM=PLL,UPT=(3),ECT=(4),ECB=PGECB,                   C
```

```
               TERMPUT=(EDIT,WAIT,NOHOLD,NOBREAK),
               OUTPUT=(PGMSG,TERM,SINGLE,DATA,NOTRANS),MF=(E,PGIOPL)
         L     R14,0(,R13)
         BR    R14
*------- SET SYSTEM MODE
SETSM    DS    0H
         STM   R14,R2,12(R13)      SAVE A FEW REGS
         MODESET KEY=ZERO,MODE=SUP
         LM    R14,R2,12(R13)      .RESTRE A FEW REGS
         BR    R14                 ..BACK TO CALLER
*------- RESET PROBLEM MODE
SETPM    DS    0H
         STM   R14,R2,12(R13)      SAVE A FEW REGS
         MODESET KEY=NZERO,MODE=PROB
         LM    R14,R2,12(R13)      .RESTRE A FEW REGS
         BR    R14                 ..BACK TO CALLER
         DROP  R12
*------- ATTENTION EXIT
AXIT     DS    0H
         USING AXIT,R15
         STM   R14,R12,12(R13)
         POST  ATTECB,123          TELL MAINLINE HAVE AN ATTN
         DROP  R15
         LM    R14,R12,12(R13)
         BR    R14
* -----  SAVE AREA
SA       DC    18F'0'
* -----  ECB LIST AND ECBS
ECBL     DS    OF                  ECBLIST
CIBECB   DC    A(0)                QEDIT ECB
         DC    A(ATTECB)           ATTENTION ECB
         DC    A(MSGECB+X'80000000')
MSGECB   DC    A(0)                MESSAGE ECB
* -----  PUTLINE DATA AREAS
PGUE     DC    2F'0'
PGECB    DC    F'0'
PGIOPL   DC    4F'0'
PLL      PUTLINE MF=L
GLL      GETLINE MF=L
PGMSG    DC    AL2(L'PGTXT+4)
         DC    AL2(0)
PGTXT    DC    CL256' '
         ORG   PGTXT
PMSY     DC    CL8' '
         DC    CL1' '
PMYD     DC    CL7' '
         DC    CL1' '
PMTS     DC    CL11' '
         DC    CL1' '
PMTX     DC    C' '
         ORG
TXL      DC    A(L'PGTXT)
* -----  QEDIT DATA AREA
COMM     DC    1F'0'               EXTRACT ANSWER
* -----  EMCS CONSOLE DATA AREAS
CN       DC    CL8'       '        EXTENDED MCS CONSOLE NAME
TN       DC    CL8'HCMCSCON'       TERMNAME
ZERO     DC    F'0'                4-BYTE CONSOLE ID
ID       DC    F'0'                4-BYTE CONSOLE ID
CS       DC    F'0'                MCS CONSOLE STATUS AREA
AL       DC    F'0'                ALET FOR PREVIOUS
ME       DC    F'0'                MESSAGE QUEUED ECB
MDBN     DC    F'0'                NEXT MDB FOR MESSAGE
* -----  MGCRE DATA AREAS
ML       MGCRE MF=L
CMA      DC    A(CML)              MGCRE..
CML      DC    AL2(L'CM)           .PARAMETER..
CM       DC    CL32'D C,CN= '        ..LIST
DW       DS    D
* -----  ATTENTION EXIT DATA AREA
BS       DS    0F
ATTECB   DC    F'0'
STAXL    STAX  AXIT,USADDR=BS,REPLACE=NO,DEFER=NO,MF=L
* -----  OPERPARM DATA AREA
OP       DS    0F
* ------------------------------------------------------------------
* MESSAGE DATA SPACE SIZE - THE MAXIMUM SIZE FOR THE MESSAGE DATA
*                           SPACE, IN MEGABYTES.
* ------------------------------------------------------------------
XYZOSTOR DC    H'150'      LIMIT VALUE
* ------------------------------------------------------------------
* AUTHORITY LEVEL - TWO BIT FLAGS REPRESENTING THE AUTHORITY LEVELS.
* ------------------------------------------------------------------
XYZOAUTH DS    0XL2        AUTHORITY LEVEL
XYZOATH1 DC    XL1'80'      AUTHORITY FLAG 1
XYZOMSTR EQU   X'80'         MASTER
XYZOAALL EQU   X'40'         ALL (SYS,IO, AND CONS)
XYZOASYS EQU   X'20'         SYS
XYZOAIO  EQU   X'10'         I/O
XYZOCONS EQU   X'08'         CONS
XYZOINFO EQU   X'04'         INFO (DEFUALT)
XYZOATH2 DC    XL1'00'      AUTHORITY FLAG 2 - RESERVED
* ------------------------------------------------------------------
* MESSAGE FORM - INDICATES HOW A MESSAGE IS DISPLAYED.
* ------------------------------------------------------------------
XYZOMFRM DS    0XL2        OPERATORS MESSAGE FORM
XYZOMFM1 DC    XL1'E0'      MESSAGE FORM FLAG 1
XYZOMFT  EQU   X'80'         DISPLAY WITH A TIME STAMP
XYZOMFS  EQU   X'40'         DISPLAY WITH THE SYSTEM NAME
XYZOMFJ  EQU   X'20'         DISPLAY WITH JOB ID/NAME
XYZOMFM2 DC    XL1'00'      MESSAGE FORM FLAG 2
* ------------------------------------------------------------------
* MESSAGE LEVEL - THE LEVEL OF MESSAGES TO BE RECEIVED BY THE CONSOLE.
* ------------------------------------------------------------------
XYZOMLVL DS    0XL2        MESSAGE LEVEL
XYZOMLV1 DC    XL1'02'      MESSAGE LEVEL FLAG 1
XYZOMLAL EQU   X'02'         RECEIVE ALL MESSAGE LEVELS (DEFAULT)
XYZOMLV2 DC    XL1'00'      MESSAGE LEVEL FLAG 2
* ------------------------------------------------------------------
* MESSAGE TYPE - THIS IS THE MONITOR VALUE. IT INDICATES WHAT EVENTS
*                THE CONSOLE WILL MONITOR.
* ------------------------------------------------------------------
XYZOMSGT DS    0XL2        MESSAGE TYPE
XYZOMTP1 DC    XL1'00'      MESSAGE TYPE FLAG 1
XYZOMTP2 DC    XL1'00'      MESSAGE TYPE FLAG 2
* ------------------------------------------------------------------
* ROUTING CODES -  A 128 BIT STRING WHERE EACH BIT REPRESENTS A
*                  ROUTE CODE. A FLAG IS INCLUDED FOR ALL AND NONE.
* ------------------------------------------------------------------
XYZORCDT DS    OCL17       ROUTING CODE DATA
XYZORCFL DC    XL1'40'      ROUTING CODE FLAG
XYZORCAL EQU   X'80'         ALL ROUTING CODES
XYZORCNO EQU   X'40'         NO ROUTING CODES (DEFAULT)
XYZORTCD DC    CL16'0'      ROUTING CODES (IF NOT ALL OR NONE)
* ------------------------------------------------------------------
* LOG COMMAND RESPONSE - SHOULD THE COMMAND RESPONSE OF A CONSOLE
*                        BE LOGGED IN THE MCS HARDCOPY LOG.
* ------------------------------------------------------------------
XYZOLOGC DC    XL1'80'     LOG COMMAND RESPONSE VALUE
XYZOLOGS EQU   X'80'         SYSTEM - LOG THE RESPONSE (DEFAULT)
* ------------------------------------------------------------------
* MIGRATION ID - SHOULD THE CONSOLE BE ASSIGNED A 1 BYTE MIGRATION ID.
* ------------------------------------------------------------------
XYZOMIG  DC    XL1'40'     MIGRATION ID FLAGS
XYZOMIGY EQU   X'80'         YES - ASSIGN AN ID
XYZOMIGN EQU   X'40'         NO  - DO NOT ASSIGN AN ID
* ------------------------------------------------------------------
* DOM - INDICATES WHAT TYPE, IF ANY, OF DELETE OPERATOR MESSAGE
*       (DOM) THE CONSOLE WILL RECEIVE. NORMAL WILL QUEUE DOMS BY
*       THE MESSAGE QUEUING CRITERIA. ALL WILL QUEUE ALL DOMS.
*       NONE WILL KEEP AND DOMS FROM BEING SENT TO THE CONSOLE.
* ------------------------------------------------------------------
XYZODOM  DC    XL1'20'     DOM VALUE
XYZODOMX EQU   X'20'         NONE
* ------------------------------------------------------------------
* KEY - THE EIGHT BYTE CHARACTER NAME USED TO ASSOCIATE GROUPS OF
*       CONSOLES.
* ------------------------------------------------------------------
XYZOKEY  DC    CL8'HCMCS'     KEY ASSIGNED TO CONSOLE ENTRY
* ------------------------------------------------------------------
* COMMAND SYSTEM - THE SYSTEM WHERE ALL COMMANDS FROM THIS CONSOLE
*                  WILL BE SENT TO EXECUTE. A '*' WILL BE
*                  CONVERTED TO THE CURRENTLY EXECUTING SYSTEM
*                  NAME.
* ------------------------------------------------------------------
XYZOCSNM DC    CL8'*'          COMMAND SYSTEM NAME
* ------------------------------------------------------------------
* ALTGRP - THE ALTERNATE GROUP OF CONSOLES THAT WILL BE SELECTED
*          FOR BACKUP IN THE EVENT OF CONSOLE FAILURE
* ------------------------------------------------------------------
XYZOALGP DC    CL8' '      ALTERNATE GROUP NAME
* ------------------------------------------------------------------
* MSCOPE DATA - THE SYSTEMS FOR WHICH THIS CONSOLE IS ELIGIBLE TO
*               RECEIVE MESSAGES FROM.  IF THE CONSOLE IS TO BE
*               SCOPED TO ALL SYSTEMS, THEN THE USER SETS
*               MCSOSALL ON.  IF A SPECIFIC LIST OF SYSTEM NAMES
```

```
*              IS TO BE SPECIFIED, THEN MCSOSLST IS SET ON AND
*              MCSOMSPT IS SET TO THE ADDRESS OF A STRUCTURE
*              CONTAINING A LIST OF SYSTEMS.  THIS STRUCTURE IS
*              MAPPED BY THE DSECT MCSOTBL.
* ------------------------------------------------------------------
XYZOMSFG DC    X'80'       MSCOPE FLAGS
XYZOSALL EQU   X'80'       *ALL SPECIFIED FOR MSCOPE
         DC    X'00'       RESERVED FOR ALIGNMENT
XYZOMSPT DC    A(0)        POINTER TO A LIST OF MSCOPE VALUES
XYZOMISC DC    XL1'48'     MISCELLANEOUS ROUTING INFORMATION
*------------------------------------------------------------------
* UNDELIVERED - SHOULD THE CONSOLE BE SENT UD MESSAGES OR NOT.
*------------------------------------------------------------------
XYZOUDN  EQU   X'40'       NO - DON'T DISPLAY UD MESSAGES  DEFAULT
*------------------------------------------------------------------
* HARDCOPY - SHOULD THE CONSOLE BE SENT THE HARDCOPY MESSAGE SET
*------------------------------------------------------------------
XYZOHDCY EQU   X'08'        YES - RECEIVE HARDCOPY MESSAGE SET
XYZOHDCN EQU   X'04'        NO - DON'T RECEIVE HARDCOPY MESSAGE SET
*------------------------------------------------------------------
         SPACE ,
XYZOFLAG DC    XL1'80'     FLAGS BYTE
XYZOVRDY EQU   X'80'       YES - OVERRIDE SECURITY PRODUCT
XYZOVRDN EQU   X'40'       NO - DON'T OVERRIDE SECURITY PRODUCT
         DC    XL6'00'     RESERVED
* ------------------------------------------------------------------
* TABLE OF MESSAGE SCOPE VALUES
* ------------------------------------------------------------------
XYZOMSNM DC     F'1'         NUMBER OF MSCOPE VALUES SPECIFIED
XYZOTSYS DC     CL8'*ALL'    STORAGE FOR 8 SYSTEM NAMES
* ------
         LTORG
* ------
WA       DS     32XL(4096)  MDB WORK AREA
* ------
         IEAVM105
         IEAVG132
         PRINT NOGEN
         IEZVG111
         IEZCOM
         IEZCIB
         IKJTCB
         IKJCPPL
         IKJGTPB
         CVT   DSECT=YES
         END   HCEMCS
```

Appendix I. Release Migration IDs

The following REXX program must run in a TSO session. It lists the assigned migration IDs and the extended MCS consoles to which the IDs are assigned. The REXX program has three processing options: LIST which only lists the status of assigned migration IDs, FREE which in addition to listing the IDs attempts to release them, and HELP (?) which returns “how-to-use” information.

While extracting the information, the REXX program calls the “CNNMID” TSO command processor that returns the console name for a console ID and vice versa. If a request for release migration IDs is made, “FREMIGID” TSO command processor is called to free the ID. The source code for both CNNMIF and FREMIGID is included in this sample.

Internet access to code

Internet access to source code found in GG24-4626 can be accessed via the Internet.

http://www.redbooks.ibm.com/redbooks

Download sample code from our redbooks.

or you can:

You can access the server by anonymous FTP:

ftp ftp.almaden.ibm.com
user: anonymous
password: your E-mail address
cd redbooks/SG244626
get migrid.code

