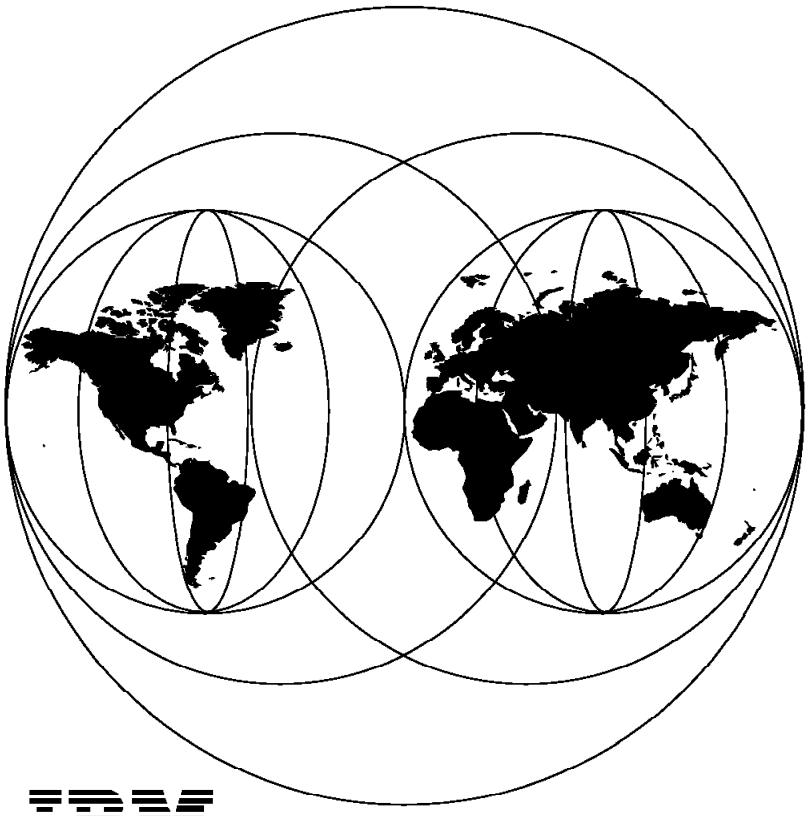
International Technical Support Organization

SG24-4582-00

MVS/ESA SP-JES3 Version 5
Implementation Guide
Release 5.1.1
Release 5.2.1

November 1995



IBM

International Technical Support Organization
Poughkeepsie Center

International Technical Support Organization

SG24-4582-00

MVS/ESA SP-JES3 Version 5
Implementation Guide
Release 5.1.1
Release 5.2.1

November 1995

Take Note!

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xv.

First Edition (November 1995)

This edition applies to MVS/ESA SP JES3 Version 5 Release 1.1, Program Number 5655-069 and MVS/ESA SP JES3 Version 5 Release 2.2, Program Number 5655-069, for use with the MVS/ESA SP Version 5 Release 2.

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

An ITSO Technical Bulletin Evaluation Form for reader′s feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 541 Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

Abstract

This document describes the new functions available with JES3 Version 5.1.1 and JES3 Version 5.2.1. It provides an overview of the changes made to JES3 in support of parallel coupled systems and gives guidelines through the migration process for JES3 Version 5.1.1 and JES3 Version 5.2.1.

JES3 operations support is significantly enhanced to exploit the MCS multisystem operations capabilities. JES3 managed operator consoles are not required any longer to manage a JES3 complex and therefore their support has been removed. JES3 unique operations features, such as functional message routing, are preserved in cases where there is no equivalent MCS support.

The following functional changes made to MVS and JES3 should be evaluated when migrating to JES3 5.2.1: JES3 sysplex operations, system symbols, automatic restart management (ARM), and MVS shared tape allocation.

This document was written for customers and IBM technical personnel working in support of MVS/ESA environments. Some knowledge of JES3 and the MVS sysplex environment is assumed.

LS (185 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help customers and IBM technical personnel understand the new functions provided by JES3 5.1.1 and JES3 5.2.1 and the migration steps required to install either of these products. The information in this publication is not intended as the specification of any programming interfaces that are provided by MVS/ESA SP Version 5 Release 1 or MVS/ESA SP Version 5 Release 2. See the PUBLICATIONS section of the IBM Programming Announcement for 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 information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. 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.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

DFSMS
Hardware Configuration Definition
IMS/ESA
OpenEdition
S/390
System/390
ESCON
IBM
MVS/ESA
PR/SM
Sysplex Timer

The following terms are trademarks of other companies:

Network File System, NFS OSF/DCE .

Windows is a trademark of Microsoft Corporation.

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.

C-bus is a trademark of Corollary, Inc.

Other trademarks are trademarks of their respective companies.

Preface

This document provides an overview of the JES3 Version 5 product. It contains a description of the functional changes made to MVS/ESA JES3 Version 5 Release 1.1 and MVS/ESA JES3 Version 5 Release 2.1, and guides installations through the migration process for converting to either of these two releases.

This document is intended for system programmers and IBM technical personnel involved in the installation of JES3.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “MVS/ESA JES3 Version 5” describes MVS/ESA JES3 Version 5 in a parallel sysplex and data sharing environment. It also provides information on JES3 changes in JES3 Version 5.1.1 and in its operating environment.
- Chapter 2, “JES3 Version 5 and JES Common Coupling Services” provides information on JES common coupling services (JESXCF) services and gives an overview of JES3 and the JES common coupling services interactions.
- Chapter 3, “JES3 Enhancements for Sysplex” describes the JES3 requirements for setting up a sysplex.
- Chapter 4, “JES3 Version 5.1.1 Migration” summarizes the MVS/ESA JES3 Version 5 Release 1.1 migration actions.
- Chapter 5, “MCS Sysplex Operations” describes MCS sysplex concepts and their interactions with JES3.
- Chapter 6, “JES3 5.2.1 Sysplex Operations” describes the new JES3 function and usage considerations associated with the enablement of the MCS sysplex operations environment.
- Chapter 7, “New Function Considerations for JES3 5.2.1” describes the new JES3 function and usage considerations for System Symbols, Automatic Restart Manager (ARM), and MVS Shared Tape Allocation.
- Chapter 8, “JES3 5.2.1 Migration” summarizes the MVS/ESA JES3 Version 5 Release 2.1 migration actions.
- Appendix A, “Sample Program Using JESXCF Services” describes a sample JES3 DSP that uses JESXCF services to monitor JES3 local systems.
- Appendix B, “SYS1.PARMLIB Definitions for a Sysplex” shows sample SYS1.PARMLIB members for a sysplex environment.
- Appendix C, “Sample ISPF Dialog to Interface Operlog” includes a sample program which allows the MVS Operations Log (OPERLOG) to be viewed from an ISPF browse session.
- Appendix D, “Sample ARM Driver Program” includes sample programs to enable use of Automatic Restart Manager (ARM) services without changing the application program code.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- JES3 Publications for Version 5 Release 2.1.

Short Title	Title	Order Number
MVS/ESA SP V5 JES3 Commands	*MVS/ESA JES3 Commands*	GC28-1444
MVS/ESA SP V5 JES3 Conversion Notebook	*MVS/ESA JES3 Conversion Notebook*	GC28-1438
MVS/ESA SP V5 JES3 Initialization and Tuning Guide	*MVS/ESA JES3 Initialization and Tuning Guide*	SC28-1455
MVS/ESA SP V5 JES3 Initialization and Tuning Reference	*MVS/ESA JES3 Initialization and Tuning Reference*	SC28-1456
MVS/ESA SP V5 JES3 Messages	*MVS/ESA JES3 Messages*	GC28-1489
MVS/ESA SP V5 JES Common Coupling Services	*MVS/ESA Programming: JES Common Coupling Services*	GC28-1497

The following publications are available to IBM-licensed customers only.

Short Title	Title	Order Number
MVS/ESA SP V5 JES3 Customization	*MVS/ESA JES3 Customization*	LY28-1853
MVS/ESA SP V5 JES3 Diagnosis	*MVS/ESA JES3 Diagnosis*	LY28-1855

- Other Publications for MVS/ESA SP Version 5.2.
- Conversion

Short Title	Title	Order Number
MVS/ESA SP V5 Conversion Notebook	*MVS/ESA Conversion Notebook*	GC28-1436

- 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

- 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

- 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

- 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

International Technical Support Organization Publications

The following redbooks contain information about the Version 5 sysplex environment. Books not already available will be published in 1996.

Short Title	Title	Order Number
MVS/ESA Implementation Guide	*MVS/ESA Version 4 Implementation Guide*	GG24-3628
JES Version 5 Overview	*MVS/ESA JES2 and JES3 Technical Announcement Presentation*	GG24-3330
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	*Workload Manager Performance Studies*	SG24-4352
WLM CICS Migration	*Migration to MVS V5 Workload Management in a CICS/VSAM Environment*	SG24-4353
Parallel Sysplex Perf.	*S/390 Parallel Sysplex Performance*	GG24-4356
RACF V2.1 Inst. and Impl.	*RACF V2R1 Installation and Implementation Guide*	GG24-4405
HCD and Dynamic I/O Reconfiguration Primer	*MVS/ESA HCD and Dynamic I/O Reconfiguration Primer*	GG24-4037
Sysplex Migration Guide	*MVS/ESA Version 5 Sysplex Migration Guide*	SG24-4581

International Technical Support Organization Publications

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

International Technical Support Organization Bibliography of Redbooks, GG24-3070.

To get a catalog of ITSO technical publications (known as “redbooks”), VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Redbooks

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their local IBM office.

Customers may order hardcopy ITSO books individually or in customized sets, called BOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain redbooks on a variety of products.

Acknowledgments

The advisor of the first edition of this document was:

Paul Rogers International Technical Support Organization, Poughkeepsie Center

The authors of the first edition of this document were:

Juha Vainkainen IBM Finland

Paul Rogers IBM ITSO

This publication is the result of a residency conducted at the International Technical Support Organization, Poughkeepsie Center.

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Andrew Schmidt S/390 Software Design Center

David Share S/390 Software Design Center

Tom Petrolino S/390 Software Design Center

Chapter 1. MVS/ESA JES3 Version 5

This chapter describes MVS/ESA JES3 Version 5 in a parallel sysplex and data sharing environment. It also provides information on JES3 changes in JES3 5.1.1 and JES3 5.2.1 in their operating environment.

MVS/ESA SP Version 5 Release 1 (MVS/ESA SP 5.1) and MVS/ESA SP Version 5 Release 2 (MVS/ESA 5.2) improve the cost effectiveness of the S/390 platform by addressing key areas of software and systems management costs. In addition, MVS/ESA SP 5.1 provides the initial support for the S/390 parallel sysplex.

The MVS/ESA sysplex was announced in September 1990 as a platform for an evolving large system computing environment. MVS/ESA SP 5.1 enhances the sysplex into a new parallel sysplex, which is a large system computing environment offering that improves price/performance through cost effective processor technology and enhanced software.

A sysplex is a collection of MVS/ESA systems that cooperate, using certain hardware and software products, to process workloads. A major difference between a sysplex and a conventional large computer system is the improved growth potential and the level of availability in a sysplex. The sysplex increases the number of processing units and MVS operating systems that cooperate, which in turn can increase the amount of work that can be processed. To facilitate this cooperation, new products have been created and old products have been enhanced for the MVS/ESA SP 5.1 operating environment.