```
/* REXX - Find in use MIGIDs and their users             */
trace "O"
/* --- */
if sysvar("SYSISPF") = "ACTIVE" then ,
  address ISPEXEC "CONTROL DISPLAY SM"
/* --- */
Parse upper arg opt .
Select
  when opt = "LIST" | opt = "" then Call Proc
  when opt = "FREE" then Call Proc
  when opt = "?" | opt = "HELP" then Call Help
  otherwise say "Bad input parameter!"
End
Exit 0
/*  Help                                               */
Help:
h.1 = "Warning:                                       "
h.2 = "  This program uses fields that are not part of "
h.3 = "  general-use programming interfaces.          "
h.4 = "Disclaimer:                                    "
h.5 = "  This sample source is provided for tutorial  "
h.6 = "  purposes only. A complete handling of error  "
h.7 = "  conditions has not been shown or attempted, and"
h.8 = "  this source has not been submitted to formal "
h.9 = "  IBM testing. This source is distributed on an "
h.10= "  'as is' basis without any warranties either  "
h.11= "  expressed or implied.                        "
h.12= "Invocation:                                    "
h.13= "  From TSO execute REXX exec 'MITPROC'.  The exec"
h.14= "  accepts one parameter with the value of LIST or"
h.15= "  FREE:                                        "
h.16= "  - LIST (the default) requests listing of the "
h.17= "    MIGID users                                "
h.18= "  - FREE requests listing of the MIGID users and "
h.19= "    frees all MIGIDs that are assigned to      "
h.20= "    inactive extended MCS consoles             "
Do i = 1 to 20
 say h.i
End
Return
/*  Chase control blocks and tell about migration IDs  */
Proc:
cvt     = c2x(storage(10,4))
cvt_64  = d2x(x2d(cvt)+x2d(64))
ucm     = c2x(storage(cvt_64,4))
ucm_F0  = d2x(x2d(ucm)+x2d(F0))
ucmf    = c2x(storage(ucm_F0,4))
ucmf_C0 = d2x(x2d(ucmf)+x2d(C0))
MIT     = c2x(storage(ucmf_C0,4))
MIT_8   = d2x(x2d(MIT)+x2d(8))
mitid   = storage(MIT,3)
/*  Find 4-byte console IDs for MIGIDs                 */
If  mitid = "MIT" then do
  Do i = 1 to 149
    CIDP = d2x(x2d(MIT_8)+(i*4))
    CID  = c2x(storage(CIDP,4))
    m = 99 + i
    If m > 127 then m = m + 1       /* Skip x'80' special */
```

```
    If CID <> "00000000"  then do
/*  Process 4-byte console IDs for a MIGID               */
      Queue "ID="CID               /* Obtain..          */
      Queue "END"                  /* .console..        */
      x = outtrap("cr.","*")       /* ..name...         */
      "CNNMID"                     /* ...for the ID.    */
      x = outtrap("OFF")           /* Determine..       */
      ST = "Active"                /* .active/inactive.. */
      Do j = 1 to cr.0             /* ..status of the cn.*/
        If word(cr.j,1) = "HEX" then iterate
        If substr(cr.j,1,4) = "DRC=" then do
          Parse var cr.j "DRC=" crc "DRSN=" crsn
          If crc = 4 then ST = "Inactive"
          Else ST = "*** ERROR ***" /* Bad input?????     */
          Iterate
        End
        cn = left(strip(cr.j),8)    /* Say result        */
        Say CID left(m,3) cn "("right(i,3)")" ST
```

```
      If opt = "FREE" & ST = "Inactive" then do
        Queue m                    /* MIGID to be..     */
        Queue "END"                /* .released.        */
        x = outtrap("cr.","*")     /*                   */
        "FREMIGID"
        x = outtrap("OFF")         /*                   */
        Do j = 1 to cr.0           /* Say RELEASE outcome*/
          If word(cr.j,1) = "HEX" then iterate
          If word(cr.j,1) = "3-DIGIT" then iterate
          Say cr.j
        End
      End
    End
  End
End
Else say "No MIT - Should not happen!"
Return
```

The attached source for the CNNMID TSO command processor provides an example of how to use the CONVCON macro service to return console name for a console ID or console ID for a console name.

```
         TITLE  'TSO COMMNAD PROCESSOR - CONSOLE NAME/ID'
CNNMID   CSECT
CNNMID   AMODE 24
CNNMID   RMODE 24
* ------
*
*  Function:
*    Return 4-byte console ID for console name and vice versa.
*  Disclaimer:
*    This sample source is provided for tutorial purposes only. A
*    complete handling of error conditions has not been shown or
*    attempted, and this source has not been submitted to formal
*    IBM testing. This source is distributed on an 'as is' basis
*    without any warranties either expressed or implied.
*  Invocation:
*    As TSO command processor 'CNNMID'.  No parameters required or
*    processed, a prompt message is issued to solicit input.
*  Function:
*   - Prompt for a console name or an ID
*   - Invoke CONVCON macro to request name to ID or ID to name
*     conversion.
*   - Show non-zero return and reason codes
*   - Show name for an ID or ID for a name
*   - Keep prompting for new name/ID input until END requested
*
*  LE ATTR: LET
*
* ------
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         SPACE 1
* -----
         SAVE  (14,12),,&SYSDATE.-&SYSTIME./CNNMID
         USING CNNMID,R12
         LR    R12,R15
         ST    R13,SA+4
         LA    R3,SA
         ST    R3,8(,R13)
         LR    R13,R3
* -----
         TM    0(R1),X'80'
         BZ    GSTRTD
         WTO   'CNNMID - MUST BE INVOKED AS A CP',ROUTCDE=(11)
         B     RETU
* -----
GSTRTD   DS    OH
         LR    R2,R1               SAVE CPPL PTR
         USING CPPL,R2
         L     R3,CPPLUPT          PICK UP UPT PTR
         L     R4,CPPLECT          PICK UP ECT PTR
         STM   R3,R4,PGUE          SAVE UPT/ECT PTRS
         DROP  R2
* ------ PROMPT FOR AN CONSOLE ID OR NAME
PRMPT    DS    OH
         LA    R1,=CL68'HEX 4-BYTE CONSOLE ID (ID=) OR 8-BYTE CONSOLE N*
               ANE (NM=) OR END?'
         LA    R0,68
         BAL   R14,TPUT            PROMPT - (R1=ADDR),(R0=LTH)
         MVC   RESP,=CL12' '
         LA    R1,RESP
         LA    R0,14
         BAL   R14,TGET            READ RESPONSE - (R1=ADDR),(R0=LTH)
         LTR   R15,R15             READ OK?
         BNZ   PRMPT               NO - TRY AGAIN
         OC    RESP,=CL12' '
         CLC   RESP(4),=CL4'END'
         BE    RETU
       XC      CONV(CONVPLEN),CONV CLEAR CONVCON PARM LIST
         CLC   =CL3'ID=',RESP      ID REQUEST?
         BE    BYID                YES
         CLC   =CL3'NM=',RESP      NAME REQUEST?
         BE    BYNM                YES
         B     PRMPT               ASK AGAIN...
* ---- INIT CONVCON PARAMETERS FOR NAME CONVERSION
BYNM     DS    OH
       MVI     CONVFLGS,CONVPFLD   SET NAME TO ID CONVERSION
       MVC     CONVFLD,RESP+3      SET CONSOLE NAME
         B     CHKIN               GO TO COMMON CODE
* ---- INIT CONVCON PARAMETERS FOR ID CONVERSION
BYID     DS    OH
         TR    RESP+3(8),C2H       CHECK..
         LA    R1,8                .INPUT..
         LA    R2,RESP+3           ..FORMALLY CORRECT..
         CLI   0(R2),C' '          ...4-BYTE CONSOLE ID...
         BE    PRMPT               SORRY - BAD INPUT
         BCT   R1,*-4-4            FINALIZE CONSOLE ID..
         PACK  FW(5),RESP+3(9)     .MASSAGE
       MVI     CONVFLGS,CONVPID    SET ID CONVERSION
       MVC     CONVID,FW           SET CONSOLE ID (4-BYTE)
```

```
         B     CHKIN             GO TO COMMON CODE
* ---- ISSUE CONVCON MACRO REQUEST
CHKIN  DS      OH                CHECK IF CONSOLE ACTIVE USING CONVCON
       MVC     CONVACRO,=C'CONV' SET ACRONYM
       MVI     CONVVRSN,CONVRID  SET VERSION
       OI      CONVGFLG,CONVNPAR SET NO AREA VERIFICATION
       CONVCON CONV              CALL CONVCON
       LTR     R15,R15           OK?
       BNZ     NRB               NO - TELL RC
       B       NRC               RETURN RESULT
* ---- SHOW NON-ZERO RETURN AND REASON CODES
NRB      DS    OH
         XR    RO,RO
         IC    RO,CONVRSN
         LA    R1,=CL16'DRC=XX DRSN=XX'
         CVD   R15,DW
         OI    DW+L'DW-1,X'0F'
         UNPK  RESP(8),DW
         MVC   4(2,R1),RESP+6
         CVD   RO,DW
         OI    DW+L'DW-1,X'0F'
         UNPK  RESP(8),DW
         MVC   12(2,R1),RESP+6
         LA    RO,16
         BAL   R14,TPUT  (1),(0)
* ---- TELL NAME FOR ID AND VICE VERSA
NRC      DS    OH
       TM      CONVFLGS,CONVPFLD  NAME TO ID CONVERSION?
       BO      TELLID             YES - TELL ID
* ---- SAY NAME
         LA    R1,CONVNAME             LET'M..
         LA    RO,8                    .KNOW THE NAME.
         B     TELLIT
* ---- SAY ID
TELLID   DS    OH
         UNPK  DW(9),CONVID(5)         CONVERT..
         LA    R1,=C'0123456789ABCDEF' .BINARY CONS IN..
         SH    R1,=AL2(C'0')           ..SUITABLE..
         TR    DW,0(R1)                ...FOR..
         LA    R1,DW                   ....HUMAN..
         LA    RO,8                    .....EYES.
TELLIT   DS    OH
         BAL   R14,TPUT                SAY (1),(0)
         B     PRMPT
* ---- ALL DONE HERE
RETU     DS    OH
         L     R13,4(,R13)
         RETURN (14,12),RC=0
* ------ PUTLINE
TPUT     DS    OH
         MVI   PGECB,0
         STM   R14,R4,PGSA
         LR    R14,RO
         BCTR  R14,0
         MVC   PGTXT(0),0(R1)
         EX    R14,*-6
         AH    RO,=H'4'            TEXT LENGTH (+4)
         STH   RO,PGMSG
         MVI   PGECB,0
         LM    R3,R4,PGUE          GET UPT/ECT PTRS
         PUTLINE PARM=PLL,UPT=(3),ECT=(4),ECB=PGECB,
               TERMPUT=(EDIT,WAIT,NOHOLD,NOBREAK),
               OUTPUT=(PGMSG,TERM,SINGLE,DATA,NOTRANS),MF=(E,PGIOPL)
         L     R14,PGSA
         LM    RO,R4,PGSA+8
         BR    R14
* ------ GETLINE
TGET     DS    OH
         MVI   PGECB,0
         STM   R14,R4,PGSA
         LM    R3,R4,PGUE          GET UPT/ECT PTRS
         GETLINE PARM=GLL,UPT=(3),ECT=(4),ECB=PGECB,                   C
               TERMGET=(EDIT,WAIT),SUBSTACK=NO,                        C
               INPUT=(ISTACK,LOGICAL),MF=(E,PGIOPL)
         XR    R2,R2
         L     R1,GLL+4            POINT AT INPUT
         LH    RO,0(,R1)           GET INPUT LTH
         CH    RO,=H'4'            COMPUTE TEXT LENGTH
         BL    TGETRC
         BE    TGETRL
         LR    R15,RO
         SH    R15,=H'4'
         L     R3,PGSA+8
         CR    R3,R15
         BNL   *+4+2
         LR    R15,R3
         LR    R2,R15
         BCTR  R15,0
         L     R14,PGSA+12
         MVC   0(0,R14),4(R1)
         EX    R15,*-6
         STORAGE RELEASE,LENGTH=(0),ADDR=(1),SP=1
         B     TGETRL
TGETRL   DS    OH
         ST    R2,PGSA+12
TGETRO   DS    OH
         LM    R14,R4,PGSA
         XR    R15,R15
         BR    R14
TGETRC   DS    OH
         LM    R14,R4,PGSA
         LA    R15,12
         BR    R14
* ------
PGSA     DC    8F'0'
PGUE     DC    2F'0'
PGECB    DC    F'0'
PGIOPL   DC    4F'0'
PLL      PUTLINE MF=L
GLL      GETLINE MF=L
PGMSG    DC    AL2(L'PGTXT+4)
         DC    AL2(0)
PGTXT    DC    CL128' '
* ------
         DROP  R12                  CNNMID
* ------
         LTORG
SA       DC    18F'0'               SAVE AREA
MID      DC    X'00'
         DS    OF
        IEZVG200 DSECT=NO           CONVCON PARAMETER LIST
RESP     DC    CL12' '
         DC    CL4' '
DW       DC    D'0'
FW       DC    F'0'
HW       DC    H'0'
C2H      DC    CL256' '
         ORG   C2H+C'A'
 C       DC    X'FAFBFCFDFEFF'
 C       ORG   C2H+C'0'
         DC    C'0123456789'
         ORG
* ------
         IKJCPPL
         IKJGTPB
         END   CNNMID
```

The attached source for the FREMIGID TSO command processor provides an example on how to use the MCSOPER REQUEST=RELEASE macro service to return a migration ID for reuse.

```
        TITLE 'MIGID RELEASE - SAMPLE PROGRAM'
FREMIGID CSECT
FREMIGID AMODE 31
FREMIGID RMODE ANY
* ------
*       This program releases an extended MCS console's MIGID.
*       It must run on TSO session.
*
*       The program prompts for a 3-digit MIGID and then
*       issues an MCSOPER REQUEST=RELEASE macro for the
*       argument MIGID.
*
*       LE ATTR: AC(1)
*       NOTE:    IKJTSOxx AUTHCMD section must be updated
*                before the command can be used.
* ------
R0      EQU   0
R1      EQU   1
R2      EQU   2
R3      EQU   3
R4      EQU   4
R5      EQU   5
R6      EQU   6
R7      EQU   7
R8      EQU   8
R9      EQU   9
R10     EQU   10
R11     EQU   11
R12     EQU   12
R13     EQU   13
R14     EQU   14
R15     EQU   15
        SPACE 1
* -----
        SAVE  (14,12),,&SYSDATE.-&SYSTIME./FREMIGID
        USING FREMIGID,R12
        LR    R12,R15
        ST    R13,SA+4
        LA    R3,SA
        ST    R3,8(,R13)
        LR    R13,R3
* -----
        TM    0(R1),X'80'
        BZ    GSTRTD
        WTO   'FREMIGID - MUST BE INVOKED AS A CP',ROUTCDE=(11)
        B     RETU
* -----
GSTRTD  DS    OH
        LR    R2,R1             SAVE CPPL PTR
        USING CPPL,R2
        L     R3,CPPLUPT        PICK UP UPT PTR
        L     R4,CPPLECT        PICK UP ECT PTR
        STM   R3,R4,PGUE        SAVE UPT/ECT PTRS
        DROP  R2
* ------ PROMPT FOR AN MIGID
PRMPT   DS    OH
        LA    R1,=CL48'3-DIGIT MIGID (RANGE 100-127,129-250)? | END'
        LA    R0,48
        BAL   R14,TPUT          (R1=ADDR),(R0=LTH)
        MVC   RESP,=CL3' '
        LA    R1,RESP
        LA    R0,3
        BAL   R14,TGET          (R1=ADDR),(R0=LTH)
        LTR   R15,R15
        BNZ   PRMPT
* ------ PARSE RESPONSE
        CH    R1,=H'3'
        BNE   PRMPT
        OC    RESP,=CL3' '
        CLC   RESP,=CL3'END'    QUIT?
        BE    RETU              YES - QUIT
        CLC   RESP,=CL3'E  '    QUIT?
        BE    RETU              YES - QUIT
        LA    R2,RESP
* ------ MASSAGE MIGID FOR MCSOPER RELEASE
NLP     DS    OH
        TM    0(R2),X'F0'
        BNO   PRMPT
        LA    R2,1(,R2)
        BCT   R1,NLP
        CLC   RESP,=CL3'128'
        BE    PRMPT
        CLC   RESP,=CL3'100'
        BL    PRMPT
        CLC   RESP,=CL3'250'
        BH    PRMPT
        PACK  DW,RESP
        CVB   R2,DW
        STC   R2,MID
* ------ RELEASE MIGID
        BAL   R14,SETSM         GET AUTHORIZED USING USER SVC
        MCSOPER REQUEST=RELEASE,MIGID=MID
        BAL   R14,SETPM         RESET AUTHORIZATION
        MVC   RESPQ,=CL32' '    CLEAR INFO TEXT
        LTR   R15,R15           MCSOPER O.K?
        BNZ   NRD               NO
        LTR   R0,R0             O.K?
        BNZ   NRB               NO
        MVC   RESPQ(16),=CL16'RELEASED'
        B     NRT               NO
* ------ TELL THE BAD NEWS
NRD     DS    OH
        CH    R15,=AL2(X'1C')
        BNE   NRA
        MVC   RESPQ,=CL32'NOT ASSIGNED/INCORRECT MIGID'
        B     NRT
NRA     DS    OH
        CH    R15,=AL2(X'20')
        BNE   NRB
        MVC   RESPQ(16),=CL16'MIGID IN USE'
        B     NRT
NRB     DS    OH
        MVC   RESPQ(16),=CL16'DRC=XX DRSN=XX'
        LA    R1,RESPQ
        CVD   R15,DW
        OI    DW+L'DW-1,X'0F'
        UNPK  RCWR,DW
        MVC   4(2,R1),RCWR+1
        CVD   R0,DW
        OI    DW+L'DW-1,X'0F'
        UNPK  RCWR,DW
        MVC   12(2,R1),RCWR+1
        B     NRT
NRT     DS    OH
        LA    R1,RESP
        LA    R0,RESPQE-RESP
        BAL   R14,TPUT          (1),(0)
        B     PRMPT             LOOP UNTIL END REQUESTED
* ------ RETURN
RETU    DS    OH
        L     R13,4(,R13)
        RETURN (14,12),RC=0
* ------ PUTLINE
TPUT    DS    OH
        MVI   PGECB,0
        STM   R14,R4,PGSA
        LR    R14,R0
        BCTR  R14,0
        MVC   PGTXT(0),0(R1)
        EX    R14,*-6
        AH    R0,=H'4'          TEXT LENGTH (+4)
        STH   R0,PGMSG
        MVI   PGECB,0
        LM    R3,R4,PGUE        GET UPT/ECT PTRS
        PUTLINE PARM=PLL,UPT=(3),ECT=(4),ECB=PGECB,                    C
              TERMPUT=(EDIT,WAIT,NOHOLD,NOBREAK),                      C
              OUTPUT=(PGMSG,TERM,SINGLE,DATA,NOTRANS),MF=(E,PGIOPL)
        L     R14,PGSA
        LM    R0,R4,PGSA+8
        BR    R14
* ------ GETLINE
TGET    DS    OH
        MVI   PGECB,0
        STM   R14,R4,PGSA
        LM    R3,R4,PGUE        GET UPT/ECT PTRS
        GETLINE PARM=GLL,UPT=(3),ECT=(4),ECB=PGECB,                    C
              TERMGET=(EDIT,WAIT),SUBSTACK=NO,                         C
              INPUT=(ISTACK,LOGICAL),MF=(E,PGIOPL)
        XR    R2,R2
        L     R1,GLL+4          POINT AT INPUT
        LH    R0,0(,R1)         GET INPUT LTH
        CH    R0,=H'4'          COMPUTE TEXT LENGTH
        BL    TGETRC
        BE    TGETRL
        LR    R15,R0
        SH    R15,=H'4'
```

```
         L     R3,PGSA+8
         CR    R3,R15
         BNL   *+4+2
         LR    R15,R3
         LR    R2,R15
         BCTR  R15,0
         L     R14,PGSA+12
         MVC   0(0,R14),4(R1)
         EX    R15,*-6
         STORAGE RELEASE,LENGTH=(0),ADDR=(1),SP=1
         B     TGETRL
TGETRL   DS    0H
         ST    R2,PGSA+12
TGETRO   DS    0H
         LM    R14,R4,PGSA
         XR    R15,R15
         BR    R14
TGETRC   DS    0H
         LM    R14,R4,PGSA
         LA    R15,12
         BR    R14
* ------
PGSA     DC    8F'0'
PGUE     DC    2F'0'
PGECB    DC    F'0'
PGIOPL   DC    4F'0'
PLL      PUTLINE MF=L
GLL      GETLINE MF=L
PGMSG    DC    AL2(L'PGTXT+4)
         DC    AL2(0)
PGTXT    DC    CL128' '
* ------ SET APF AUTHORIZATION
SETSM    DS    0H
         STM   R14,R0,12(R12)
         MODESET MODE=SUP,KEY=ZERO
         LM    R14,R0,12(R12)
         BR    R14                 .....BACK TO CALLER
* ------ SET MODE=PROB.
SETPM    DS    0H
         STM   R14,R0,12(R12)
         MODESET MODE=PROB,KEY=NZERO
         LM    R14,R0,12(R12)
         BR    R14                 .....BACK TO CALLER
* ------
         DROP  R12                 FREMIGID
* ------
         LTORG
SA       DC    18F'0'              SAVE AREA
MID      DC    X'00'
RESP     DC    CL3' '
         DC    CL1' '
RESPQ    DC    CL32' RELEASED'
RESPQE   DS    0C
RCWR     DS    CL3
DW       DC    D'0'
         PRINT NOGEN
         IKJCPPL
         IKJGTPB
         IKJTCB
         CVT   DSECT=YES
         END   FREMIGID
```

Appendix J. Authorizing TSO Users for the CONSOLE Command

The following steps allow TSO/E users to issue the TSO/E CONSOLE and CONSPROF commands to activate an extended MCS console.

Note: You must use a RACF controlled TSO LOGON as opposed to a UADS LOGON.

In the example, it is assumed that the RACF user profile is already defined:

- Issue SETROPTS to activate the TSOAUTH resource class:

  ```
  SETROPTS CLASSACT(TSOAUTH)
  ```

- 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. To reset your access to the CONSOLE resource in the TSOAUTH class, issue command:

  ```
  PERMIT CONSOLE CLASS(TSOAUTH) RESET(ALL)
  ```

- Issue RACF PERMIT to authorize the user to use the CONSOLE command:

  ```
  PERMIT CONSOLE CLASS(TSOAUTH) ID(user_ID) ACCESS(READ)
  ```

  To limit from which TSO/E terminal the user can initiate an extended MCS console session, you may specify the following:

  ```
  PERMIT CONSOLE CLASS(TSOAUTH) ID(TAPE1) ACCESS(READ)
   WHEN(TERMINAL(terminal_ID))
  ```

  In this example, user TAPE1 can enter the TSO/E CONSOLE command only from the terminal specified by terminal_ID.

  Note: The TERMINAL class must also be active for this support to take effect.

- To refresh the TSOAUTH resource class using SETROPTS RACLIST, issue the following:

  ```
  SETROPTS RACLIST(TSOAUTH) REFRESH
  ```

To ensure that a user of an extended MCS console (TSO/E CONSOLE command) can activate the console, the MVS.MCSOPER.console_name profile in the RACF OPERCMDS class must grant the user READ access. For a TSO/E user, console-name is the TSO/E user ID that issues the TSO/E CONSOLE command. For an application program, console-name is the name specified on the MCSOPER macro.

You take the following steps to give users access to the RACF OPERCMDS class:

- Issue the SETROPTS command to activate the OPERCMDS class if it is not already activated for operator command processing:

  ```
  SETROPTS CLASSACT(OPERCMDS)
  ```

- Issue the SETROPTS command to activate generic profiles for the class:

  ```
  SETROPTS GENERIC(OPERCMDS)
  ```

- Issue RDEFINE to establish a profile for MVS.MCSOPER.*:

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

- Give the extended MCS console user access to the resource:

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

  Note: If you want RACF to control what console names a given user can use, you must create discrete MVS.MCSOPER.console_name profiles and give the user access only to those profiles.

- Issue SETROPTS RACLIST commands to refresh the OPERCMDS class:

  ```
  SETROPTS RACLIST CLASS(OPERCMDS) REFRESH
  SETROPTS GENERIC CLASS(OPERCMDS) REFRESH
  ```

See also *RACF V2 Security Administrator's Guide*.

Appendix K. Define, Redefine, and Delete CPF Entry

The following CMDCPF MPF command installation exit lets you define, redefine, and delete CPF entries.

Internet access to code

Internet access to source code found in GG24-4626 can be accessed via the Internet.

http://www.redbooks.ibm.com/redbooks

Download sample code from our redbooks.

or you can:

You can access the server by anonymous FTP:

```
ftp ftp.almaden.ibm.com
user: anonymous
password: your E-mail address
cd redbooks/SG244626
get cpf.code
```