1.1 MVS/ESA SP 5.1.0 Enhancements

The S/390 parallel sysplex is a fundamentally new structure that enables parallel processing and data sharing across MVS systems. MVS/ESA SP Version 5 Release 1 enhancements include:

- Improved availability through parallel processing
- Reduced system management complexity
- The capability to manage applications and transactions according to a service level agreed to by the business
- Enhancements to OpenEdition services that include:
  - Open interfaces integrated into the MVS Base Control Program
  - The benefits of an open distributed system structure through the incorporation of OSF/DCE components into MVS as a new interface called OpenEdition DCE Base Services
  - A REXX command interface to OpenEdition services
  - DFSMS/MVS Network File System feature access to the Hierarchical File System (HFS)
  - Internationalization support for worldwide languages
  - Integrated sockets support
  - Improved application debugging capability for applications that use threads

1.1.1 Parallel Sysplex Support

The parallel sysplex provides high performance data sharing through a coupling facility. The coupling facility gives high performance multisystem data sharing capability to authorized applications, such as MVS subsystems. Use of a coupling facility by software subsystems, such as IMS/ESA Database Manager (IMS DB), ensures integrity and consistency of data throughout the sysplex.

Multisystem data sharing makes the sysplex ideal for parallel processing, particularly for online transaction processing. Figure 1 shows the hardware parts of a sysplex.

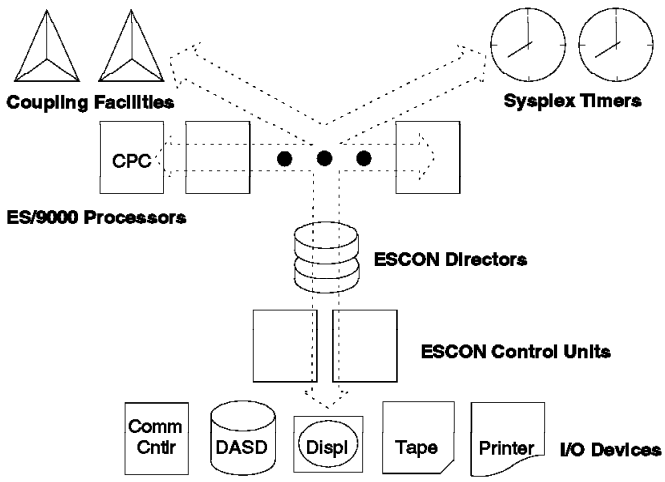


Figure 1. Parallel Sysplex

Coupling facility The coupling facility enables high performance multisystem data sharing.

Coupling facility links Coupling facility links provide high-speed connectivity between the coupling facility and the central processor complexes (CPCs) that use them.

Sysplex Timer The sysplex timer synchronizes the time-of-day (TOD) clocks in multiple CPCs in a sysplex.

System/390 Processors Selected models of S/390 processors can take advantage of a sysplex. These models include large water-cooled processors, air-cooled processors, and microprocessor clusters.

ESCON directors Enterprise systems connection (ESCON) channels enhance data access and communication in the sysplex. The ESCON Directors add dynamic switching capability for ESCON channels.

ESCON control units ESCON control units and I/O devices in a sysplex provide the increased connectivity necessary among a greater number of systems.

Since the introduction of the sysplex, IBM has developed technologies that enhance sysplex capabilities. These sysplex enhancements provide high performance data sharing through a new coupling technology. A coupling facility gives high performance multisystem data sharing capability to authorized applications. MVS subsystem′s, such as Information Management System (IMS), exploitation of the coupling facility ensures the integrity and consistency of data throughout the entire sysplex.

In short, a parallel sysplex builds on the base sysplex capability, and it increases the number of central processor complexes (CPCs) and MVS images that can directly share work. The coupling facility enables high performance, multisystem data sharing across the increased number of systems. In addition, workloads can be dynamically balanced across systems with the help of the MVS workload manager function.

The MVS/ESA SP 5.1 parallel sysplex may include up to 32 systems and improves communication and data sharing among those systems. Both MVS/ESA JES3 Version 5 and MVS/ESA JES2 Version 5 exploit the sysplex capabilities through JES common coupling services.

1.1.2 JES Common Coupling Services

JES common coupling services (JESXCF) is a new MVS component available with Version 5. JESXCF is an XCF multisystem application and provides, based on XCF coupling services, common inter-processor and intra-processor communication services for both JES3 and JES2 subsystems. The JESXCF services are tailored to meet JES specific requirements and are provided through macros that enable communication among JES members in a sysplex. The macros are available to either JES3 or JES2 and can be used *only in JES environments*, except for the send message service that can be used in all environments. JES3 uses the JESXCF services to:

- Provide a mechanism for JES dispatchable units (JDU - a generic term for JES3 DSPs and FCTs or JES2 PCEs) running in a JES complex to exchange data and to communicate using the MVS cross-system coupling facility (XCF) services
- Enable JDUs to request notification about XCF-monitored system events

JES2 and JES3 connect to the JESXCF address space during initialization. After the JES has successfully attached to a JESXCF group, JES routines running on that JES can use the JES common coupling services to communicate with the other JESs in the same JESXCF group. Figure 2 on page 4 shows the relationship between JES, JESXCF, and XCF.

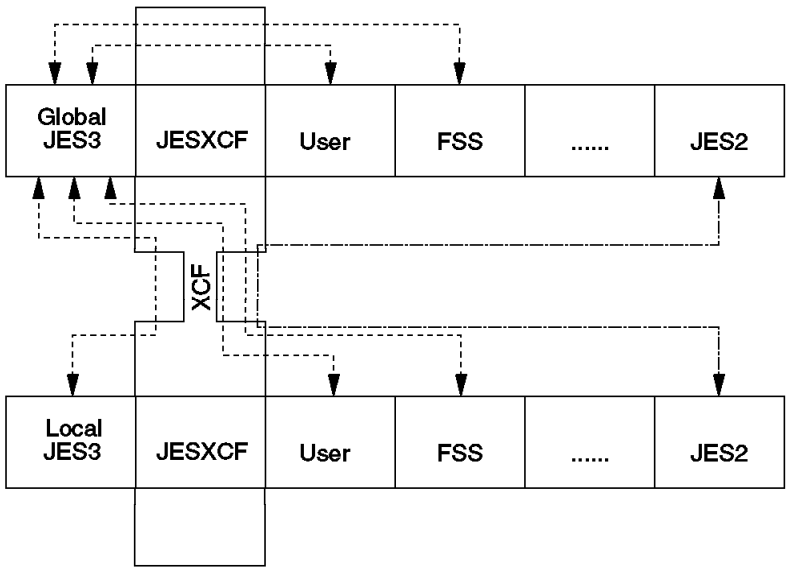


Figure 2. JESXCF Exploitation

Note: A sysplex configuration is required for a JES3 Version 5 complex or a JES2 Version 5 MAS. If the sysplex consists of only one MVS system, XCF can be configured in local mode (IEASYSxx PLEXCFG=XCFLOCAL), which does not require a couple data set. However, activation of other MVS/ESA SP 5.1 functions, for example workload management (WLM), may require a single-system sysplex configuration with a couple data set (IEASYSxx PLEXCFG=MONOPLEX).

1.2 JES3 5.1.1 Enhancements Overview

The following enhancements have been made to JES3 in Version 5 Release 1.1:

- JES3 XCF exploitation
  - See 2.1, “XCF Concepts” on page 17.
- JES3 common coupling macro services and exits
  - See 2.2, “JESXCF Concepts” on page 21.
- JES3 main device scheduler restart
  - See 3.3, “JES3 5.1.1 Enhancements for Sysplex” on page 38.
- Support for sysplex and 32 processors in a complex
  - See 3.5, “JES3 Version 5 and Sysplex” on page 39.
- Removal of the CPUID from initialization
  - See 4.1.1, “CPUID Removal from Initialization” on page 53.
- 4-digit device number support
  - See 4.1.4, “4-Digit Device Support” on page 54.
- Support for modifying the DEVICE initialization statement
  - See 4.12, “Updating DEVICE Initialization Statements” on page 62.

1.2.1 JES3 XCF Exploitation

JES3 Version 5 uses the MVS cross-system coupling facility (XCF) for inter-processor communication. This eliminates the dedicated JES3-managed CTCs. Because the JES3-managed CTCs are eliminated, all systems in a JES3 complex must be migrated to JES3 Version 5.1.1 level at the same time.

Replacing the JES3 CTCs with XCF communications allows installations to reduce their hardware investment to run a JES3 complex by sharing hardware resources with other components that also use XCF signalling. The use of ESCON CTCs allows improved monitoring of the hardware and greater distance between processors in a sysplex. XCF can use multiple concurrent communication paths between processors, thus reducing the potential for single failures disrupting operations. XCF also lets communication paths be added, changed, or deleted dynamically between processors. When the XCF communication hardware is changed, a JES3 complex-wide re-IPL and JES3 warmstart is avoided because the JES3 initialization stream does not have to be changed.

1.2.2 Removal of the CPUID from Initialization

The requirement to define CPUIDs in the JES3 initialization stream to JES3 Version 5.1.1 has been removed. The CPUID, which was specified on the MAINPROC initialization statement, identified to JES3 what system was being IPLed. Each time the CPUID was changed on a processor, the MAINPROC statement had to be updated resulting in a complex-wide re-IPL and JES3 warmstart. This was particularly difficult for installations running PR/SM. All potential CPUID numbers had to be included in the initialization stream to avoid having to warmstart JES3 with each CPUID change.

Because the CPUID and the MODEL parameters are removed from the MAINPROC statement, the JES3 system name is required to match the MVS system name. That is, the processor name specified on the MAINPROC statement must be the same name specified on the SYSNAME parameter in the IEASYSxx parmlib member.

1.2.3 JES3 32-Way Support

JES3 Version 5 has been enhanced in order to support from one to 32 systems in a sysplex. This allows up to 32 members to be connected in a JES3 sysplex. A sysplex can be made up of multiple JES3 global complexes.

S/390 parallel processing uses data sharing technology to allow up to 32 systems to concurrently access shared data. JES3 Version 5.1.1 is also enhanced to allow 32 mains to exist in a JES3 complex implemented as a sysplex. The expanded complex can include a variety of processors, high-end to low-end, as long as there are processors of sufficient processing power to assume the role of the global, including the event of a dynamic system interchange (DSI).

Note: DFSMS/MVS Version 1 Release 2 allows up to eight system or sysplex names to be defined to a storage management subsystem (SMS) complex configuration. This alleviates the eight-system limitation. Although an SMS configuration can still only contain eight names, those names can be a combination of both system names and system group names. A system group is all systems that are part of the same sysplex and are running the SMS, minus any systems in the sysplex that are explicitly defined in the SMS configuration.

JES3 does not provide data set awareness and scheduling services for the SMS data sets when the DFSMS/MVS Version 1.2 configuration is defined using sysplex names (as opposed to system names). If the SMS configuration is defined using a combination of both system names and system group names, JES3 SMS data sets services are available on those systems whose name matches the system names defined in the SMS configuration. This also means that JES3 SMS data set services are available at most on seven systems in a SMS complex with more than eight MVS systems.

DFSMS/MVS 1.3 provides support for 32 system and/or system group names, allowing JES3 SMS data set services to be available on all systems in a 32 main sysplex.

1.2.4 Modifying the DEVICE Initialization Statement

When a JES3 complex is expanded to include new JES3 processors (up to 32 mains), the task of updating the initialization stream to define all devices to each main becomes more complex.

To alleviate the work involved in modifying the DEVICE initialization statement, two ISPF macros are provided in the SYS1.SAMPLIB. The IATDEVA macro allows the system programmer to add new DEVICE statements (a single device, a list of devices, or a range of devices by specifying a “model” DEVICE statement). The IATDEVC macro allows the programmer to add or delete a main processor or group of main processors automatically to a group of devices.