```
        TITLE 'MPF COMMAND CPF - MANIPULATE COMMAND PREFIXES'
CMDCPF   CSECT
CMDCPF   AMODE 31                      31 BIT ADDRESSING MODE
CMDCPF   RMODE ANY                     31 BIT RESIDENCE
*
*
*    DESCRIPTIVE NAME - COMMAND PREFIX FACILITY SUPPORT COMMAND
*
*  FUNCTION   - VALIDATES A CPF DEFINE, DELETE, OR REDEFINE REQUEST
*               AND THEN INVOKES THE CPF SERVICE ROUTINE IEAVG710
*               TO PROCESS THE REQUEST.
*
*   Command Syntax:                            (Short form)
*
*   CPF DEFINE                                    N/A
*      ,PREFIX= prefix-address                    P=
*      ,OWNER=prefix-owner-address                O=
*     <,SCOPE= SYSPLEX | SYSTEM>                  S=
*     <,FAILDISP= PURGE | RETAIN | SYSPURGE>      F=
*     <,REMOVE= NO | YES>                         R=
*
*   CPF REDEFINE                                  N/A
*      ,PREFIX= prefix-address                    P=
*     <,OWNER= prefix-owner-address>              O=
*     <,CURSYS= system-name-address>              CS=
*     <,NEWSYS= system-name-address>              NS=
*
*   CPF DELETE                                    N/A
*     <,PREFIX= prefix-address>                   P=
*     <,CURSYS= system-name-address>              CS=
*
*-------------------------------------------------------------------
*-----------    Specification of a DEFINE request    ---------------
*-------------------------------------------------------------------
*
*   DEFINE -
*       This CPF function is used to establish the definition of
*       a new command prefix.
*
*       PREFIX -
*          This keyword specifies an 8-byte command prefix.
*          The prefix may be comprised of any character in the
*          range X'41' - X'FE', inclusive.  This is a required
*          keyword.
*
*       OWNER -
*          This keyword specifies an 8-byte name that
*          identifies the subsystem owning the command prefix
*          (e.g. JES2, JES3, IMS etc.).  The
*          name may be comprised of any character in the range
*          X'41'-X'FE', inclusive. This is a required keyword.
*
*       SCOPE -
*         The value of this keyword specifies the scope of the
*         prefix.  Either SYSTEM or SYSPLEX may be specified.
*         This is an optional keyword. The default is SYSPLEX.
*
*       FAILDISP-
*         The value of this keyword specifies the failure
*         disposition of the prefix. Either PURGE, RETAIN, or
*         SYSPURGE may be specified.  This is an optional keyword.
*         The default is PURGE.
*
*         A command prefix defined with a FAILDISP of PURGE will
*         be automatically deleted by Comm Task when the receiving
*         system is partitioned from the sysplex or the defining
*         address space terminates.
*
*         A command prefix defined with a FAILDISP of RETAIN will
*         persist despite sysplex partitioning or address space
*         termination. In this case it is the responsibility of
*         the owning subsystem to either delete the command prefix,
*         or redefine it for another system.
*
*         A command prefix defined with a FAILDISP of SYSPURGE will
*         be automatically deleted by Comm Task when the receiving
*         system is partitioned from the sysplex, but NOT when the
*         defining address space terminates.
*
*       REMOVE-
*         Specifies whether on not (YES or NO) the command prefix is
*         removed from the command text prior to be executed on the
*         receiving system.
*
*         NOTE: Command prefixes defined with a FAILDISP of PURGE
*               cannot be redefined.
```

```
*                                                                     R15      EQU   15
*--------------------------------------------------------------------* ------
*------------   Specification of a REDEFINE request  ----------------          BAKR  R14,0                    SAVE CALLER'S REGISTERS
*--------------------------------------------------------------------          LR    RBASE,R15                ESTABLISH MODULE BASE
*                                                                             USING CMDCPF,RBASE
*   REDEFINE -                                                                L     CMDXPTR,0(R1)            GET CMDX ADDRESS
*      This CPF function is used to alter the existing                        USING CMDX,CMDXPTR             ACCESS THE CMDX
*      definition of a command prefix.                                        L     BUFFPTR,CMDXCLIP         GET THE COMMAND BUFFER ADDRESS
*                                                                             USING CMDXCLIB,BUFFPTR         ACCESS THE BUFFER
*      PREFIX -                                                               LA    R2,CMDXCMDI              ACCESS START OF TEXT
*          This keyword specifies an 8-byte command prefix.                  SLR   R8,R8                    DITTO
*          The prefix may be comprised of any character in the                ICM   R8,3,CMDXCMDL            GET CMD LTH
*          range X'41' - X'FE', inclusive.  This is a required                BZ    RC0                      NOTHING - SPLIT
*          keyword.                                                           CH    R8,=H'3'                 MIN CPF LENGTH?
*                                                                             BNE   *+4+6+4                  NO - CHECK ALTERNATIVE 2
*      OWNER -                                                                CLC   0(3,R2),=CL4'CPF'        CPF COMMAND?
*          This keyword specifies an 8-byte name that                         BNE   RC0                      NO - EXIT
*          identifies the subsystem owning the command prefix                 CLC   =CL4'CPF',0(R2)          CPF COMMAND?
*          (e.g. JES2, JES3, IMS etc.).  The                                  BNE   RC0                      NO - NOTHING DO
*          name may be comprised of any character in the range       * ------ PROCESS CPF COMMAND
*          X'41'-X'FE', inclusive. This is an optional keyword.               LA    R0,DYNE-DYN
*                                                                             GETMAIN  RU,LV=(0),SP=230      OBTAIN DYNAMIC STORAGE
*      CURSYS -                                                               LR    R11,R1                   SAVE DYNAMIC AREA ADDRESS
*          This keyword specifies the name of the system for                  USING DYN,R11
*          which the prefix was defined. This is an optional                  XC    DYN(DYNWL-DYN),DYN       CLEAR
*          keyword. The default is the system name in the                     LA    R13,DYNSA                POINT AT SAVE AREA
*          CVTSNAME field of the CVT                                          MVC   DYNSA+4(4),=CL4'F1SA'    *CHAIN*
*                                                                   * ------ PARSE OPERANDS.....................
*      NEWSYS -                                                               LA    R7,3(,R2)                POINT PAST 'CPF'
*          This keyword specifies the the name of a new system                LA    R8,0(R8,R2)              POINT PAST COMMAND
*          to which commands with this prefix should be routed.               MVI   DYNSW,0
*          This is an optional keyword.  The default is the system  SCNOP    DS    0H
*          name in the CVTSNAME field of the CVT.                             CR    R7,R8                    ALL DONE?
*                                                                             BNL   DONOP                    YES - CHECK OUTCOME
*--------------------------------------------------------------------          CLI   0(R7),C' '               ANYTHING?
*-------------   Specifications for a DELETE request  ---------------          BNE   FNDOP                    YES - FIND WHAT
*--------------------------------------------------------------------          LA    R7,1(,R7)                POINT AT NEXT INPUT BYTE
*                                                                             B     SCNOP                    LOOP...
*                                                                   * ------ PARSE POSITIONALS..................
*   DELETE -                                                        FNDOP    DS    0H
*      This CPF function is used to delete the definition of                  CLC   =CL7'DEFINE,',0(R7)
*      a command prefix.                                                      BE    DEFRQ
*                                                                             CLC   =CL7'DELETE,',0(R7)
*      PREFIX -                                                               BE    DELRQ
*          This keyword specifies an 8-byte command prefix.                   CLC   =CL9'REDEFINE,',0(R7)
*          The prefix may be comprised of any character in the                BE    RDERQ
*          range X'41' - X'FE', inclusive.  This is a required                LA    R1,ERRMF                 ERROR QUALIFIER - FUNCTION
*          keyword.                                                           BAL   R3,EINFO                 ERROR MESSAGE
*                                                                             B     DONOP                    EXIT
*      CURSYS -                                                     * ------ PROCESS DEFINE REQUEST..............
*          This keyword specifies the name of the system for        DEFRQ    DS    0H
*          which the prefix was defined. This is an optional                  LA    R7,7(,R7)                POINT AT FIRST KW-OPERAND
*          keyword. The default is the system name in the           DEFKL    DS    0H
*          CVTSNAME field of the CVT                                          BAL   R3,FNDKW                 PARSE KW-OPERAND
*                                                                             B     *+4(R15)                 RC TEST..
*--------------------------------------------------------------------          B     DEFKOK                   .O.K.
*      Link-Edit Attr: RENT REUS REFR                                         B     DEFKIK                   .INVALID KW
*--------------------------------------------------------------------          B     DEFKIP                   .INVALID KW PARAMETER
*      Activation:                                                  * ------ CHECK KW.(R9 POINTS TO NEXT).........
*           Add the following definition to your installation       DEFKOK   DS    0H
*           MPFLSTxx member:                                                   CLC   =C'PREFIX=',DYNPK
*           CMD USEREXIT(CMDCPF)                                               BE    DEFPR
*           Note that you can have only one CMD statement in                   CLC   =C'P=',DYNPK
*           your MPFLSTxx concatenation.                                       BE    DEFPR
*--------------------------------------------------------------------          CLC   =C'OWNER=',DYNPK
R0       EQU   0                                                             BE    DEFOW
R1       EQU   1                                                             CLC   =C'O=',DYNPK
R2       EQU   2                                                             BE    DEFOW
R3       EQU   3                                                             CLC   =C'SCOPE=',DYNPK
R4       EQU   4                                                             BE    DEFSC
CMDXPTR  EQU   5                                                             CLC   =C'S=',DYNPK
R5       EQU   5                                                             BE    DEFSC
BUFFPTR  EQU   6                                                             CLC   =C'FAILDISP=',DYNPK
R6       EQU   6                                                             BE    DEFFA
R7       EQU   7                                                             CLC   =C'F=',DYNPK
R8       EQU   8                                                             BE    DEFFA
R9       EQU   9                                                             CLC   =C'REMOVE=',DYNPK
R10      EQU   10                                                            BE    DEFRE
R11      EQU   11                                                            CLC   =C'R=',DYNPK
RBASE    EQU   12                                                            BE    DEFRE
R12      EQU   12                                                            B     DEFKIK                   ERROR MESSAGE
R13      EQU   13                                                   DEFPR    DS    0H
R14      EQU   14                                                            TM    DYNSW,DYNSWP             HAVE A PREFIX ALREADY?
```