1.2.5 4-Digit Device Number Support

MVS/ESA SP 5.1 allows four hexadecimal digits to be used as a device numbers. This theoretically increases the maximum number of devices that can be defined to a single MVS image from 4,096 to 65,536. The MVS control blocks (UCBs) representing the I/O devices have been restructured and some parts have been moved above 16MB. This allows approximately 1500 *additional* I/O devices to be defined in to the same virtual storage area below 16MB, that was used in the previous releases of MVS.

Four digit device numbers make it easier for installations to share the same I/O definition files (IODFs) on multiple systems and also provide more flexibility for assigning unique device numbers across a sysplex.

JES3 Version 5.1.1 also allows both 3-digit and 4-digit device number specifications on commands, initialization statements, and write-to-operator responses. On commands and statements where the use of device numbers and device types could be ambiguous, a preceding slash (/) must be used to indicate a device number.

Messages and fields in control blocks that define a device number accommodate 4-digit device numbers in JES3 Version 5.1.1.

1.2.6 JES Common Coupling Macro Services and Exits

The JES common coupling services provide a set of macros that enable communication among *JES members* of a sysplex. The macros are available in both JES3 and JES2 environments. JES3 uses the JESXCF services as a replacement for its CTC communication among JES3 main processors. In addition to the macros, the JESXCF services also provide exits that can be used to monitor or tailor messages sent among the JESes.

The JES common coupling services include:

- Attaching and detaching a JESXCF group that can be used independently from the JES provided common coupling service communications.
- Sending, receiving and acknowledging the reception of messages (for example, data sets, files, commands).
- Using exits to modify JES system messages. Specifically, JESXCF services provides three exit points: transport exit IXZXIT01, receive exit IXZXIT02, and JESXCF ATTACH/DETACH exit IXZXIT03.

Note: The IBM recommendation is to use the JESXCF installation exits, standard JES3 exits, and JES3 DSPs when JES3 system customization is required, although JESXCF services can be used outside standard JES3 exits or DSPs.

1.2.7 JES3 Main Device Scheduler Restart

JES3 main device scheduler (MDS) restart is improved by adding multiple secondary SETUP FCTs. The main processor connect processing does not begin until after the MDS restart processing has completed. In prior JES3 releases, the MDS restart processing runs under a single FCT. JES3 Version 5.1.1 creates multiple secondary FCTs during the MDS restart (for the MDSREST DSP located in IATMDRS). These FCTs reallocate the MDS resources (volumes, data sets, and devices) to the jobs that had them allocated and determine which main processors are eligible to continue processing the jobs.

1.3 MVS/ESA SP 5.2.0 Enhancements

The S/390 parallel sysplex is a fundamentally new structure that enables parallel processing and data sharing across MVS systems. The MVS/ESA SP Version 5 Release 2 enhancements are described in the following sections.

1.3.1 System Symbols

MVS/ESA SP 5.2 enhances the ability to share (replicate) resource definitions in a multisystem environment. You can use the same commands, dynamic allocations, parmlib members, and job control language for started tasks in all systems in the sysplex by using system symbols to represent unique values; the system replicates the definitions so all systems can use them. If you allow all systems to share resource definitions in a replicated environment, you can view the multisystem environment as a single system image.

1.3.2 Master JCL in PARMLIB

This support allows you to specify, in SYS1.PARMLIB, an alternate version of the master scheduler job control language (commonly called master JCL).

Previously, you could specify alternate master JCL only in the MSTJCLxx load module in SYS1.LINKLIB. Each time you made a change to the master JCL, you had to reassemble MSTJCLxx and link-edit the load module again. When you specify the master JCL in the MSTJCLxx parmlib member, it is easier to make changes because the assemble and link-edit are not required.

1.3.3 Message Suppression Indicators

Improved message suppression indicators (commonly known as IMSI characters) allow you to suppress any combination of the following system initialization messages:

- Informational messages
- Prompt for master catalog response
- Prompt for system parameters response

The advantage to using the new IMSI characters is that they allow greater flexibility when suppressing system initialization messages than in previous releases of MVS.

1.3.4 XES Dynamic Reconfiguration

The structure alter function allows XES to dynamically change the size of a coupling facility structure (either expand or contract the structure) and to reapportion the use of storage in the structure between entries and data elements without disrupting its use by structure connectors. This function, which enhances the continuous availability of coupling facility structure data, permits installations to continue running their applications while changes in the structure are being made. Changes to the structure might be required for such reasons as growth in the application data, variable use of the structure during different business periods, or growth in the number of processors or applications.

The dynamic reconfiguration of XES is available only to structures allocated in a coupling facility with CFLEVEL=1.

The MVS IXLLIST services provide increased function when used with a structure allocated in a coupling facility with CFLEVEL=1. These functions enable improved cooperative processing of data in a list structure by the set of connectors to that structure.

1.3.5 Dynamic SSI

You can define and manage subsystems without requiring an IPL. Previously, subsystems could be defined only at IPL through the IEFSSNxx parmlib member. The dynamic SSI support includes a set of authorized system services that subsystems can invoke to:

- Define and add a subsystem
- Activate a subsystem
- Deactivate a subsystem
- Swap subsystem functions
- Store and retrieve subsystem-specific information
- Define subsystem options
- Query subsystem information

The dynamic SSI support includes also the SETSSI and DISPLAY SSI operator commands.

All of the features of the dynamic SSI support last only for the life of the IPL.

1.3.6 MVS System Logger

MVS/ESA SP Version 5 Release 2 extensions to the parallel sysplex bring value to single-system as well as multisystem configurations. The MVS system logger provides an integrated single system image MVS logging facility for exploitation by system and subsystem components. The system logger exploits coupling facility technology. The system logger supports a real-time log-merge capability, so that records being written by multiple instances of a subsystem on multiple systems appear in logical order in a single merged log stream. In addition, the system logger provides for the “hardening” of log records on a failure-independent, persistent medium. The system logger is initially exploited by the operations log component of MVS (optional replacement for SYSLOG) and logrec error recording.

1.3.7 MVS Operations Log

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.

The OPERLOG is operationally independent of SYSLOG; that is, an installation can choose to run with either or both. A failure of the OPERLOG when the OPERLOG is active and SYSLOG (or the hardcopy message log) is not, causes SYSLOG to be activated automatically in an effort to reduce the impact to the operations and problem determination tasks. The SYSLOG records are written in the MVS format.

1.3.8 Automatic Restart Management

The purpose of automatic restart management (ARM) is to provide fast efficient restarts for critical applications when they fail. ARM is an MVS recovery function that improves the time required to restart an application by automatically restarting the batch job or started task (STC) when it unexpectedly terminates. These unexpected outages may be the result of an abend, system failure, or the removal of a system from the sysplex.

Automatic restart management is a function introduced in MVS/ESA SP 5.2.0. It runs in the cross-system coupling facility (XCF) address space and maintains its own dataspaces. ARM requires a primary couple data set to contain policy information as well as status information for registered elements. It supports both JES2 and JES3 environments.

ARM is a function of the XCF component and operates on sysplex scope in both JES2 and JES3 environments as shown in Figure 3.

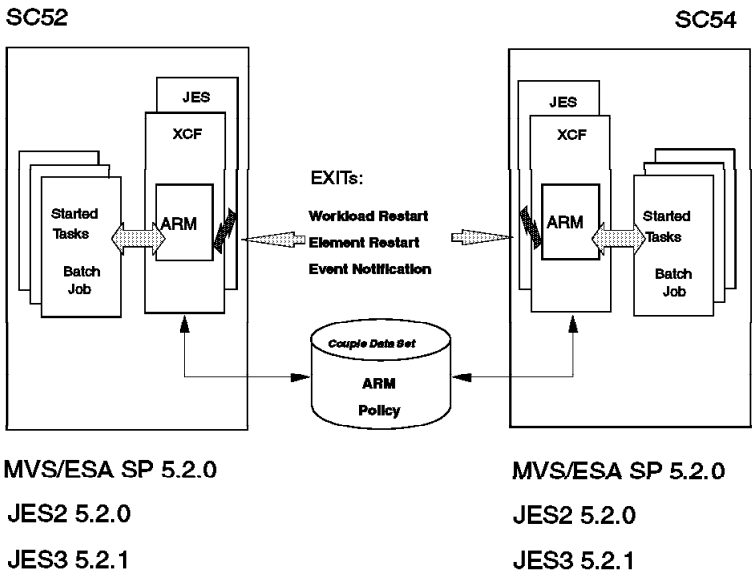


Figure 3. Automatic Restart Management Environment

1.3.9 Shared Tape Support

MVS/ESA SP 5.2 also includes tape allocation enhancements:

- The selection algorithm has changed the way it selects tape devices to allocate. It considers the type of request, unit information on the request, and the characteristics of each available tape device. Based on these factors, it chooses the optimal device to allocate. With MVS/ESA SP 5.2, an installation might notice a difference in which devices the system selects for a particular job.
- Also new for MVS/ESA SP 5.2 is the automatic tape switching. This allows you to have a category of tape devices known as automatically switchable. These devices are not dedicated to any one system; rather they can be used by all systems in a sysplex that are connected through a coupling facility. Automatic tape switching improves the management of tape devices. You can define eligible devices as automatically switchable. The system that receives a tape device allocation request assigns an available automatically switchable device without requiring that the device be varied online for the allocating system or taken offline from another system.

  Automatic tape switching helps you control costs by reducing both the number of dedicated tape devices required in the sysplex and the amount of operator intervention required to manage the tape devices.

  Tape devices defined as automatically switchable cannot be simultaneously JES3 managed.

1.3.10 UCB Services

With MVS/ESA SP 5.2, you can specify that a UCB exist above 16MB to conserve common virtual storage while maintaining compatibility with existing interfaces, such as access methods and services. This support enables you to increase the number of devices in your I/O configuration beyond the limits that existed before MVS/ESA SP 5.2. You can take advantage of 4-digit device numbers without the limit caused by common storage constraints below 16MB.

For each device definition, the system creates a UCB to contain device information. Before MVS/ESA SP 5.2, the fact that many programs have dependencies on 24-bit UCB addresses required that the UCBs exist in common storage below 16MB. With this support, a UCB can exist above 16MB because the system can create a 24-bit view into that UCB. When you allocate a device, the system creates the view, known as a captured UCB, in the private storage area of the address space.

If you are not using standard IBM services, for example, you are building your own data extent block (DEB), you might need to explicitly control capturing of an above 16MB UCB. To do this, you use the IOSCAPU macro to explicitly capture and release UCBs. This support uses the MVS/ESA SP 5.2 shared pages function. With the shared pages function, which is available through the IARVSERV macro, virtual storage areas can be defined in address spaces or data spaces and can be shared by programs. This sharing reduces the amount of processor storage required and the I/O necessary to support data applications that require access to the same data. For example, IARVSERV provides a way for a program running below 16MB, in 24-bit addressing mode, to access data above 16MB that it shares with 31-bit addressing mode programs.

1.4 JES3 5.2.1 Enhancements Overview

The following enhancements have been made to JES3 in Version 5 Release 2.1:

- JES3 sysplex operations
- Optional use of the MVS operations log and system logger
- Support for automatic restart management (ARM)
- Support for system symbolic variables
- Support for UCB virtual storage constraint relief
- Coexistence with shared tape allocation

1.4.1 JES3 Sysplex Operations

- JES3 console support is merged with MVS console support and improves systems management and automated operations by enabling the sysplex operations features of MVS/ESA.

  Work to create a single multi-system console manager for MVS began in 1979. A major milestone in this evolution was MVS 2.2.0 and JES3 2.2.1, which allowed JES3 commands to be entered from MCS consoles and JES3 responses to be routed back to the MCS consoles. JES3 still provided a single-system image for operations by doing command and message transportation between the systems in the JES3 complex. MVS/ESA 4.1.0 introduced an MCS multisystem console manager (MCS sysplex) that provides console consolidation in a sysplex. A console can be physically

attached to one system in the sysplex and control all systems in the sysplex, thus providing single-system image operations. MCS in MVS/ESA 4.1.0 also introduced a set of general use programming interfaces to allow programs to activate an MCS console, issue commands, and receive messages. These extended MCS consoles have the same capabilities as DIDOCS consoles including sysplex-wide scope. MCS sysplex uses XCF services for command and message transportation between systems in a sysplex.

Until JES3 5.2.1, JES3 continued to control message transportation, even in a JES3 complex contained in a sysplex. This caused a conflict as both MCS sysplex and JES3 tried to transport the console traffic. The problem is solved by *not* activating the MCS sysplex when JES3 is installed and letting JES3 be responsible for the transportation of console traffic between systems. The disadvantage of that solution is, that many of the enhanced features of the MCS sysplex are lost.

Multiple console support (MCS) sysplex is enabled when running JES3 5.2.1. Therefore, you have sysplex-wide control from any console on any system for the following consoles:

- MCS
- NetView (and similar products)
- System console
- Extended MCS consoles
- TSO (CONSOLE command)

The complete console environment and the various components that make up this environment are shown in Figure 4.

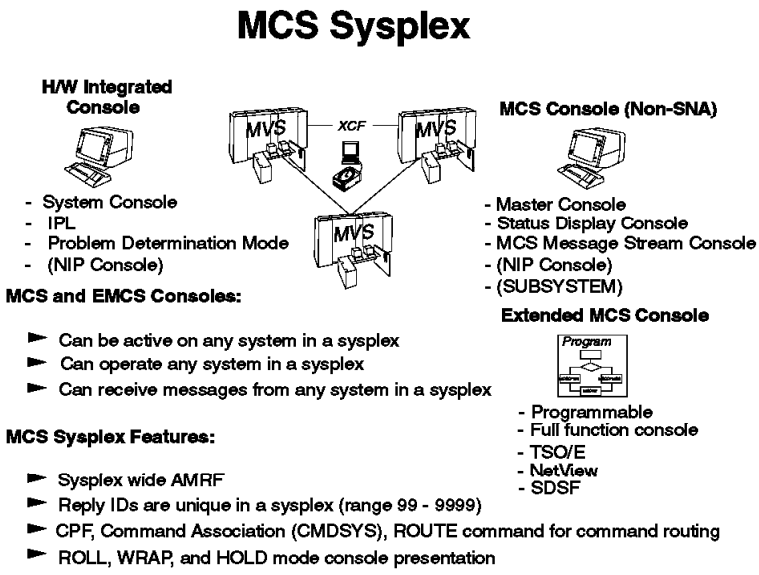


Figure 4. MCS Sysplex Environment

The transport of messages and commands between systems in a sysplex is controlled by MCS:

  - For command routing within the sysplex, the MVS ROUTE command replaces the JES3 *SEND command.
    - The *SEND command continues to support NJE command routing.
  - The command prefix facility (CPF) is used to register JES3 command synonyms.
- JES3 *does not* control the flow of messages and commands between systems in the JES3 complex.
- Message processing facility (MPF) processing, including MPF exits, occurs only on the originating system:
  - JES3 provides an option to pass all message traffic *received* on the JES3 global through MPF processing on the global.
- MVS subsystems see all messages on the SSI on all systems eligible to display the message.
- JES3 command processors support 4-byte console IDs and preserve the command and response tokens (CART) which appropriately identify command responses, enabling improved automated operations.

JES3 5.2.1 improves systems management and automated operations by enabling the sysplex operations features of MVS/ESA. System and subsystems can use these features in all JES environments without special considerations to the alternative console implementations in the previous JES3 environments. Sysplex-wide operation control is available from consoles on any system rather than a subset of the consoles specified to JES3 on the global processor.

- MVS and JES3 hardcopy log may now be maintained using the operations log (OPERLOG). The MVS operations log provides sysplex-wide services using the MVS system logger.
- Optionally a JES3 maintained DLOG may be used for logging.
- JES3 RJP and NJE console support is redesigned to exploit extended MCS consoles.
- Deleted JES3 functions:
  - Support for JES3 managed operator and MLOG (JES3 log format written to a hardcopy media or a JES3 managed display console) consoles is removed, as they are no longer required to manage the sysplex.
  - JES3 managed action message retention facility (JMRF) - totally replaced by MCS AMRF.
  - Support for entry of JES3 commands from a BDT session, along with the corresponding command authorization exit IATUX56.

1.4.2 JES3 Automatic Restart Management Support

Automatic restart management is a policy driven MVS restart manager providing improved availability for exploiting subsystems and applications. ARM uses *event driven* failure recognition such as, “end of memory” and “sysgone” to trigger restart processing for affected registered failing applications. ARM does not perform the initial starts of the applications (in either the initial IPL or subsequent IPLs).

In order to become eligible for ARM restart processing, batch jobs and started tasks use ARM services (IXCARM REGISTER) during their initialization (even

during restart initialization) to request ARM restart for any of the following conditions:

- Unexpected termination - restart attempt on the same system on which it had been running when the termination occurred.
- System termination or removal from the sysplex - ARM refers to the active policy and invokes an installation exit (if one exists) to determine the allowable systems for restarting.

  An allowable system must belong to the same JES3 complex under which the terminated work was originally started. JES XCF services (IXZXIXIF) are used to extract the eligible system names.

The registration request identifies, among other things, the ARM policy definitions for the subject work. The ARM policy indicates how ARM is to process work after a restart condition occurs. The ARM policy parameters specify how to restart work, the conditions when to restart, and optionally the JCL or the command that is to be used for restarting the work. A policy may also describe related work that has to be restarted after a failure.

ARM-registered work that is terminated by a MVS CANCEL or FORCE command is not restarted by ARM unless the ARMRESTART parameter is specified. ARM-registered work that is terminated by a JES3 *F J=jobno,C|CP|CO (batch jobs only) command is not restarted by ARM unless the ARMR parameter is specified.

APPC transactions, TSO users, system address spaces, initiators, and batch jobs in a DJC net are not eligible for ARM controlled restarts.

When work initially registers, ARM determines whether a batch job or a started task is doing the registration. During the ARM registration process, the JES that was responsible for a batch job's JCL processing gets notified that the work is ARM registered. The ARM registration with JES requests JES to hold the job after it ends so that ARM can restart it if appropriate. The JES ARM register function returns a job token that uniquely identifies the job.

If ARM subsequently restarts a batch job, it uses the original JCL (also referred to as persistent JCL) unless the installation policy or an exit routine provides an overriding definition. If overriding JCL is used to restart an ARM registered batch job, ARM dynamically allocates an internal reader to submit the restart JCL. For this, MVS allocation provides a new dynamic allocation text unit that tells JES that jobs submitted through the internal reader being allocated must run on the system on which it was submitted (overriding any system affinities).

ARM saves the original start command when the registering work is a started task. This is referred to as persistent command text. At restart time, ARM uses the saved start command to initiate the restart unless the installation or an exit has provided an override method. ARM restarts a started task by internally issuing an operator command (usually, but not necessarily, a START command).

JES3 has an existing operator command "*RESTART,main,jobname or jobno" to restart a job that is already in execution. The *RESTART,main command causes the failure option for the job to be taken. The *RESTART command is also accepted for jobs that are registered with ARM. When a job terminates, depending on the FAILURE= value, JES3 either puts the job into operator hold, or notifies ARM and indicates that the job is held awaiting ARM restart. When ARM is notified, ARM attempts to restart registered work and queries JES3 as to

whether the job is restartable. For example, batch jobs in a DJC net are not ARM restartable. Depending on the JES3 response, ARM may restart the job (the *RESTART command is ignored) or it may not restart the job (the *RESTART command processed normally).

1.4.3 UCB Virtual Storage Constraint Relief

The MVS/ESA 5.2 UCB virtual storage constraint relief allows UCBs to be defined to reside above 16MB to conserve common virtual storage while maintaining compatibility with existing standard services. MVS/ESA SP 5.2 supports the following:

- Allows 4-digit device numbers to be fully exploited.
- Provides relief for the below 16MB virtual storage problem by allowing the UCB to be defined in 31-bit storage above 16MB.
- Compatibility is maintained with old 24-bit UCB manipulation interfaces. This is implemented through the use of the MVS captured storage function, which allows you to reference a 31-bit UCB through a temporary 24-bit window.
- JES3 is internally changed to use the captured UCBs where appropriate and to use UCB macro services to obtain device related information.

This UCB VSCR support serves the two-fold purpose of removing device definition constraints and providing relief for constrained below 16MB private areas. The majority of this support is contained in the MVS Input/Output Supervisor (IOS) and MVS allocation (ALLOC) components; however, there is a large number of other components and products that are impacted in lesser, but possibly more widespread, ways.

The movement of the UCB prefix above 16MB occurred in MVS/ESA 5.1. This provided sufficient VSCR to allow an additional 1300 DASD UCBs beyond the old 4096 limit to be defined without using any more 24-bit virtual storage. The UCB VSCR support provides additional VSCR by allowing the UCB common section (UCBOB) with stub, the UCB device dependent section, the UCB common extension, and the device dependent extension to be moved above 16MB at the option of the installation if the UIM allows it. To maintain compatibility with existing interfaces that have a dependency on 3-byte UCB pointers, the UCB VSCR support allows the UCB common section plus stub, the device dependent section, device class extension (DCE) and the common extension to be captured. A captured UCB can be thought of as a way to reference the 31-bit UCB through a temporary 24-bit window (to be shared between address spaces without being in AR-mode and without replicating data, and to be viewed using 24-bit addresses).

Traditionally, UCBs have been accessible through data management control blocks such as the data extent block (DEB), allocation control blocks such as the task input/output table (TIOT) and from the communications vector table (CVT). A number of these control blocks use a 3-byte field for the UCB address. MVS allocation ensures that existing programs continue to work by using captured UCB addresses in the TIOT fields.

All JES3 spool access methods use the JES3 spool I/O driver for spool I/O. The JES3 spool I/O driver is an MVS IOS driver and provides an interface between the spool access methods and the MVS IOS. The spool I/O driver initiates spool I/O through the STARTIO service and processes I/O termination through JES3 exit routines that are invoked by IOS. Because the spool I/O driver is invoked from both the JES3 address space and user address spaces, the actual (4-byte)

spool device UCB addresses (stored in the JES3 spool I/O parameter block) are used in the STARTIO requests. JES3 initialization translates (IOSCAPU service) the captured UCB address, obtained from spool extent′s DEB, to the actual UCB address.

The JES3 JESEXCP service initiates tape, unit record, or BSC RJP device I/O using the MVS EXCP IOS driver. The EXCP IOS driver supports only 24-bit UCB addresses. Normally MVS allocation returns a captured UCB address (through TIOT), but JES3 does not use MVS allocation for the JESEXCP devices. If the JESEXCP target UCB address in the support units table has a 4-byte address, JESOPEN service captures the UCB. The captured UCB is then used in the EXCP requests. Finally, JESCLOSE service releases the storage for the captured UCB.

1.4.4 JES3 and System Symbols

This support allows for the replication of installation definitions by allowing the use of system symbolic variables in the source JCL for started tasks, TSO logon procedures, MVS commands, and dynamic allocations.

In JES3, JCL conversion can occur on a different system than where the demand select job is to be executed. Therefore, in order for the converting system to have the correct symbol table, JES3 picks up the correct symbol table and passes it with the JCL to the converting system.