```
         BO    DEFKID                YES - ISSUE MESSAGE
         OI    DYNSW,DYNSWP          SET 'HAVE A PREFIX'
         MVC   DYNPR,DYNPO           SAVE PREFIX
         B     DEFCM                 TO CHECK MORE
DEFOW    DS    0H
         TM    DYNSW,DYNSWO          HAVE A OWNER ALREADY?
         BO    DEFKID                YES - ISSUE MESSAGE
         OI    DYNSW,DYNSWO          SET 'HAVE A OWNER'
         MVC   DYNOW,DYNPO           SAVE OWNER
         B     DEFCM                 TO CHECK MORE
DEFSC    DS    0H
         TM    DYNSW,DYNSWS          HAVE A SCOPE ALREADY?
         BO    DEFKID                YES - ISSUE MESSAGE
         OI    DYNSW,DYNSWS          SET 'HAVE A SCOPE'
         CLC   =C'SYSPLEX ',DYNPO    SYSPLEX?
         BNE   DEFSCA                NO
         MVI   DYNSC,C'X'            SET 'HAVE A SYSPLEX'
         B     DEFCM                 TO CHECK MORE
DEFSCA   DS    0H
         CLC   =C'SYSTEM ',DYNPO     SYSTEM?
         BNE   DEFKIP                NO - ERROR
         MVI   DYNSC,C'M'            SET 'HAVE A SYSTEM'
         B     DEFCM                 TO CHECK MORE
DEFFA    DS    0H
         TM    DYNSW,DYNSWF          HAVE A FAILDISP ALREADY?
         BO    DEFKID                YES - ISSUE MESSAGE
         OI    DYNSW,DYNSWF          SET 'HAVE A FAILDISP'
         CLC   =C'PURGE ',DYNPO      PURGE?
         BNE   DEFFAA                NO
         MVI   DYNFA,C'P'            SET 'HAVE A PURGE'
         B     DEFCM                 TO CHECK MORE
DEFFAA   DS    0H
         CLC   =C'RETAIN ',DYNPO     RETAIN?
         BNE   DEFFAB                NO
         MVI   DYNFA,C'R'            SET 'HAVE A RETAIN'
         B     DEFCM                 TO CHECK MORE
DEFFAB   DS    0H
         CLC   =C'SYSPURGE',DYNPO    SYSPURGE?
         BNE   DEFKIP                NO - MESSAGE
         MVI   DYNFA,C'S'            SET 'HAVE A SYSPURGE'
         B     DEFCM                 TO CHECK MORE
DEFRE    DS    0H
         TM    DYNSW,DYNSWR          HAVE A REMOVE ALREADY?
         BO    DEFKID                YES - ISSUE MESSAGE
         OI    DYNSW,DYNSWR          SET 'HAVE A REMOVE'
         CLC   =C'YES ',DYNPO        YES?
         BNE   DEFREA                NO
         MVI   DYNRE,C'Y'            SET 'HAVE A YES'
         B     DEFCM                 TO CHECK MORE
DEFREA   DS    0H
         CLC   =C'NO ',DYNPO         NO?
         BNE   DEFKIP                NO - MESSAGE
         MVI   DYNRE,C'N'            SET 'HAVE A NO'
         B     DEFCM                 TO CHECK MORE
DEFCM    DS    0H
         CLI   0(R9),C' '           ALL DONE?
         BE    DEFRQC                YES - CHECK REQUIRED PARMS
         LR    R7,R9                 POINT AT NEXT PARM
         CR    R7,R8                 MORE INPUT
         BL    DEFKL                 YES - LOOP...
         B     DEFRQC                CHECK REQUIRED PARMS
* ------ CHECK REQUIRED PARAMETERS...........
DEFRQC   DS    0H
         TM    DYNSW,DYNSWP+DYNSWO   REQUIRED PREFIX AND OWNER?
         BO    DEFRQA                PRESENT - CON'T
         LR    R7,R2                 POINT AT COMMAND
         B     DEFKMK                ISSUE ERROR MSG
* ------ PROVIDE DEFAULTS....................
DEFRQA   DS    0H
         TM    DYNSW,DYNSWR           HAVE A REMOVE ALREADY?
         BO    *+4+4                  YES - O.K.
         MVI   DYNRE,C'N'             SET 'HAVE A NO'
         TM    DYNSW,DYNSWF           HAVE A FAILDISP ALREADY?
         BO    *+4+4                  YES - O.K.
         MVI   DYNFA,C'P'             SET 'HAVE A PURGE'
         TM    DYNSW,DYNSWS           HAVE A SCOPE ALREADY?
         BO    *+4+4                  YES - O.K.
         MVI   DYNSC,C'X'             SET 'HAVE A SYSPLEX'
* ------ ISSUE CPF MACRO.....................
DEFCPF   DS    0H
         XC    DYNL(DYNLLN),DYNL
         MVC   DYNL01,=CL4'CPFP'
         OI    DYNL02,X'01'
         OI    DYNL03,X'80'           DEFINE
         CLI   DYNSC,C'M'             SCOPE SYSTEM?
         BE    SKIP470                YES
         OI    DYNL04,X'80'           SYSPLEX
SKIP470  DS    0H
         CLI   DYNFA,C'R'             FAILURE RETAIN?
         BNE   SKIP480                NO
         OI    DYNL04,X'60'           RETAIN
         B     SKIP490
SKIP480  DS    0H
         CLI   DYNFA,C'S'             FAILURE SYSPURGE?
         BNE   SKIP490                NO
         OI    DYNL04,X'20'           SYSPURGE
SKIP490  DS    0H
         CLI   DYNRE,C'Y'             REMOVE YES?
         BNE   SKIP520                NO
         OI    DYNL04,X'10'           YES
         B     SKIP520
SKIP520  DS    0H
         MVC   DYNL05,DYNPR           PREFIX
         MVC   DYNL06,DYNOW           OWNER
         LA    15,DYNL
         ST    15,DYNL09
         LINK  EP=IEAVG710,MF=(E,DYNL09)
         LTR   R15,R15                GOOD RC?
         BNZ   DEFRC                  NO - CHECK IT
         BAL   R3,OKMS                ISSUE MESSAGE
         B     DONOP                  EXIT
* ------ CHECK RC FROM CPF...................
DEFRC    DS    0H
         BAL   R3,CPFRC               ISSUE MESSAGE
         B     DONOP                  EXIT
* ------ ISSUE VARIOUS ERROR MESSAGES........
DEFKID   DS    0H
         LA    R1,ERRDU                ERROR QUALIFIER - DUPLICATE
         B     DEFKER
DEFKIK   DS    0H
         LA    R1,ERRIL                ERROR QUALIFIER - INVALID KW
         B     DEFKER
DEFKIP   DS    0H
         LA    R1,ERRIP                ERROR QUALIFIER - INVALID PARM
         B     DEFKER
DEFKMK   DS    0H
         LA    R1,ERRMK                ERROR QUALIFIER - MISSING KW
         B     DEFKER
DEFKER   DS    0H
         BAL   R3,EINFO                LINK TO ERROR MESSAGE
         B     DONOP                   EXIT
* ------ PROCESS DELETE REQUEST..............
DELRQ    DS    0H
         LA    R7,7(,R7)               POINT AT FIRST KW-OPERAND
DELKL    DS    0H
         BAL   R3,FNDKW                PARSE KW-OPERAND
         B     *+4(R15)                RC TEST..
         B     DELKOK                  .O.K.
         B     DEFKIK                  .INVALID KW
         B     DEFKIP                  .INVALID KW PARAMETER
* ------ CHECK KW.(R9 POINTS TO NEXT)........
DELKOK   DS    0H
         CLC   =C'PREFIX=',DYNPK
         BE    DELPR
         CLC   =C'P=',DYNPK
         BE    DELPR
         CLC   =C'CURSYS=',DYNPK
         BE    DELCS
         CLC   =C'CS=',DYNPK
         BE    DELCS
         B     DEFKIK                  ERROR MESSAGE
DELPR    DS    0H
         TM    DYNSW,DYNSWP            HAVE A PREFIX ALREADY?
         BO    DEFKID                  YES - ISSUE MESSAGE
         OI    DYNSW,DYNSWP            SET 'HAVE A PREFIX'
         MVC   DYNPR,DYNPO             SAVE PREFIX
         B     DELCM                   TO CHECK MORE
DELCS    DS    0H
         TM    DYNSW,DYNSWCS           HAVE A CURSYS ALREADY?
         BO    DEFKID                  YES - ISSUE MESSAGE
         OI    DYNSW,DYNSWCS           SET 'HAVE A CURSYS'
         MVC   DYNCS,DYNPO             SAVE CURSYS
         B     DELCM                   TO CHECK MORE
DELCM    DS    0H
         CLI   0(R9),C' '             ALL DONE?
         BE    DELRQC                  YES - CHECK REQUIRED PARMS
```

```
         LR    R7,R9                POINT AT NEXT PARM
         CR    R7,R8                MORE INPUT
         BL    DELKL                YES - LOOP...
         B     DELRQC               CHECK REQUIRED PARMS
* ------ CHECK REQUIRED PARAMETERS..........
DELRQC   DS    0H
         TM    DYNSW,DYNSWP         REQUIRED PREFIX?
         BO    DELCPF               PRESENT - CON'T
         LR    R7,R2                POINT AT COMMAND
         B     DEFKMK               ISSUE ERROR MSG
* ------ ISSUE CPF MACRO.....................
DELCPF   DS    0H
         XC    DYNL(DYNLLN),DYNL
         MVC   DYNLO1,=CL4'CPFP'
         OI    DYNLO2,X'01'
         OI    DYNLO3,X'20'         DELETE
         MVC   DYNLO5,DYNPR         PREFIX
         TM    DYNSW,DYNSWCS        CURSYS GIVEN?
         BZ    SKIP543              NO
         MVC   DYNLO7,DYNCS         CURSYS
SKIP543  DS    0H
         LA    15,DYNL
         ST    15,DYNLO9
         LINK  EP=IEAVG710,MF=(E,DYNLO9)
         LTR   R15,R15              GOOD RC?
         BNZ   DEFRC                NO - CHECK IT
         BAL   R3,OKMS              ISSUE MESSAGE
         B     DONOP                EXIT
* ------ PROCESS REDEFINE REQUEST............
RDERQ    DS    0H
         LA    R7,9(,R7)             POINT AT FIRST KW-OPERAND
RDEKL    DS    0H
         BAL   R3,FNDKW              PARSE KW-OPERAND
         B     *+4(R15)              RC TEST..
         B     RDEKOK                .O.K.
         B     DEFKIK                .INVALID KW
         B     DEFKIP                .INVALID KW PARAMETER
* ------ CHECK KW.(R9 POINTS TO NEXT)........
RDEKOK   DS    0H
         CLC   =C'PREFIX=',DYNPK
         BE    RDEPR
         CLC   =C'P=',DYNPK
         BE    RDEPR
         CLC   =C'OWNER=',DYNPK
         BE    RDEOW
         CLC   =C'O=',DYNPK
         BE    RDEOW
         CLC   =C'CURSYS=',DYNPK
         BE    RDECS
         CLC   =C'CS=',DYNPK
         BE    RDECS
         CLC   =C'NEWSYS=',DYNPK
         BE    RDENS
         CLC   =C'NS=',DYNPK
         BE    RDENS
         B     DEFKIK                ERROR MESSAGE
RDEPR    DS    0H
         TM    DYNSW,DYNSWP          HAVE A PREFIX ALREADY?
         BO    DEFKID                YES - ISSUE MESSAGE
         OI    DYNSW,DYNSWP          SET 'HAVE A PREFIX'
         MVC   DYNPR,DYNPO           SAVE PREFIX
         B     RDECM                 TO CHECK MORE
RDEOW    DS    0H
         TM    DYNSW,DYNSWO          HAVE A OWNER ALREADY?
         BO    DEFKID                YES - ISSUE MESSAGE
         OI    DYNSW,DYNSWO          SET 'HAVE A OWNER'
         MVC   DYNOW,DYNPO           SAVE OWNER
         B     RDECM                 TO CHECK MORE
RDECS    DS    0H
         TM    DYNSW,DYNSWCS         HAVE A CURSYS ALREADY?
         BO    DEFKID                YES - ISSUE MESSAGE
         OI    DYNSW,DYNSWCS         SET 'HAVE A CURSYS'
         MVC   DYNCS,DYNPO           SAVE CURSYS
         B     RDECM                 TO CHECK MORE
RDENS    DS    0H
         TM    DYNSW,DYNSWNS         HAVE A NEWSYS ALREADY?
         BO    DEFKID                YES - ISSUE MESSAGE
         OI    DYNSW,DYNSWNS         SET 'HAVE A NEWSYS'
         MVC   DYNNS,DYNPO           SAVE NEWSYS
         B     RDECM                 TO CHECK MORE
RDECM    DS    0H
         CLI   0(R9),C' '            ALL DONE?
         BE    RDERQC                YES - CHECK REQUIRED PARMS
```

```
         LR    R7,R9                POINT AT NEXT PARM
         CR    R7,R8                MORE INPUT
         BL    RDEKL                YES - LOOP...
         B     RDERQC               CHECK REQUIRED PARMS
* ------ CHECK REQUIRED PARAMETERS..........
RDERQC   DS    0H
         TM    DYNSW,DYNSWP+DYNSWO  REQUIRED PREFIX AND OWNER?
         BO    RDECPF               PRESENT - CON'T
         LR    R7,R2                POINT AT COMMAND
         B     DEFKMK               ISSUE ERROR MSG
* ------ ISSUE CPF MACRO.....................
RDECPF   DS    0H
         XC    DYNL(DYNLLN),DYNL
         MVC   DYNLO1,=CL4'CPFP'
         OI    DYNLO2,X'01'
         OI    DYNLO3,X'40'         REDEFINE
         MVC   DYNLO5,DYNPR         PREFIX
         MVC   DYNLO6,DYNOW         OWNER
         TM    DYNSW,DYNSWCS        CURSYS GIVEN?
         BZ    JUMP543              NO
         MVC   DYNLO7,DYNCS         CURSYS
JUMP543  DS    0H
         TM    DYNSW,DYNSWNS        NEWSYS GIVEN?
         BZ    JUMP550              NO
         MVC   DYNLO8,DYNNS         NEWSYS
JUMP550  DS    0H
         LA    15,DYNL
         ST    15,DYNLO9
         LINK  EP=IEAVG710,MF=(E,DYNLO9)
         LTR   R15,R15              GOOD RC?
         BNZ   DEFRC                NO - CHECK IT
         BAL   R3,OKMS              ISSUE MESSAGE
         B     DONOP                EXIT
* ------
DONOP    DS    0H
         CLI   DYNSW,0                ANY OPERANDS?
         BNE   FREEM                  YES - SKIP INFO
         BAL   R3,INFO                ISSUE INGO MESSAGE
* ------
FREEM    DS    0H
         LA    R0,DYNE-DYN
         FREEMAIN RU,LV=(0),A=(R11),SP=230 FREE THE STORAGE
         B     RC4                    DONE
* ------ RETURN ETC..........................
RC0      DS    0H
         XR    R15,R15                SYSTEM TO PROCESS COMMAND
         B     RET
RC4      DS    0H
         LA    R15,4                  EXIT PROCESSED THE COMMAND
         B     RET
RET      DS    0H
         PR                           RETURN TO CALLER
* ------ ISSUE MESSAGE.....................
OKMS     DS    0H
         LA    R6,MSG2                POINT AT MESSAGE
         B     INFOL
INFO     DS    0H
         LA    R6,MSG1                POINT AT MESSAGE
INFOL    DS    0H
         MVC   DYNWL(MWTOE-MWTO),MWTO  INIT WTO MF=L PLIST
         WTO   TEXT=(R6),MCSFLAG=(REPLY),CONSID=CMDXC4ID,MF=(E,DYNWL)
         AH    R6,0(,R6)              POINT AT...
         LA    R6,2(,R6)              .NEXT MESSAGE.
         CLI   0(R6),X'FF'            E.O.L?
         BER   R3                     YES - RETURN
         B     INFOL
* ------ ISSUE SYNTAX ERROR MESSAGE.........
EINFO    DS    0H
         OI    DYNSW,DYNSWER          INDICATE ERRORS DETECTED
         MVC   DYNWL(MWTOE-MWTO),MWTO  INIT WTO MF=L PLIST
         MVC   DYNWA(MSGEE-MSGE),MSGE  MESSAGE MODEL
         LA    R6,DYNWA               POINT AT MESSAGE
         LR    R15,R8                 BUILD..
         SR    R15,R7                 .ERROR..
         CH    R15,=AL2(L'MSGEI)      ..INFO..
         BNH   *+4+4                  ...MESSAGE.
         LH    R15,=AL2(L'MSGEI)
         BCTR  R15,0
         MVC   MSGEI-MSGE(0,R6),0(R7)
         EX    R15,*-6
         MVC   MSGER-MSGE(L'MSGER,R6),0(R1)
         WTO   TEXT=(R6),MCSFLAG=(REPLY),CONSID=CMDXC4ID,MF=(E,DYNWL)
```