&SYSCLONE and up to 100 installation-defined symbolic variables can be defined in addition to &SYSPLEX and &SYSNAME. The symbolic variables are defined at IPL time. Any of these symbolic variables can be used wherever &SYSNAME and &SYSPLEX may be used in parmlib, in JCL, in dynamic allocation, and on operator commands.

1.4.5 JES3 Coexistence with MVS Shared Tape Allocation

MVS/ESA 5.2 allocation has changed the non-JES3 managed tape devices selection algorithms as follows:

- MVS/ESA 5.2 allocation is enhanced to automatically assign/unassign a shared, non-JES3 managed, automatically switchable tape device for a requestor in a parallel sysplex. The automatically switchable tape devices are not dedicated to any one system; however, they are accessible only to one system at any given point of time. A coupling facility is used to maintain the sysplex-wide allocation status of the automatically switchable tape devices.
- The “VARY dddd,AUTOSWITCH,ON or OFF” command provides the ability to dynamically change the AUTOSWITCH attribute of a tape device. The MVS VARY ONLINE command, when it is issued for an automatically switchable device, brings the device online on the issuing system but does not assign it.
- When a JES3 managed tape device is brought online to JES3, it is also automatically varied online to MVS. If the device is assignable, JES3 uses multi-system assign that allows the device to be accessible from all the systems in the JES3 complex.

Chapter 2. JES3 Version 5 and JES Common Coupling Services

This chapter provides information on JES common coupling services (JESXCF), XCF services, XES services, and gives an overview of the JES3 and the JES common coupling services interactions. These components provide the basis for a parallel sysplex environment. Figure 5 shows a multisystem JES3 environment and the relationship of the sysplex services provided by JESXCF, XCF, and XES. JES3 uses JESXCF services, which calls XCF to provide communication between the members of a MAS. The communication path is determined by XCF. XCF drives the CTC paths and calls XES to use the coupling facility as the communication path.

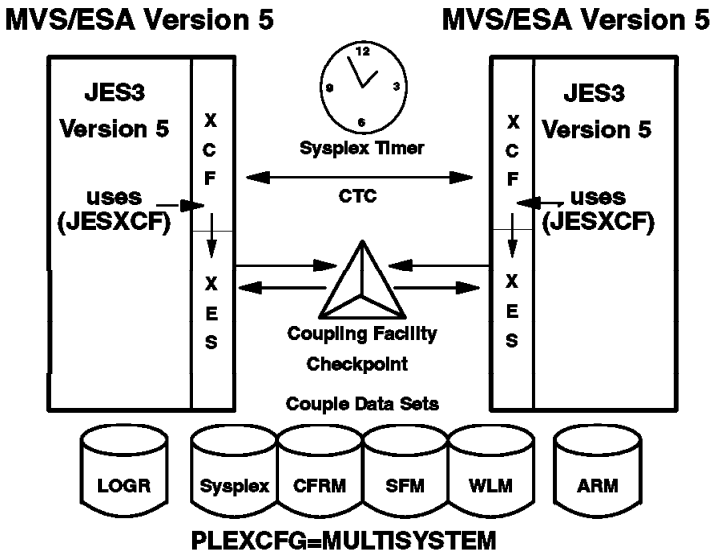


Figure 5. JES3 Multisystem Environment

Note: The general term “sysplex” is used in this chapter to denote any of the three sysplex configurations (XCFLOCAL, MONOPLEX, and MULTISYSTEM) that can be specified on the PLEXCFG= parameter in the IEASYSxx parmlib member. See 3.1, “Sysplex Terminology” on page 35 for a description of the modes.

2.1 XCF Concepts

The MVS cross-system coupling facility (XCF) allows up to 32 MVS systems to communicate in a sysplex. XCF provides the MVS coupling services that allow multisystem application functions (programs) on one MVS system to communicate (send and receive data) with functions on other MVS systems. The multisystem application functions can:

- Define themselves to XCF and leave XCF
- Inform others of their status (for example, active or failed)
- Obtain information about the status of the other instances of the application
- Send and receive messages

In XCF terms, a specific function (one or more routines) of a multisystem application is a *member*. A member resides on one system in the sysplex and can use XCF services to communicate with other members of the same *group*. A group in XCF is defined as the set of related members defined to XCF by the multisystem application. A group can span one or more of the systems in a sysplex and represents a complete logical entity to XCF.

Once a member becomes defined, XCF associates the member with a specific task or job step task (if specified), an address space, and a system. The association of the member with a specific task, job step task, or address space is for termination (resource clean-up) processing only. When the task, job step task, or address space that a member is associated with terminates, XCF terminates the member, thus preventing the member from further use of XCF services. Note, however, that XCF services may be requested by all tasks and SRBs in the member address space as long as the member is active.

Members of XCF groups are unique within the sysplex. However, XCF allows you to define more than one member from the same task or address space, and have those members belong to different XCF groups. This option may be used if the number of members required exceeds the maximum 511 members in a group. Figure 6 shows the XCF group and member association.

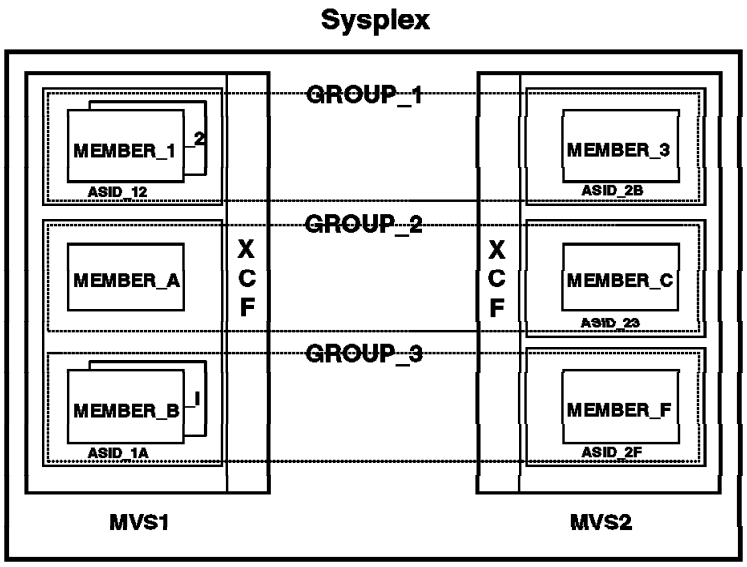


Figure 6. XCF Group and Member Relationship

2.1.1 XCF Services

The following services are provided by XCF:

- **Group Services** - XCF group services provide ways for defining members to XCF, establishing them as part of a group, and allowing them to find out about the other members in the group. If a member identifies a group user routine, XCF uses this routine to notify the member about changes that occur to members of the group, or systems in the sysplex. With a group user

routine, members can have the most current information about the other members in their group without having to query the system.

Specifically, XCF provides the following for setting up, making changes to, and obtaining information about groups and members:

IXCJOIN This macro service defines a member to an XCF group so the member can use the XCF signalling and status monitoring services.

IXCCREAT This macro service defines a member to XCF to be used later during execution.

IXCLEAVE IXCLEAVE, IXCQUIES, IXCDELET, and IXCTERM macro services disassociate members from XCF services. (IXCLEAVE and IXCDELET also disassociate a member from its group.)

IXCSETUS This macro service changes a member's user state value (to be explained later in this chapter).

IXCMOD This macro service changes a member's status-checking interval (to be explained later in this chapter).

IXCQUERY This macro service provides you with information about groups, members, and systems in the sysplex.

- **Status monitoring services** - Status monitoring services are authorized services that notify member exit routines of changes to the status of other members of their group and the systems in the sysplex (the group exit routine). Members can request that XCF monitor their activity (the status exit routine).

  There are two kinds of monitoring services in XCF:

  - System status monitoring

    System status monitoring services monitor systems in the sysplex. The monitoring function uses the system status fields, which are periodically updated. All active application group exits receive control if a system fails to update its status field within a defined interval.

  - Member status monitoring

    Member status monitoring lets a member actively monitor its status. The services are optional, but if a member does use the services, then it has to update its status field periodically. The member status exit gets control if the member fails to update its status field within a specified interval. When the member status exit confirms that the member's state changed, XCF notifies other members about the change through the group exit routines.

  XCF status monitoring services provide a way for members to actively participate in determining their own operational status, and to notify other members of their group when that operational status changes.

  The concept of permanent status recording is closely related to both member states and user states. When a member has permanent status recording, XCF maintains a record of the member's existence (including the member's current member state and user state values) even when the member is dormant or has failed. For permanent status recording to be available, an XCF configuration with a couple data set is required.

- **Signalling services** - Signalling services include authorized assembler macros and an exit routine that members of a group use to communicate

with other members of their group (macros IXCMSGO and IXCMSGI, and the message exit routine).

The signalling services control the exchange of messages between members. The sender of a message requests services from XCF signalling services. XCF uses storage areas to communicate between members in the same system, and it uses the signalling paths to send messages between systems in the sysplex.

XCF signalling services are the primary means of communication between members of an XCF group. Members may send messages to each other as follows:

- A member sends a message by invoking the IXCMSGO macro service. Note that XCF does not necessarily deliver messages in the order in which they were sent.
- To receive the message, the receiving XCF member must be active and must have provided a message user routine. XCF gives control to the message user routine under an SRB. The message user routine receives the message by invoking the IXCMSGI macro service.

Prior to MVS/ESA SP 5.1, the message data had to be sent from and received into a single buffer. In MVS/ESA SP 5.1, a message may consist of message data in a single buffer or in multiple buffers. Message data can also be received into a single buffer or multiple buffers.

Note: XCF services are available only to authorized assembler programs. See *MVS/ESA Programming: Sysplex Services Guide* for more information.

2.1.2 XES Services

Sysplex services for data sharing (XES) allows subsystems running in a sysplex to have data sharing by using a coupling facility. XCF uses XES services when the XCF signalling is defined to go through a coupling facility list structure. JES3 uses XCF signalling services indirectly through JESXCF and XCF:

- Through a coupling facility, when the XCF signalling is directed to use the coupling facility instead of XCF CTCs
- Through XCF CTCs, when the XCF signalling is directed to use XCF CTCs instead of a coupling facility

XCF calls XES services, as shown in Figure 7 on page 21, when the path of communication is a coupling facility instead of CTCs. The communication path is determined by the PATHIN and PATHOUT definitions. The definitions for PATHIN and PATHOUT are initially defined in the COUPLExx parmlib member and can be modified by the SETXCF operator command.

JES3 systems use XES services when the signalling paths used by XCF are a structure in a coupling facility.

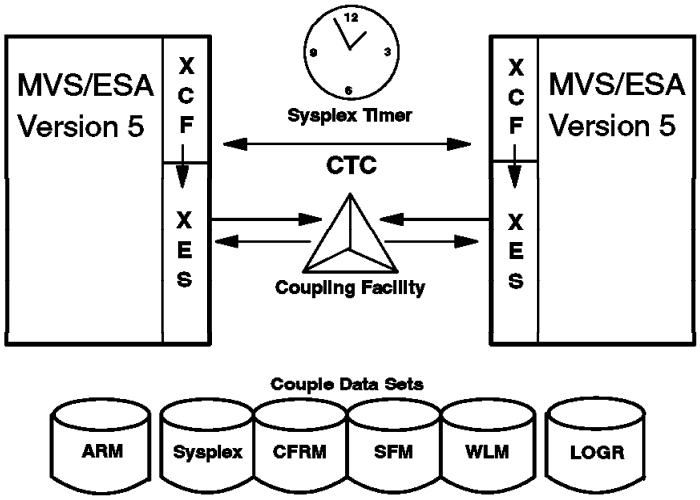


Figure 7. Sysplex Service for Data Sharing

2.2 JESXCF Concepts

The JESXCF address space is used by JES3 to exchange data with other JES3 members in a JES3 complex. This allows JES3 members to communicate using JESXCF. JESXCF uses XCF to do the communication. XCF has the choice of using defined paths through either CTCs or the coupling facility, as shown in Figure 8.