```
         BR    R3                    RETURN
* ------ FIND KEYWORD.....................
FNDKW    DS    OH
         MVC   DYNPK,=CL16' '        CLEAR KW
         MVC   DYNPO,=CL16' '        CLEAR OPERAND
         LA    R15,4                 INVALID KEYWORD RC
         LR    R14,R7                DITTO
FNDKWL   DS    OH
         CR    R14,R8                END OF INPUT?
         BNLR  R3                    YES - RETURN
         CLI   0(R14),C'='           END OF KEY
         BE    FNDKWP                YES - FIND PARAMETER
         LA    R14,1(,R14)           POINT AT NEXT INPUT CHAR
         LR    RO,R14
         SR    RO,R7
         CH    RO,=H'8'              KW TOO LONG?
         BHR   R3                    YES - RETURN
         B     FNDKWL                LOOP...
FNDKWP   DS    OH
         LR    R1,R14
         SR    R1,R7
         BNPR  R3                    INVALID KW LENGTH
         MVC   DYNPK(0),0(R7)        SAVE..
         EX    R1,*-6                .KW
         LA    R15,8                 INVALID PARAMETER RC
         LA    R14,1(,R14)           PAST '='
         LR    R9,R14                POINT AT PARAMETER
FNDKWPL  DS    OH
         CR    R14,R8                END OF INPUT?
         BE    FNDKWPF               YES - PROCESS PARM
         CLI   0(R14),C' '           END OF PARM
         BE    FNDKWPF               YES - SAVE PARAMETER
         CLI   0(R14),C','           END OF PARM
         BE    FNDKWPF               YES - SAVE PARAMETER
         LA    R14,1(,R14)           POINT AT NEXT INPUT CHAR
         LR    RO,R14
         SR    RO,R9
         CH    RO,=H'8'              KW TOO LONG?
         BHR   R3                    YES - RETURN
         B     FNDKWPL               LOOP...
FNDKWPF  DS    OH
         LR    R1,R14
         SR    R1,R9
         BNPR  R3                    INVALID LENGTH
         BCTR  R1,0
         MVC   DYNPO(0),0(R9)        SAVE..
         EX    R1,*-6                .PARAMETER
         XR    R15,R15               GOOD RC
         LR    R9,R14                POINT PAST KW PARAMETER
         CLI   0(R14),C' '           LAST OPERAND DONE?
         BER   R3                    YES - RETURN
         LA    R9,1(,R14)            POINT AT NEXT KW PARAMETER
         BR    R3                    RETURN
* ------ RETURN/REASON CODE MESSAGE.....
CPFRC    DS    OH
         CH    R15,=H'12'            RC 12?
         BNE   *+4+2                 NO
         XR    RO,RO                 SET REASON = 0
         LR    R6,RO                 COPY REASON
         SRL   R6,2                  COMPUTE..
         BCTR  R6,0                  .REASON..
         MH    R6,=AL2(EL+2)         ..MESSAGE..
         AL    R6,CPFRCV-4(R15)      ...ADDRESS.
         MVC   DYNWL(MWTOE-MWTO),MWTO  INIT WTO MF=L PLIST
         WTO   TEXT=(R6),MCSFLAG=(REPLY),CONSID=CMDXC4ID,MF=(E,DYNWL)
         BR    R3                    RETURN
* -----
CPFRCV   DC    A(CPFDF4)             RETURN CODE
         DC    A(CPFDF8)
         DC    A(CPFDFC)
* -----
EL       EQU   46
CPFDF4 DS    OH                    REASON CODE
       DC AL2(EL),CL(EL)'PREFIX CONTAINED INVALID CHARACTERS    (4/4)'
       DC AL2(EL),CL(EL)'OWNER CONTAINED INVALID CHARACTERS     (4/8)'
       DC AL2(EL),CL(EL)'REDEFINE FOR A PREFIX FAILDISP=PURGE   (4/C)'
CPFDF8 DS    OH                    REASON CODE
       DC AL2(EL),CL(EL)'DELETE/REDEFINE - PREFIX NOT FOUND     (8/4)'
       DC AL2(EL),CL(EL)'DEFINE - PREFIX ALREADY EXISTS         (8/8)'
       DC AL2(EL),CL(EL)'DEFINE - PREFIX IS A SUBSET            (8/C)'
       DC AL2(EL),CL(EL)'DEFINE - PREFIX IS A SUPERSET         (8/10)'
       DC AL2(EL),CL(EL)'REDEFINE - INVALID NEWSYS             (8/14)'
       DC AL2(EL),CL(EL)'REDEFINE - PREFIX/SYSTEM ALREADY EXIST(8/18)'
       DC AL2(EL),CL(EL)'DELETE/REDEFINE - NO CPF TABLE EXISTS (8/1C)'
CPFDFC DS    OH                    REASON CODE
       DC AL2(EL),CL(EL)'ABEND/BROADCAST UPDATE FAILED          (C/-)'
* ------
         DROP  R11 ,                   DYN
         LTORG
* ------
MWTO     WTO   TEXT=0,MCSFLAG=(REPLY),MF=L
MWTOE    DS    OC
* ------
MSG1     DC AL2(48),CL48'UCPF01I CPF COMMAND SYNTAX                    '
         DC AL2(48),CL48'CPF DEFINE        | DELETE        | REDEFINE'
         DC AL2(48),CL48'   ,PREFIX=   (P=) ,PREFIX=        ,PREFIX= '
         DC AL2(48),CL48'   ,OWNER=    (O=)                 ,OWNER=  '
         DC AL2(48),CL48'  <,SCOPE=>   (S=)                            '
         DC AL2(48),CL48'  <,FAILDISP=>(F=)                            '
         DC AL2(48),CL48'  <,REMOVE=>  (R=)                            '
         DC AL2(48),CL48'                   <,CURSYS=> (CS=)<,CURSYS=>'
         DC AL2(48),CL48'                              (NS=)<,NEWSYS=>'
         DC AL2(48),CL48'      SCOPE=SYSPLEX | SYSTEM                  '
         DC AL2(48),CL48'      FAILDISP=PURGE | RETAIN | SYSPURGE      '
         DC AL2(48),CL48'      REMOVE=NO | YES                         '
         DC AL2(48),CL48'UCPF01I END OF DISPLAY                        '
         DC XL2'FFFF'
* ------
MSG2     DC AL2(28),CL28'UCPF02I CPF COMMAND ISSUED'
         DC XL2'FFFF'
* ------
MSGE     DC    AL2(MSGEE-MSGE)
         DC    C'UCPF03I - '
MSGER    DC    CL12' '
         DC    C' - '
MSGEI    DC    CL18' '
MSGEE    DS    OC
* ------
DEFSW    DC    AL1(DYNSWP+DYNSWO+DYNSWS+DYNSWF,DYNSWR) ALLOWED KWS
         DC    AL1(DYNSWP+DYNSWO)                      REQUIRED KWS
* ------
ERRSY    DC    CL(L'MSGER)'SYNTAX ERROR'
ERRDU    DC    CL(L'MSGER)'DUPLICATE KW'
ERRIL    DC    CL(L'MSGER)' ILLEGAL KW '
ERRIP    DC    CL(L'MSGER)'IVNALID PARM'
ERRMK    DC    CL(L'MSGER)' MISSING KW '
ERRMF    DC    CL(L'MSGER)'INVALID FUNC'
* ------
DYN      DSECT
DYNSA    DS    18F
DYNPR    DS    CL8
DYNOW    DS    CL8
DYNSC    DS    XL1
DYNFA    DS    XL1
DYNRE    DS    XL1
DYNCS    DS    CL8
DYNNS    DS    CL8
         CPF   MF=(L,DYNL)
DYNWL    WTO   TEXT=0,MCSFLAG=(REPLY),MF=L
DYNSW    DS    X
DYNSWP   EQU   X'80'              PREFIX
DYNSWO   EQU   X'40'              OWNER
DYNSWS   EQU   X'20'              SCOPE
DYNSWF   EQU   X'10'              FAILDISP
DYNSWR   EQU   X'08'              REMOVE
DYNSWCS  EQU   X'04'              CURSYS
DYNSWNS  EQU   X'02'              NEWSYS
DYNSWER  EQU   X'01'              ERRORS DETECTED
DYNPK    DS    CL9
DYNPO    DS    CL9
DYNWA    DS    CL128
DYNE     DS    OD
         PRINT NOGEN
         IEZVX101
         END   CMDCPF
```

Appendix L. Special Notices

This publication is intended to help customers and IBM technical personnel understand how to set up a multiple console support (MCS) environment in a MVS/ESA sysplex and the steps required to exploit and activate MCS features. The information in this publication is not intended as the specification of any programming interfaces that are provided by MVS/ESA Version 5 Release 2. See the PUBLICATIONS section of the IBM Programming Announcement for MVS/ESA JES2 and MVS/ESA JES3 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

Enterprise System/3090
ES/3090
ESA/390
InfoWindow
RACF
System/390
Enterprise System/9000
ES/9000
IBM
MVS/ESA
S/390
Virtual Machine/Enterprise Systems Architecture
VM/ESA

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communication company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Other trademarks are trademarks of their respective companies.

Appendix M. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

M.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How To Get ITSO Redbooks” on page 233.

Short Title	Title	Order Number
JES2 Implementation	*MVS/ESA SP-JES2 Version 5 Implementation Guide*	SG24-4583
JES3 Implementation	*MVS/ESA SP-JES3 Version 5 Implementation Guide*	SG24-4582
Version 5 Implementation Guide	*MVS/ESA Version 5 Implementation Guide*	SG24-4584
MVS 5.1 Presentation Guide	*MVS/ESA 5.1.0 Technical Presentation Guide*	GG24-4137
RACF V2.1 Inst. and Impl.	*RACF V2.1 for MVS Presentation Guide*	GG24-4281
DFSMS/MVS 1.2.0 Parallel Sysplex Support	*DFSMS/MVS 1.2.0 S/390 Parallel Sysplex Support*	GG24-4400
WLM Performance Studies	*System/390 MVS/ESA Version 5 Workload Manager Performance Studies*	SG24-4352
Parallel Sysplex Perf.	*System/390 MVS/ESA Version 5 Parallel Sysplex Performance*	GG24-4356
RACF V2.1 Inst. and Impl.	*RACF V2R1 Installation and Implementation Guide*	GG24-4405
HCD Primer	*MVS/ESA HCD and Dynamic I/O Reconfiguration Primer*	SG24-4037
Sysplex Migration Guide	*MVS/ESA Version 5 Sysplex Migration Guide*	SG24-4581

A complete list of International Technical Support Organization publications, known as redbooks, with a brief description of each, may be found in:

International Technical Support Organization Bibliography of Redbooks, GG24-3070.

M.2 MVS/ESA Publications

These publications are also relevant as further information sources.

A publication whose order number begins with the prefix **LY** is available to IBM-licensed customers only.

- Conversion

Short Title	Title	Order Number
MVS/ESA SP V5 Conversion Notebook	*MVS/ESA Conversion Notebook*	GC28-1436
MVS/ESA SP V5 JES2 Migration Notebook	*MVS/ESA JES2 Migration Notebook*	GC28-1437

Short Title	Title	Order Number
MVS/ESA SP V5 JES3 Conversion Notebook	*MVS/ESA JES3 Conversion Notebook*	GC28-1438

- Operations

Short Title	Title	Order Number
MVS/ESA SP V5 Planning: Operations	*MVS/ESA Planning: Operations*	GC28-1441
MVS/ESA SP V5 System Commands	*MVS/ESA System Commands*	GC28-1442
NetView Automation Planning	*NetView Automation Planning*	SC31-7082

- I/O Configuration Management

Short Title	Title	Order Number
MVS/ESA SP V5 HCD: Planning	*MVS/ESA Hardware Configuration Definition: Planning*	GC28-1445
HCD: User's Guide	*MVS/ESA Hardware Configuration Definition: User's Guide*	SC33-6468

- Multi-System Configuration Management

Short Title	Title	Order Number
Sysplex Overview	*System/390 MVS Sysplex Overview: An Introduction to Data Sharing and Parallelism*	GC28-1208
Sysplex Systems Management	*System/390 MVS Sysplex Systems Management*	GC28-1209
Sysplex Hardware and Software Migration	*System/390 MVS Sysplex Hardware and Software Migration*	GC28-1210
Sysplex Application Migration	*System/390 MVS Sysplex Application Migration*	GC28-1211
MVS/ESA SP V5 Setting Up a Sysplex	*MVS/ESA Setting Up a Sysplex*	GC28-1449
MVS/ESA SP V5 Sysplex Services Guide	*MVS/ESA Programming: Sysplex Services Guide*	GC28-1495
MVS/ESA SP V5 Sysplex Services Reference	*MVS/ESA Programming: Sysplex Services Reference*	GC28-1496
MVS/ESA SP V5 Planning: Global Resource Serialization	*MVS/ESA Planning: Global Resource Serialization*	GC28-1450
ESCON Manager Planning Guide	*Planning for the Enterprise Systems Connection Manager Version 1 Release 3*	GC23-0423
ESCON Director Planning	*Planning for the 9032 Enterprise Systems Connection Director*	GA23-0364
ESCON Migration Planning	*ESCON: Planning for Migration*	GG66-3181
Sysplex Timer Planning	*Planning for the 9037 Sysplex Timer*	GA23-0365

- Initialization and Tuning

Short Title	Title	Order Number
MVS Initialization and Tuning Guide	*MVS/ESA Initialization and Tuning Guide*	SC28-1451
MVS Initialization and Tuning Reference	*MVS/ESA Initialization and Tuning Reference*	SC28-1452

- Problem Determination and Recovery

Short Title	Title	Order Number
Recovery and Reconfiguration Guide	*MVS/ESA Recovery and Reconfiguration Guide*	GC28-1458

- Customization

Short Title	Title	Order Number
MVS/ESA SP V5 Installation Exits	*MVS/ESA Installation Exits*	SC28-1459
TSO Programming	*MVS TSO Programming*	GC28-1460

- Application Development

Short Title	Title	Order Number
Authorized Assembler Services Guide	*MVS/ESA Programming: Authorized Assembler Services Guide*	GC28-1467
Authorized Assembler Services Library	*MVS/ESA SP V5 Authorized Assembler Services Reference, Volumes 1-4*	GC28-1475, GC28-1476, GC28-1477, GC28-1478
MVS/ESA SP V5 Callable Services for HLL	*MVS/ESA Programming: Callable Services for High-Level Languages*	GC28-1464
MVS/ESA SP V5 Assembler Services Guide	*MVS/ESA Programming: Assembler Services Guide*	GC28-1466

- Messages and Codes

Short Title	Title	Order Number
MVS/ESA SP V5 System Messages	*MVS/ESA System Messages Volumes 1 - 5*	GC28-1480, GC28-1481, GC28-1482, GC28-1483, GC28-1484
Dump Output Messages	*MVS/ESA Dump Output Messages*	GC28-1485
MVS System Codes	*MVS/ESA System Codes*	GC28-1486
Routing and Descriptor Codes	*MVS/ESA Routing and Descriptor Codes*	GC28-1487

- Diagnosis: Tools and Services

Short Title	Title	Order Number
MVS/ESA SP V5 Diagnosis: Procedures	*MVS/ESA Diagnosis: Procedures*	LY28-1844
MVS/ESA SP V5 Diagnosis: Tools and Service Aids	*MVS/ESA Diagnosis: Tools and Service Aids*	LY28-1845
MVS/ESA SP V5 Diagnosis: Reference	*MVS/ESA Diagnosis: Reference*	LY28-1872
Data Areas	*MVS/ESA SP V5 Data Areas, Volumes 1-5*	LY28-1857, LY28-1858, LY28-1859, LY28-1860, LY28-1861
IPCS User's Guide	*MVS/ESA Interactive Problem Control System (IPCS) User's Guide*	GC28-1490
IPCS Commands	*MVS/ESA Interactive Problem Control System (IPCS) Commands*	GC28-1491

- RMF

Short Title	Title	Order Number
RMF GIM	*MVS/ESA Resource Measurement Facility Version 5 General Information*	GC33-6481
RMF User's Guide	*MVS/ESA Resource Measurement Facility Version 5 User's Guide*	GC33-6483
RMF Reports	*MVS/ESA Analyzing Resource Measurement Facility Version 5 Reports*	LY33-9178
RMF Messages and Codes	*MVS/ESA Resource Measurement Facility Version 5 Messages and Codes*	GC33-6484

- Systems Architecture and Hardware

Short Title	Title	Order Number
ESA/390 Principles of Operation	*Enterprise Systems Architecture/390 Principles of Operation*	SA22-7201