JESXCF is an XCF multisystem application and provides, based on XCF coupling services, common inter-processor and intra-processor communication services for both JES3 and JES2 subsystems. The JESXCF services are tailored to meet JES specific requirements and are provided through macros that enable communication among JES members in a sysplex. The macros are available to either JES3 or JES2 and can be used *only in JES environments*, except for the send message service, which can be used in all environments. The following macro services are available:

IXZXIXAT Create an attachment to JESXCF

IXZXIXMB Build a mailbox

IXZXIXSM Send a message

IXZXIXRM Receive a message

IXZXIXAC Acknowledge receipt of a message

IXZXIXIF Request JES member information

IXZXIXMC Clear a mailbox of messages

IXZXIXMD Delete a mailbox

IXZXIXDT Detach from a JESXCF group

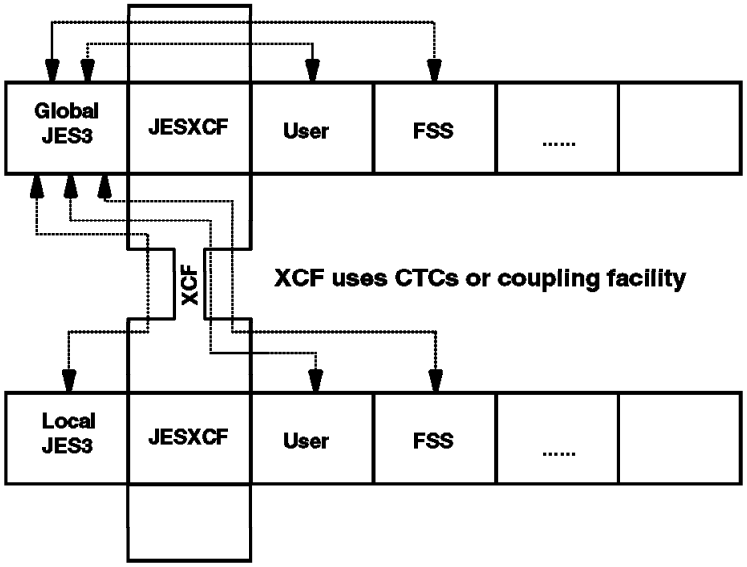


Figure 8. JESXCF Communication in a JES3 Environment

2.2.1 JESXCF in JES3 Environment

On each main in the JES3 complex, JES3 executes in its own address space as the primary subsystem of MVS. The JES3 address space on the global is responsible for maintaining information on:

- Jobs introduced into the system
- Scheduling jobs
- Allocation and deallocation of spool space
- Allocation and deallocation of JES3 managed devices
- Initialization and termination of functional subsystems

If JES3 or a user address space on a local or a user address space on the global requires information that only the JES3 global address space can supply, the requesting address space must be able to pass a request to the JES3 global address space for the information. An address space on a local or a user address space on the global uses the JES3 SSISERV macro to pass the request to the global. After the global has received and processed the request, the JES3 global address space passes the requested information back to the local. JES3 on the global uses the JSERV macro to pass information to a user or to JES3 on a local.

An address space on a local or a user address space on the global usually initiates communications. However, the JES3 global address space can initiate communication to another JES3 address space or to an FSS address space on a local.

When a subsystem communication request is issued, it can be processed on the same main or passed to another main. If an address space issues a request or a response, and the address space that receives the information resides on the

same main, JES3 uses intra-processor communications. If an address space issues a request or response, and the address space that receives the information resides on another main, JES3 uses inter-processor communications.

The JES3 JSERV or SSISERV macro services provide communication between address spaces:

JSERV JES3 global address space issues a JSERV macro to:

- Send a response or information required by a user address space or a JES3 address space on a local
- Start the initialization of an FSS address space

SSISERV Subsystem interface modules and other subtasks of JES3 use the SSISERV macro to communicate with JES3. Parameters on the SSISERV macro are used to specify the:

- Types of requests:

  WAIT The requestor waits for a response.

  REPLY The requestor waits or uses an exit routine to process the response.

  ACK The requestor receives an acknowledgement.

  COMM The requestor does not require a response.

  RESP The requestor is responding to an earlier WAIT, REPLY, or ACK type of request.

  EOMT The end-of-memory type is a special staging area cleanup request where the requestor wants to free all outstanding staging areas.

  PURGE A JES3 function uses JSERV or SSISERV to signal that a staging area (a message in JESXCF terms) has been completely processed and can be discarded.

- Function code of service provided by the JES3 global address space
- The main processor to which the request is to be sent

JES3 Version 5 uses the JES common coupling services as a replacement for its inter-processor CTC communication and intra-processor communication among JES3 mains processors, FSS address spaces, and user address spaces. JES3 continues to use the JSERV and SSISERV services for communication requests. Internally these requests are transformed to corresponding JESXCF requests to invoke the JESXCF services for the actual data transmission.

The JESXCF communication is in the form of messages. In the JESXCF context, a message is any data, including XCF events, that comprises a data packet. A data packet is the data that one JES3 member sends to itself or another member within the JESXCF group. The maximum size of the data is 60KB. If the quantity of data being sent is larger than 60KB, it can be broken into parts and transported as a multi-segment message.

JESXCF messages are received through mailboxes. A mailbox is a logical queue of ordered messages and is maintained by JESXCF in a location associated with a JES3 member. Messages are held in mailboxes until the receiving JES3 receives them or clears the mailbox. When a message is

received, it is preceded by a message envelope. The message envelope is the header for the message and contains information that includes:

- The addresses of the sender and receiver of the message
- An identifier of the message type
- An offset to the actual message

JESXCF requires an acknowledgement for every message that is received. This is required so that JESXCF knows when it can free whatever resources it held for that message.

2.2.2 Create an Attachment to JESXCF (IXZXIXAT)

Before JESXCF messaging (communication) can be used by a JES3 member, the member has to create an attachment to a JESXCF group. During JES3 initialization, each member attaches to a JESXCF group and identifies itself by a unique member name and a JESXCF group name representing the whole JES3 complex. Figure 9 shows how JES3 initialization processing relates to the JESXCF processing.

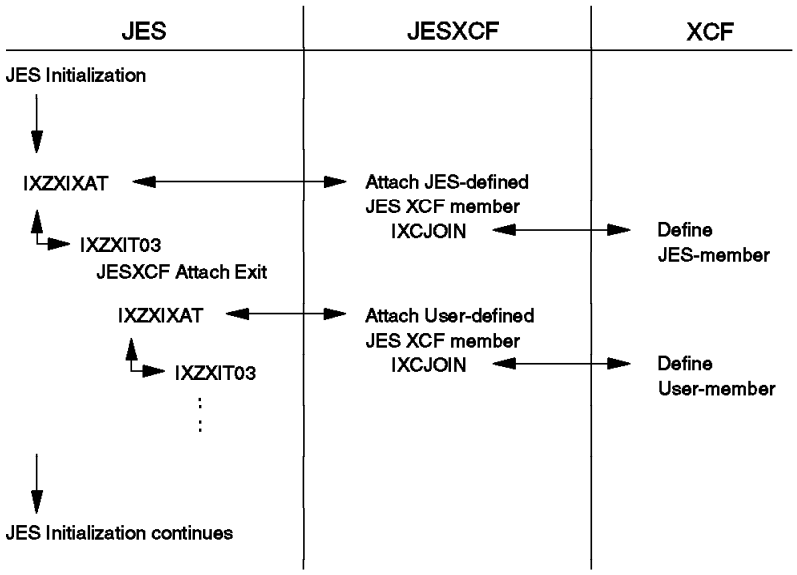


Figure 9. JES3 Initialization Process

During the JES3 initialization or the JES3 FSS connect processing, the JESXCF attach service is invoked (through the IXZXIXAT macro). The JESXCF attach service creates an address space for JESXCF (if it does not already exist) and attaches the JES3 member to a JESXCF group. During the JESXCF attach processing, the MVS XCF group services are invoked (through IXCJOIN) to create an XCF member for the JES3 member. JES3 uses as the XCF group name the XCFGRPNM= specification from the OPTIONS initialization statement if present. If the XCFGRPNM parameter is not specified, the home node name is used. The default name for the home node is N1. See 3.6.2, “JES3 XCF Group Name” on page 42 for specification of the XCF group name. JES3 uses as the XCF member name the main processor name defined on the NAME= parameter on the MAINPROC initialization statement. The JES3 FSSs use as the XCF member name the FSSNAME= specification on the FSSDEF statement.

Installations may create their own independent attachments to JESXCF. If an installation-defined JESXCF group is required, it should be created in the JESXCF attach/detach exit IXZXIT03. Exit IXZXIT03 receives control during the JESXCF attach/detach processing. Care should be taken not to inadvertently cause JESXCF to enter a recursive loop caused by the attach service calling the attach exit.

When JESXCF returns control after a successful attach JESXCF group request, a default mailbox (named SYSJES$DEFAULT) is supplied. This mailbox collects system event data. System event data includes any XCF events on any JES3 member of the JESXCF group and events on the MVS system under which it runs. JES3 does not use the default mailbox and deletes it. The default mailbox is deleted by first attaching to it using the IXZXIXMB service, and then deleting it through the IXZXIXMD service.

Note: Building the SYSJES$DEFAULT default mailbox at this time allows the collection of data while the member is waiting for initialization processing to complete. All messages sent to the default mailbox must be processed and acknowledged. If the default mailbox messages are not processed, JESXCF is not notified to clean up resources held for each message sent to it. This can eventually cause an “out of buffer” condition.

A successful attach JESXCF group request also returns a JESXCF group token that identifies the XCF group. JES3 Version 5 saves the group token in the SVTJXGT field in IATYSVT and TVTXJXGT field in IATYTVTX. A JES3 Version 5 FSS saves its group token in the FSCBJXGT field in IATYFSCB. All subsequent JESXCF service requests require this group token as input.

MVS XCF permanent status recording is used for JESXCF members if the sysplex configuration permits it.

2.2.2.1 Attach/Detach Exit (IXZXIT03)

Exit IXZXIT03 receives control during the IXZXIXAT JESXCF processing, and it may obtain a data area for later IXZXIT01 and IXZXIT02 processing, or it may provide an additional attachment to JESXCF. The IXZXIT03 exit receives control after the JES3 member has joined an XCF group but before either IXZXIT01 or IXZXIT02 receives control.

During detach processing, IXZXIT03 receives control to free the data area obtained during the attach processing or to drop the attachment from JESXCF. After IXZXIT03 receives control, neither IXZXIT01 nor IXZXIT02 receive control for the XCF group from which JES3 is detaching.

Exit IXZXIT03 communicates with JESXCF by setting flag bits and changing data fields in the parameter list passed to the exit.

2.2.3 Build a Mailbox (IXZXIXMB)

JES3 functions that intend to receive messages must build a mailbox (a logical queue of ordered messages that is maintained by JESXCF). This also implies that all JES3 functions that are to receive the JES3 global initiated communication (for example FSS order processing) must create a mailbox. When a JES3 function retrieves and acknowledges a message, JESXCF removes that message from the mailbox. A mailbox may also be cleared (IXZXIXMC) without receiving queued messages. When a mailbox is cleared, JESXCF acknowledges those messages it clears.

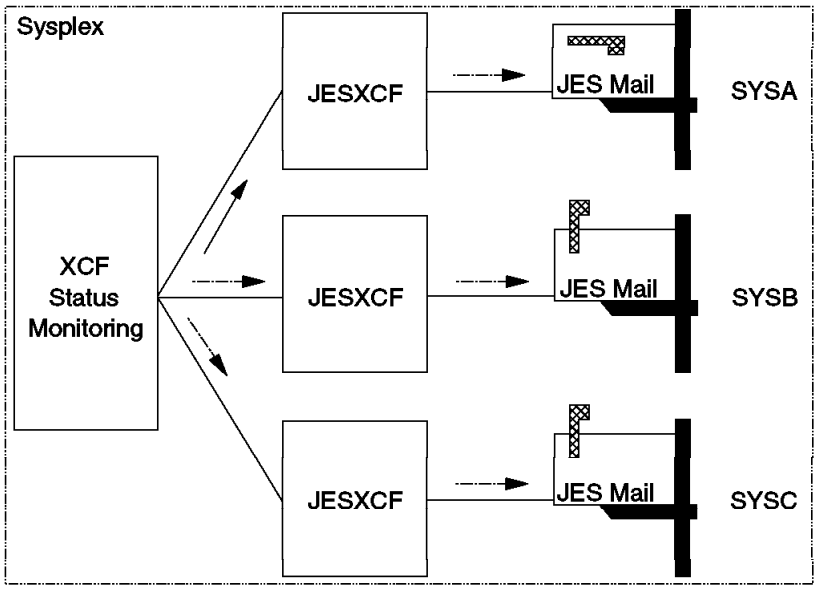


Figure 10. JES3 Mailboxes

Note: A message sender that requested an acknowledgement for messages (for example, a user address space issues SSISERV type ACK) is not required to create a mailbox to receive acknowledgements for the messages.

Mailboxes separate from the default mailbox are used by JES3 to collect messages and acknowledgements sent from and to JES3 functions (including user defined attachments). JES3 builds mailboxes in the following functions:

- FSS/FSA Listen Task - IATFCLT builds a mailbox to receive orders and post FSI requests from the JES3 global.
- Subsystem Communication Initialization - IATINM3 establishes the JESXCF attachment for the JES3 address space and processes the default mailbox by building it and then deleting it.
- Main Service DSP Driver - IATMSDR builds mailboxes that are used in the main connect processing. The mailbox names are formed by appending “SYSJES” with a main processor name.
- FSS/FSA Connect/Disconnect SSI - IATSICD establishes the JESXCF attachment for the CIFSSs and the WTR FSSs. IATSICD processes the default mailbox by building it and then deleting it.
- JES3 Subsystem Communication DESTQ Services - IATSSDS creates a JESXCF mailbox for the DESTQ entries. This is done by specifying as the mailbox name the name that is included in the destination queue entry prefixed by “SYSJES.” All IBM provided mailbox names are prefixed with “SYSRES.”

  The exceptions to this naming convention are the GMS-type and FSS-type destinations. The FSS mailbox names are constructed by converting the FSID into an EBCDIC value and appending it onto the end of the prefix “SYSJES.” The GMS mailbox names are created by appending the main

processor name after prefix “SYSJESMS.” One mailbox for each main processor is created.

The build a mailbox request (IXZXIXMB) must identify a “post exit” routine. This post exit routine receives control under the task that issued the IXZXIXAT macro when JESXCF places a message into a mailbox. The exit routine posts the function that owns the mailbox and returns control to the calling program. This posting action notifies the function that there is a message to receive.

2.2.4 Send a Message (IXZXIXSM)

The JESXCF send a message service is the process of one JES3 member sending a data packet to the same or another JES3 function in the same JESXCF group. A send message request can be issued from a function running under a JES3 main task, a JES3 subtask, JES3 FSS address space, or a user address space. Once a message is sent, JESXCF delivers the message to the requested receiver determined by both the XCF member name and mailbox name. See Figure 11 on page 29 for an example of sending a message.

Unlike XCF, JESXCF maintains the order of the messages sent. All messages are received in the order in which they are sent. Multi-segment messages are not sent until the entire message (all segments) is available.

The message sender identifies the destination of the message by supplying the member name and the mailbox name of the receiver. This receiving member can be any member of the JESXCF group, including the sending member.

2.2.4.1 Message Types

Delivery and subsequent message processing is determined by the message type characteristics that are specified on the IXZXIXSM macro:

COMM This is the default request type for the IXZXIXSM macro. COMM indicates that the message being sent is communicated asynchronously, the sender does not request an acknowledgement, and JESXCF does not attempt multisystem recovery if either the sending or receiving member fails prior to message delivery.

This type is used when the message data is informational and there is no critical follow-on processing.

ASYNCACK Indicates that the message being sent is sent asynchronously and the sender requests an acknowledgement from the receiver once the message is processed.

This type is used when the message data is critical and the sender does not need to know immediately that the message data was received, but the sender requires to know that the message is processed successfully before any further processing can be done by the sending function.

ASYNC Indicates that the message being sent is sent asynchronously. This option is similar to REQTYPE=COMM, except that ASYNC requests that JESXCF provide recovery by resending the message if the receiving member fails prior to acknowledging the message.

This option is used when the message data is critical and the sender does not need to know that the message data was received but must ensure that JESXCF delivers the message to the receiver.

SYNC Indicates that the message is sent synchronously. The sender is placed in a wait state until an acknowledgement is received for the message. JESXCF provides recovery even if the receiving member fails prior to receiving or acknowledging the message.

This option is used when the message data is critical and the sender cannot continue until an acknowledgement (with data) is received.

JES3 Version 5 subsystem communication uses for the JSERV and SSISERV types the following JESXCF services:

WAIT IXZXIXSM request SYNC

REPLY IXZXIXSM request ASYNCACK

ACK IXZXIXSM request ASYNC

COMM IXZXIXSM request COMM

RESP IXZXIXAC (with data)

PURGE IXZXIXAC

2.2.4.2 Transport Exit (IXZXIT01)

Exit IXZXIT01 enables you to view, modify, add to, or reroute a message prior to the message being delivered to the receiving member. This exit gets control for the send message and the acknowledge message macro invocations.

Exit IXZXIT01 receives control under the same task that issues the IXZXIXSM or IXZXIXAC macros. JESXCF invokes the exit after a message packet has been created but before that packet is sent to its target member, and when the acknowledging member replies to the originating system for all acknowledgements except for the request COMM type message acknowledgements.

The data passed to this exit is in a data space; therefore, the data can be accessed through an access register. The exit receives control in access register address space control (AR ASC) mode. AR1 contains the ALET of the data space containing the exit parameters.

2.2.5 Receive a Message (IXZXIXRM)

Once a message is sent to a mailbox, the receiving member is informed through the POST exit routine that was specified on the POSTXIT= parameter of the IXZXIXMB macro. This posting action notifies the mailbox owner that there is a message to receive.

When the receiving function issues the receive macro, IXZXIXRM, the receive message exit (IXZXIT02) gets control first. This exit allows the receiving member to view or modify the message and to receive any extents that have been added to it by the transport exit. Once the message is received, the receiving function acknowledges the mail, informs JESXCF of its receipt, and if requested by the sending member, also returns acknowledgement data to the sender.

The receiver of a message must:

- Provide the group token that represents the JESXCF member and group
- Identify the mailbox from which to retrieve the message

- Indicate the type of message to be received:
  - Only system event messages
  - Only acknowledgement messages
  - Only JES- or installation-created messages

  Each segment of a multi-segment message is presented as a single message, and all segments are ordered as originally sent. However, individual messages from multiple members might not be ordered in the mailbox in a timestamp order.

JES3 Version 5 subsystem communication services receive messages (staging areas) sent to the global and place them on a destination queue located off the destination queue table (DSQ). The destination queue that the staging area is placed on is determined by the FUNC parameter on the SSISERV macro. JES3 DSPs retrieve the queued staging areas through the DSQLOC macro service.

See Figure 11 for an overview summary of JESXCF message transmission and reception.

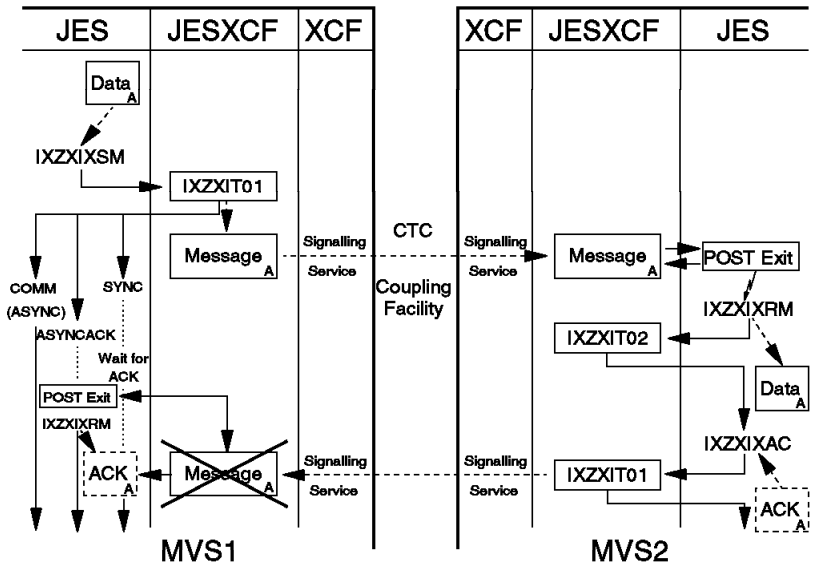


Figure 11. Summary of JESXCF Message Transmission and Reception

2.2.5.1 Receive Exit (IXZXIT02)

The IXZXIT02 exit enables you to view or modify messages before they are retrieved from a mailbox. The exit can also retrieve any message extents that the transport exit IXZXIT01 might have added. This exit gets control for IXZXIXRM (receive message) macro invocations.

The data passed to this exit is in a data space; therefore, the data can be accessed only through an AR/GPR pair. The exit receives control in AR ASC mode. AR1 contains the ALET of the data space containing the exit parameters.

2.2.6 Acknowledge Receipt of a Message (IXZXIXAC)

All received messages *must* be acknowledged to JESXCF independent of whether the sender requested an acknowledgement or not. After a message is acknowledged, JESXCF releases all resources held by the original message.

If the sender of the message requests an acknowledgement, the JESXCF returns an acknowledgement message as follows:

- To the mailbox specified on the ASYNCACK type send request as the mailbox into which the acknowledgement message is to be returned
- To the area specified on the SYNC type send request

An acknowledgement may contain message data and a return code that is to be returned with the acknowledgement. The data is made available as part of the acknowledgement message. The format and content of the data and meaning of the return code must have been agreed to by the sending and receiving JES3 functions.

2.2.7 Request JES Member Information (IXZXIXIF)

The request JES3 member information service (IXZXIXIF) returns a record of data about the specific group member, and also information about the MVS system on which the member is running. The data is returned either through a mailbox or is placed directly into the requestor's data area in an array format. If the data is returned to the mailbox, JESXCF treats it as system event data.

The returned member information includes:

- Request token that was returned from IXZXIXIF service
- The release level of the JES product
- JESXCF maintenance level
- XCF group name
- XCF member name
- MVS system name that the JES is running on
- User state information as set by IXZXIXUS macro service
- XCF member token
- XCF sysplex token
- Member status:
  - Member is active, connection between the JESXCF address space and the JES address space is functioning.
  - MVS XCF state of the member is active but the connection between the JESXCF address space and the JES3 address space is not functioning. The probable cause is a JES3 ABEND.
  - Both MVS XCF status and JESXCF connection status indicate that the member is not active.
- XCF system token

2.2.8 Clear a Mailbox of Messages (IXZXIXMC)

The clear a mailbox service (IXZXIXMC) cleans the specified mailbox of all messages that are in the mailbox but have not yet been received. Every message that is removed from the mailbox is acknowledged to the sending JES3 member with an indication that the message was cleared but not processed by the receiver.