Common I/O Device Commands	*Enterprise Systems Architecture/390: Common I/O Device Commands*	SA22-7204
ESCON I/O Interface	*Enterprise systems Architecture/390: ESCON I/O Interface*	SA22-7202
Sysplex Timer Planning	*Planning for the 9037 Sysplex Timer*	GA23-0365

- Time Sharing Option Extensions (TSO/E)

Short Title	Title	Order Number
TSO/E V2 Command Reference	*TSO/E Version 2 Command Reference*	SC28-1881
TSO/E V2 Customization	*TSO/E Version 2 Customization*	SC28-1872

- Other Products

Short Title	Title	Order Number
RACF V2 Security Administrator's Guide	*RACF V2 Security Administrator's Guide*	SC23-3726
NetView Installation and Administration	*NetView Installation and Administration Guide (MVS)*	SC31-7084
NetView Administration Reference	*NetView Administration Reference*	SC31-7080
DFSMShsm Implementation and Customization Guide	*DFSMS/MVS DFSMShsm Implementation and Customization Guide*	SH21-1078
DFSMSdss Storage Administration Reference	*DFSMS/MVS DFSMSdss Storage Administration Reference*	SC26-4929
DFSMSdss Storage Administration Guide	*DFSMS/MVS DFSMSdss Storage Administration Guide*	SC26-4930
VTAM V4R2 Network Implementation Guide	*VTAM V4R2 Network Implementation Guide*	SC31-6494
VTAM V4R2 Programming	*VTAM V4R2 Programming*	SC31-6496
VTAM V4R2 Resource Definition Reference	*VTAM V4R2 Resource Definition Reference*	SC31-6498

M.3 OS/390 Publications

These publications are also relevant as further information sources.

A publication whose order number begins with the prefix **LY** is available to IBM-licensed customers only.

- I/O Configuration Management

Short Title	Title	Order Number
OS/390 HCD Planning	*OS/390 Hardware Configuration Definition Planning*	GC28-1750
OS/390 HCD User's Guide	*OS/390 HCD User's Guide*	SC28-1848

- RMF

Short Title	Title	Order Number
RMF Messages and Codes	*OS/390 RMF Messages and Codes*	GC28-1948
RMF Performance Management Guide	*OS/390 RMF Performance Management Guide*	SC28-1951
RMF User's Guide	*OS/390 RMF User's Guide*	SC28-1949
RMF Diagnosis Guide	*OS/390 RMF Diagnosis Guide*	LY28-1132
RMF Report Analysis	*OS/390 RMF Report Analysis*	SC28-1950
RMF Reference Summary	*OS/390 RMF Reference Summary*	SX22-0044
RMF Programmer's Guide	*OS/390 RMF Programmer's Guide*	SC28-1952
RMF Data Areas	*OS/390 RMF Data Areas*	LY28-1134

- Multi-System Configuration Management

Short Title	Title	Order Number
OS/390 Parallel Sysplex Overview	*OS/390 Parallel Sysplex Overview: An Introduction*	GC28-1860
OS/390 Parallel Sysplex Systems Management	*OS/390 Parallel Sysplex Systems Management*	GC28-1861
OS/390 Parallel Sysplex Hardware and Software Migr	*OS/390 Parallel Sysplex Hardware and Software Migr*	GC28-1862
OS/390 Parallel Sysplex Application Migration	*OS/390 Parallel Sysplex Application Migration*	GC28-1863
OS/390 V1R2.0 MVS Setting Up a Sysplex	*OS/390 MVS Setting Up a Sysplex*	GC28-1779
OS/390 V1R2.0 MVS Sysplex Services Guide	*OS/390 MVS Programming: Sysplex Services Guide*	GC28-1771
OS/390 V1R2.0 MVS Sysplex Services Reference	*OS/390 MVS Programming: Sysplex Services Reference*	GC28-1772

- Application Development

Short Title	Title	Order Number
OS/390 V1R2.0 MVS Auth Assembler Services Reference ALE-DYN	*OS/390 MVS Programming: Authorized Assembler Services Reference, Volume 1 (ALESERV-DYNALLO)*	GC28-1764
OS/390 V1R2.0 MVS Auth Assembler Services Reference ENF-ITT	*OS/390 MVS Programming: Authorized Assembler Services Reference, Volume 2 (ENFREQ-ITTFMTB)*	GC28-1765
OS/390 V1R2.0 MVS Auth Assembler Services Reference LLA-SDU	*OS/390 MVS Programming: Authorized Assembler Services Reference, Volume 3 (LLACOPY-SDUMPX)*	GC28-1766
OS/390 V1R2.0 MVS Auth Assembler Services Reference SET-WTO	*OS/390 MVS Programming: Authorized Assembler Services Reference, Volume 4 (SETFRR-WTOR)*	GC28-1767
OS/390 V1R2.0 MVS Auth Assembler Services Guide	*OS/390 MVS Programming: Authorized Assembler Services Guide*	GC28-1763
OS/390 V1R2.0 MVS Callable Services for HLL	*OS/390 MVS Programming: Callable Services for High-Level Languages*	GC28-1768

- Initialization and Tuning

Short Title	Title	Order Number
OS/390 V1R2.0 MVS Initialization and Tuning Guide	*OS/390 MVS Initialization and Tuning Guide*	SC28-1751
OS/390 V1R2.0 MVS Initialization and Tuning Reference	*OS/390 MVS Initialization and Tuning Reference*	SC28-1752

- Conversion and Exits

Short Title	Title	Order Number
OS/390 V1R2.0 MVS Installation Exits	*OS/390 MVS Installation Exits*	SC28-1753
OS/390 V1R2.0 MVS Conversion Notebook	*OS/390 MVS Conversion Notebook*	GC28-1747

- Commands and Operations

Short Title	Title	Order Number
OS/390 V1R2.0 MVS System Commands Summary	*OS/390 MVS System Commands Summary*	GX22-0040
OS/390 V1R2.0 MVS System Commands	*OS/390 MVS System Commands*	GC28-1781
OS/390 V1R2.0 MVS Planning: Operations	*OS/390 MVS Planning: Operations*	GC28-1760

- Messages and Codes

Short Title	Title	Order Number
OS/390 V1R2.0 MVS System Messages, Vol 1 (ABA-ASA)	*OS/390 MVS System Messages, Volume 1 (ABA-ASA)*	GC28-1784
OS/390 V1R2.0 MVS System Messages, Vol 2 (ASB-EWX)	*OS/390 MVS System Messages, Volume 2 (ASB-EWX)*	GC28-1785
OS/390 V1R2.0 MVS System Messages, Vol 3 (GDE-IEB)	*OS/390 MVS System Messages, Volume 3 (GDE-IEB)*	GC28-1786
OS/390 V1R2.0 MVS System Messages, Vol 4 (IEC-IFD)	*OS/390 MVS System Messages, Volume 4 (IEC-IFD)*	GC28-1787
OS/390 V1R2.0 MVS System Messages, Vol 5 (IGD-IZP)	*OS/390 MVS System Messages, Volume 5 (IGD-IZP)*	GC28-1788
OS/390 V1R2.0 MVS Dump Output Messages	*OS/390 MVS Dump Output Messages*	GC28-1749
OS/390 V1R2.0 MVS System Codes	*OS/390 MVS System Codes*	GC28-1780
OS/390 V1R2.0 MVS Routing and Descriptor Codes	*OS/390 MVS Routing and Descriptor Codes*	GC28-1778

- Diagnosis: Tools and Services

Short Title	Title	Order Number
OS/390 V1R2.0 MVS Diagnosis: Tools and Service Aids	*OS/390 MVS Diagnosis: Tools and Service Aids*	LY28-1845
OS/390 V1R2.0 MVS Diagnosis: Procedures	*OS/390 MVS Diagnosis: Procedures*	LY28-1844
OS/390 V1R2.0 MVS Diagnosis: Reference	*OS/390 MVS Diagnosis: Reference*	LY28-1872

- TSO/E

Short Title	Title	Order Number
TSO/E Administration	*OS/390 TSO/E Administration*	SC28-1966
TSO/E REXX Reference	*OS/390 TSO/E REXX Reference*	SC28-1975
TSO/E General Information	*OS/390 TSO/E General Information*	GC28-1964
TSO/E Customization	*OS/390 TSO/E Customization*	SC28-1965
TSO/E Programming Guide	*OS/390 TSO/E Programming Guide*	SC28-1970
TSO/E Programming Services	*OS/390 TSO/E Programming Services*	SC28-1971
TSO/E CLISTs	*OS/390 TSO/E CLISTs*	SC28-1973
TSO/E Guide to SRPI	*OS/390 TSO/E Guide to the Server-Requester Programming Interface*	SC28-1976
TSO/E User's Guide	*OS/390 TSO/E User's Guide*	SC28-1968
TSO/E REXX User's Guide	*OS/390 TSO/E REXX User's Guide*	SC28-1974
TSO/E System Programming Command Reference	*OS/390 TSO/E System Programming Command Reference*	SC28-1972
TSO/E Command Reference	*OS/390 TSO/E Command Reference*	SC28-1969
TSO/E Diagnosis: Data Areas	*OS/390 TSO/E System Diagnosis: Data Areas*	LY28-1138
TSO/E Messages	*OS/390 TSO/E Messages*	GC28-1978
TSO/E Primer	*OS/390 TSO/E Primer*	GC28-1967

How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL http://www.redbooks.ibm.com/redbooks.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet**

  Type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```
- **Redbooks Home Page on the World Wide Web**

  http://w3.itso.ibm.com/redbooks/redbooks.html
- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.
- **ITSO4USA category on INEWS**
- **Online** — send orders to:

  USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet)

  IBMMAIL — send orders to:

In United States:	usib6fpl at ibmmail
In Canada:	caibmbkz at ibmmail
Outside North America:	bookshop at dkibmbsh at ibmmail

  Internet — send orders to:

In United States:	usib6fpl@ibmmail.com
In Canada:	lmannix@vnet.ibm.com
Outside North America:	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29554	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada (toll free)	1-800-267-4455
Outside North America (long distance charge)	(+45) 48 14 2207

- **1-800-IBM-4FAX (United States)** or **(+1) 415 855 43 29 (Outside USA)**

  Ask for:
  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services**

  Send note to softwareshop@vnet.ibm.com

- **Redbooks Home Page on the World Wide Web**

  http://www.redbooks.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

- **Please put me on the mailing list for updated versions of the IBM Redbook Catalog.**

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number

- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

Glossary

This glossary defines terms used in this publication.

access. The ability to obtain the use of a protected resource.

access authority. An authority related to a request for a type of access to protected resources. In RACF, the access authorities are NONE, EXECUTE, READ, UPDATE, CONTROL, and ALTER.

access list. Synonym for standard access list. See also conditional access list.

accessor environment element (ACEE). A description of the current user, including user ID, current connect group, user attributes, and group authorities. An ACEE is constructed during user identification and verification.

ACEE. See accessor environment element.

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.

address space. The complete range of addresses available to a program. See also virtual address space.

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

AUDIT request. The issuing of the RACROUTE macro with REQUEST=AUDIT specified. An AUDIT request is a general-purpose security-audit request that can be used to audit a specified resource name and action.

AUTH request. The issuing of the RACROUTE macro with REQUEST=AUTH specified. The primary function of an AUTH request is to check a user′s authorization to a RACF-protected resource or function. The AUTH request replaces the RACHECK function. See also authorization checking.

authority. The right to access objects, resources, or functions. See access authority, class authority, and group authority.

authorization checking. The action of determining whether a user is permitted access to a protected resource. RACF performs authorization checking as a result of a RACROUTE REQUEST=AUTH or RACROUTE REQUEST=FASTAUTH.

automated operations. Automated procedures to replace or simplify actions of operators in both systems and network operations.

background. (1) In multiprogramming, the environment in which low-priority programs are executed. (2) Under TSO, the environment in which jobs submitted through the SUBMIT command or SYSIN are executed. One job step at a time is assigned to a region of central storage, and it remains in central storage to completion. Contrast with foreground.

background job. (1) A low-priority job, usually a batched or non-interactive job. (2) Under TSO, a job entered through the SUBMIT command or through SYSIN. Contrast with foreground job.

base segment. Synonym for RACF segment.

BAL. Basic assembler language

broadcast data set. Under TSO, a system data set containing messages and notices from the system operator, administrators, and other users. Its contents are displayed to each terminal user when he logs on the system, unless suppressed by the user.

CART. Command and response token.

central storage. (1) In System/390 virtual storage systems, the storage of a System/390 computing system from which the central processing unit can directly obtain instructions and data, and to which it can directly return results. (Formerly referred to as “real” storage″.) (2) Synonymous with processor storage.

CLPA. Common link pack area

class. A collection of RACF-defined entities (users, groups, and resources) with similar characteristics. The class names are USER, GROUP, DATASET, and the classes that are defined in the class descriptor table.

CNGRPxx. The SYS1.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.

configuration. The arrangement of a computer system or network as defined by the nature, number, and chief characteristics of its functional units.

console. That part of a computer used for communication between the operator or user and the computer.

console group. In MVS, a group of consoles defined in CNGRPxx, each of whose members can serve as an alternate console in console or hardcopy recovery or as a console to display synchronous messages.

CONSOLxx. The SYS1.PARMLIB member used to define message handling, command processing, and MCS 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.

cross-system coupling facility (XCF). XCF is a component of MVS that provides functions to support cooperation between authorized programs running within a sysplex.

CSA. Common service area

DASD. Direct access storage device.

deallocate. To release a resource that is assigned to a specific task.

dedicated. Pertaining to the assignment of a system resource - a device, a program, or a whole system - to an application or purpose.

default group. In RACF, the group specified in a user profile that is the default current connect group.

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 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

entry area. In MVS, the part of a console screen where operators can enter commands or command responses.

extended MCS console. In MVS, a console other than an MCS console from which operators or programs can issue MVS commands and receive messages. An extended MCS console is defined through an OPERPARM segment.

EBCDIC. Extended binary coded decimal interchange code

ECSA. Extended common service area

exit points. The place in the code where a routine (exit) receives control from the system.

extended binary coded decimal interchange code (EBCDIC). A set of 256 characters, each represented by 8 bits.

facility. (1) A feature of an operating system, designed to service a particular purpose, for example, the checkpoint/restart facility. (2) A measure of how easy it is to use a data processing system. Together with system performance, a major factor on which the total productivity of an installation depends. (3) Anything used or available for use in furnishing communication service. Commonly, a general term for communications paths.

foreground. (1) in multiprogramming, the environment in which high-priority programs are executed. (2) Under TSO, the environment in which programs are swapped in and out of central storage to allow CPU time to be shared among terminal users. All command processor programs execute in the foreground. Contrast with background.

foreground job. (1) A high-priority job, usually a real-time job. (2) A teleprocessing or graphic display job that has an indefinite running time during which communication is established with one or more users at local or remote terminals. (3) Under TSO, any job executing in a swapped region of central storage, such as a command processor or a terminal user's program. Contrast with background job.

full-capability console. An MCS console that can receive messages and send commands. See message-stream console and status-display console.

general resource. Any system resource, other than an MVS data set, that is defined in the RACF class descriptor table. General resources are DASD volumes, consoles, operator commands, tape volumes, load modules, terminals, IMS and CICS transactions, and installation-defined resource classes.

general-use programming interface (GUPI). An interface that IBM makes available for use in customer-written programs with few restrictions and that does not require knowledge of the detailed design or implementation of the IBM software product. See also product-sensitive programming interface (PSPI).

generic profile. A resource profile that can provide RACF protection for one or more resources. The resources protected by a generic profile have similar names and identical security requirements. For example, a generic data-set profile can protect one or more data sets.

group. A collection of RACF-defined users who can share access authorities for protected resources.

group name. A name that uniquely identifies a group of users to the system.

group profile. A profile that defines a group. The information in the profile includes the group name, profile owner, and users in the group.

GUPI. General-use programming interface.

hardcopy log. In systems with multiple console support or a graphic console, a permanent record of system activity.

hardware. Physical equipment, as opposed to the computer program or method of use; for example, mechanical, magnetic, electrical, or electronic devices. Contrast with software.

hardware configuration dialog. In MVS, a panel program that is part of the hardware configuration definition. The program allows an installation to define devices for MVS system configurations.

HCD. Hardware configuration dialog.

I/O. input/output

IBM-defined exit. The point in source code where IBM has added an exit point where an installation routine can receive control from the operating system. Contrast with installation-defined exit.

initial program load (IPL). The initialization procedure that causes an operating system to begin operation.

initialization parameter. An installation-specified parameter that controls the initialization and ultimate operation of MCS.

initialization statement. An installation-specified statement that controls the initialization and ultimate operation of MCS.

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.

interface. Hardware, software, or both, that links systems, programs, or devices.

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

JES3 complex. A multiple MVS system environment that allows JES3 subsystem consoles and MCS consoles with a logical association to JES3 to receive messages and send commands across systems.

keyword. A part of a command operand or SYS1.PARMLIB statement that consists of a specific character string (such as NAME= on the CONSOLE statement of CONSOLxx).

keyword parameter. A parameter that consists of a keyword, followed by one or more values. Contrast with positional parameter. See also parameter.

line number. A number associated with a line in a console screen display.

logoff. The procedure by which a user ends a console session.

logon. The procedure by which a user begins a console session.

master console. In an MVS system or sysplex, the main console used for communication between the operator and the system from which all MVS commands can be entered. The first active console with AUTH(MASTER) defined becomes the master console in a system or sysplex.

master console authority. In a system or sysplex, a console defined with AUTH(MASTER) other than the master console from which all MVS commands can be entered.

MCS. Multiple console support.

MCS console. A non-SNA device defined to MVS that is locally attached to an MVS system and is used to enter commands and receive messages.

message processing facility (MPF). A facility used to control message retention, suppression, and presentation.

message queue. A queue of messages that are waiting to be processed or waiting to be sent to a terminal.

message-stream console. An MCS console that 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 SYS1.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.

MVS. Multiple virtual storage.

NIP. Nucleus initialization program.

no-consoles condition. A condition in which the system is unable to access any full-capability console device.

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.

operand. (1) That which is operated upon. An operand is usually identified by an address part of an instruction. (2) Information entered with a command name to define the data on which a command processor operates and to control the execution of the command processor.

operations log. In MVS, the operations log is a central record of communications and system problems for each system in a sysplex.

operator commands. Statements that system operators may use to get information, alter operations, initiate new operations, or end operations.

operator message. A message from an operating system directing the operator to perform a specific function, such as mounting a tape reel; or informing the operator of specific conditions within the system, such as an error condition.

OPERLOG. The operations log.

OPERPARM. In MVS, a segment that contains information about console attributes for extended MCS consoles running on TSO/E.

OPERPARM segment. The portion of a RACF profile containing information relating to the user′s extended MCS console session.

out-of-line display area. For status-display and full-capability MCS consoles, areas of the screen set aside for formatted, multi-line display of status information written in response to certain MVS and subsystem commands.

owner. The user or group who creates a profile, or is named the owner of a profile. The owner can modify, list, or delete the profile.

parameter. (1) A variable that is given a constant value for a specific purpose or process. (2) See keyword parameter, positional parameter.

password. A unique string of characters that a program, computer operator, or user must supply to meet security requirements for gaining access to data.

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 SYS1.PARMLIB member that controls the PFK table settings for MCS consoles in a system.

positional parameter. A parameter that must appear in a specified location, relative to other parameters. Contrast with keyword parameter. See also parameter.

printer. A device that writes output data from a system on paper or other media.

product-sensitive programming interface (PSPI). A programming interface intended to be used only for specialized tasks such as: diagnosis, modification, monitoring, repairing, tailoring, and tuning of the IBM software product and that depends on or requires the customer to understand significant aspects of the design and implementation of the IBM software product. See also general-use programming interface (GUPI).

profile. Data that describes the significant characteristics of a user, a group of users, or one or more computer resources.

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.

PSPI. Product-sensitive programming interface.

program status word (PSW). A doubleword in main storage used to control the order in which instructions are executed, and to hold and indicate the status of the computing system in relation to a particular program.

pseudo-master console. A subsystem-allocatable console that has system command authority like that of an MCS master console.

PSW. Program status word.

queue. A line or list formed by items in a system waiting for processing.

RACF. Resource Access Control Facility. In this book RACF generally denotes any security product thet provides resource access control through the MVS security authorization facility (SAF) interface.

RACF database. A collection of interrelated or independent data items stored together without unnecessary redundancy, to serve Resource Access Control Facility (RACF).

RACF-protected. Pertaining to a resource that has either a discrete profile, an applicable generic profile, or a file or directory that doesn′t have a profile, but is protected with the File Security Packet (FSP). A data set that is RACF-protected by a discrete profile must also be RACF-indicated.

RACF segment. The portion of a RACF profile that contains the fundamental information about a user, group, or resource. This RACF segment contains information that is common to several applications. Also called base segment.

remote operations. Operation of remote sites from a host system.

roll mode. The MCS 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.

SAF. Security authorization facility

setup. The preparation of a computing system to perform a certain function.

SNA. Systems Network Architecture

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.

SQA. System queue area

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. A secondary or subordinate system, usually capable of operating independently of, or asynchronously with, a controlling system.

subsystem-allocatable console. A console managed by a subsystem like JES3 or NetView used to communicate with an MVS system.

SVC. Supervisor call instruction

symbol. (1) A representation of something by reason of relationship, association, or convention. (2) In MVS, a group of 1 to 8 characters, including alphanumeric characters and the three characters: #,

@, $. The symbol begins with either an alphabetic character or one of the three characters (#,@,$).

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. (1) A multiple-MVS system environment that allows MCS consoles or extended MCS consoles to receive messages and send commands across systems. (2) A set of MVS systems communicating and cooperating with each other through certain multisystem hardware components and software services to process customer workloads.

system console. In MVS, a console attached to the processor controller used to initialize an MVS system.

systems network architecture (SNA). The total description of the logical structure, formats, protocols, and operational sequences for transmitting information units through a communication 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.

Time Sharing Option Extensions (TSO/E). A licensed program that is based on the Time Sharing Option (TSO). It allows MVS users to interactively share computer time and resources.

TSO. Time-sharing option. See Time Sharing Option Extensions (TSO/E)

TSO/E. Time Sharing Option Extensions

UACC. Universal access authority.

undelivered message. An action message or WTOR that cannot be queued for delivery to the expected console. MVS delivers these messages to any console with the UD console attribute in a system or sysplex.

universal access authority (UACC). The default access authority that applies to a resource if the user or group is not specifically permitted access to the resource. The universal access authority can be any of the access authorities.

user identification and verification. The acts of identifying and verifying a RACF-defined user to the system during logon or batch job processing. RACF identifies the user by the user ID and verifies the user by the password or operator identification card supplied during logon processing or the password supplied on a batch JOB statement.

user identifier (user ID). (1) A unique string of characters that identifies an operator to the system. This string of characters limits the functions and information the operator can use. (2) The identification associated with a user or job.

user profile. A description of a RACF-defined user that includes the user ID, user name, default group name, password, profile owner, user attributes, and other information. A user profile can include information for subsystems such as MCS and TSO.

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 hard-copy 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.

XCF. Cross-system coupling facility.

Index

Special Characters

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Printed in U.S.A.