Typically, the clear mailbox service is used by a JES3 function that has restarted and does not want to process messages that were received from the previous JES3 start.

2.2.9 Delete a Mailbox (IXZXIXMD)

The delete mailbox service (IXZXIXMD) deletes a mailbox and frees any associated resources. All messages still in the mailbox are deleted, and appropriate acknowledgements are returned to the sender of the messages.

When JES3 terminates (except JES3 FSS), mailboxes are not deleted. However, when JES3 is restarted, the mailboxes are rebuilt with the same names they had prior to the termination.

2.2.10 Detach from a JESXCF Group (IXZXIXDT)

The detach from a JESXCF group service (IXZXIXDT) deactivates and removes a JES from an XCF group.

The JES3 FSS/FSA Connect/Disconnect SSI routine invokes IXZXIXDT during FSS disconnect processing. JES3 mains do not detach from the JESXCF group during JES3 termination.

2.2.11 JESXCF Message Types

Each message is prefixed by a message envelope, mapped by IXZYIXEN, for the following types of messages in a mailbox:

- System events (SYSEVENT)
- Acknowledgements (ACKS)
- Messages (MESSAGES)

Figure 12 on page 32 shows the JESXCF message mapping structures.

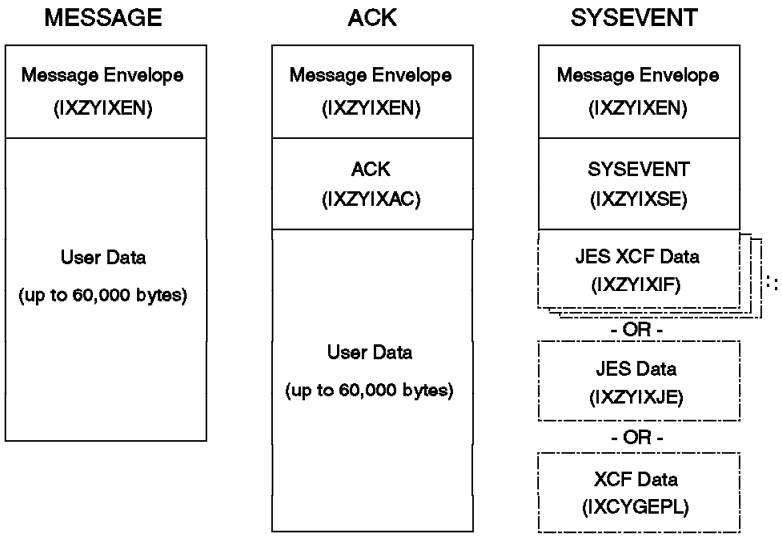


Figure 12. JESXCF Message Mapping

2.2.11.1 Message Envelope (IXZYIXEN)

The message envelope includes information such as:

- Status of the message in the mailbox:
  - Message has been resent to the receiving system because the receiving system was re-IPLed.
  - Message has been rerouted.
  - Message was present in the mailbox when the attacher disconnected.
  - Message has been received.
  - Message has been checkpointed.
- Maintenance level of the JESXCF component
- Message sequence number
- Address information of the receiver of the message (group name, member name, and mailbox name)
- Address information of the sender of the message (group name, member name, and mailbox name)
- Type of the send message request:
  - Synchronous message
  - Asynchronous message that does not return an acknowledgement message to the sender
  - Asynchronous message that returns an acknowledgement message to the sender
  - Asynchronous message that will not be resent to the receiver if the receiving system re-IPLs. No acknowledgement will be sent to the sender of the message
  - Acknowledgement message
  - Acknowledgement message sent because time specified on the on the IXZXIXSM macro has expired
- Single segment or multi-segment message indicators.

- Content of the message:
  - A system event
  - An acknowledgement message
  - Application message
- Length of the message not including the envelope

2.2.11.2 System Event (SYSEVENT)

The system event data includes three types of system event data:

JESXCF Data (IXZYIXIF) JESXCF data provides information about the JES and XCF connections, such as notification of an event detected by the JESXCF address space or such as termination of the connection between JESXCF and the JES address space. The data returned by the IXZXIXIF service is described under 2.2.7, “Request JES Member Information (IXZXIXIF)” on page 30.

JES Data (IXZYIXJE) JES data provides a notification of events that the JESXCF address space has detected, such as termination of the connection between JESXCF and the JES address space. The JESXCF data includes:

- Event type:
  - Connection between JESXCF and specified JES terminated
- Group name of the member whose connection terminated
- Member name of the member whose connection terminated
- The request token for the message that timed out

XCF Data (IXCYGEPL) The XCF data is the same that is passed to the XCF group user exit of an active member. XCF data notifies about changes that occur to the XCF members of a group, or changes to the systems in the sysplex. See *MVS/ESA Programming: Sysplex Services Guide* for more information. The XCF data includes:

- Member data provided through the IXCJOIN that established the group exit
- Invocation event:
  - 1 - Member state event
  - 2 - User state event
  - 7 - Member status update missing reported by subsystem′s status exit
  - 8 - Member status update missing detected by subsystem monitor
  - 9 - Member status update no longer missing
  - 11 - System reported active
  - 12 - System update missing
  - 13 - System update resume
  - 14 - System reported going
  - 15 - System reported gone
  - 16 - System detected missing
  - 17 - System detected gone

  - 18 - System failure detection interval updated
- Member state before the action (listed above) was taken
- Member state after the action
- Member's current status
- XCF group name
- XCF member name
- Member token
- System name where member is (was) active
- Time this event occurred
- User data

2.2.11.3 Acknowledgement (ACK)

An acknowledgement provides notification information on delivery of messages issued through the IXZXIXSM macro service. The acknowledgement data includes:

- Request token for the message that this acknowledgement is for
- Return code information returned by the receiving routine
- Length of the data returned to the sender through the IXZXIXAC macro service

2.2.11.4 User Data (MESSAGE)

JESXCF does not impose any format requirements on user data.

For more information about JESXCF, see *MVS/ESA Programming: JES Common Coupling Services*.

A sample JES3 DSP that uses JESXCF services is shown in Appendix A, "Sample Program Using JESXCF Services" on page 153. The DSP monitors system events from JESXCF. When an XCF "system reported gone" is received for a local JES3 main, the local is flushed (using the *START local_name FLUSH operator command). The DSP varies a local main online to the global (using the *VARY local_name ON operator command) when an XCF "system reported active" system event is received. The *VARY local_name ON is required because the flush processing varies the local main offline.

Chapter 3. JES3 Enhancements for Sysplex

Sysplex stands for *Sys*tems Com*plex*. A sysplex implementation without a coupling facility is referred to as a *base sysplex*. When the implementation includes a coupling facility, it is called a *parallel sysplex*. This chapter describes the JES3 requirements for setting up a sysplex.

The base sysplex, introduced in September 1990, lays the groundwork for simplified multisystem management through the cross-system coupling facility (XCF). In a base sysplex, central processing complexes (CPC) are connected through channel-to-channel (CTC) adapters and a shared coupling data set. When the sysplex consists of multiple MVS systems running on two or more processors, MVS requires that the processors be connected to the same Sysplex Timer. MVS uses the Sysplex Timer to synchronize TOD clocks across systems. The base sysplex allows up to eight MVS systems in a sysplex.

MVS/ESA SP 5.1.0 extends the sysplex to provide high-performance data sharing through a new coupling technology. It also increases the number of MVS systems in a sysplex from eight to 32. The capability of linking many systems and providing multisystem data sharing makes the sysplex ideal for parallel processing, particularly for online transaction processing.

MVS/ESA SP 5.2 with JES3 5.2.1 provides sysplex-wide control from any console on any system in a sysplex. All consoles in the sysplex, except for JES3 remote consoles and NJE consoles, have the ability to receive message traffic from all systems in the sysplex and have the ability to send commands to any system in the sysplex. In previous JES3 releases, only consoles defined to JES3 with proper associations and attached to the global processor could control all systems in the JES3 complex.

3.1 Sysplex Terminology

There are a number of terms dealing with sysplex that are used throughout this document. Some, you may be familiar with but others are new.

Sysplex A sysplex is a set of one or more MVS systems that are initialized into the same sysplex with the same sysplex name and are fully connected by XCF signalling services. They must also all be connected to the same couple data set except in the case of an XCFLOCAL sysplex. The word *sysplex* is formed from the contraction of **sys**tems and com**plex**.

Parallel Sysplex When the sysplex includes a coupling facility, it is referred to as a parallel sysplex. This applies, even when the signalling paths are not coupling facility channels.

Sysplex Timer Also referred to as an external time reference (ETR). The Sysplex Timer (IBM 9037) is an external clock used for synchronizing the TOD clocks of CPCs that form the sysplex. It is only required if the sysplex consists of more than one CPC. MVS parameters in SYS1.PARMLIB and MVS system commands use the abbreviation ETR when referring to the Sysplex Timer.

CTC A CTC is used for channel-to-channel connectivity. It is a form of direct connection between processors or between channels of the same processor. In the context of this document, this type of connection refers to an ESCON channel operating in CTC mode. CTC links may be used when a sysplex is made up of two or more systems.

CF Channels Coupling facility channels are high bandwidth fiber optic links that provide high speed connectivity between the coupling facility and CPCs that use the coupling facility. Coupling facility channels are directly attached between the CPCs and the coupling facility. They can be used not only for transporting shared data but as an alternative to ESCON CTCs. With this option, fewer ESCON channels are necessary, and the sysplex connectivity is simplified and easier to manage.

Coupling Facility The coupling facility (CF) is either a S/390 microprocessor CPC or a PR/SM LPAR on an ES/9000, 711 (H5) based CPC. It forms the base for high performance multisystem data sharing among certain ES/9000 processors and within a cluster of S/390 microprocessors. MVS/ESA services are available to authorized applications, such as subsystems and MVS components, to use the coupling facility to cache data, exchange status, and access sysplex lock structures so that techniques conducive to high performance data sharing and fast failure recovery can be implemented.

Couple Data Set There are four types of couple data sets: *SYSPLEX, CFRM, SFM* and *WLM*. These data sets are shared by all the systems that make up the sysplex and are necessary depending on the policies you define to help manage resources and the workloads for the sysplex. The sysplex couple data set is mandatory and contains XCF related data about the sysplex, systems, groups, members and general status information. The sysplex couple data set is required to run a sysplex in either multisystem or monoplex mode. Data in the sysplex couple data set cannot be combined with the policy definitions for the other couple data sets. However, you can combine the CFRM, SFM and WLM policies in the same couple data set if you desire to.

XCF The cross-system coupling facility (XCF) is the operating system component that controls members and groups, provides inter-member communications (exchange data, programs, and so on), and monitoring services for members.

XES Cross-system extended services (XES) is an extension to the XCF component. XES provides the sysplex services that applications and subsystems use to share data held in the coupling facility. These services support the sharing of cached data and common queues while maintaining data integrity and consistency.

CFRM Policy The coupling facility resource management (CFRM) policy lets you determine how MVS is to manage the coupling facility resources. It resides in the CFRM couple data set.

SFM Policy The sysplex failure management (SFM) policy determines how the system reacts to system failures and signalling connectivity failures, and how PR/SM reconfiguration actions are to be managed. You can define a number of policies in the SFM couple data set, but only one policy can be active.

WLM Policy In the workload management (WLM) policy, you set out the service goals for your various workloads. This policy resides in the WLM couple data set.

Group A group is a collection of related members.

Member A member is a specific function (one or more routines) of a multisystem application that is joined to XCF and assigned to a group by the multisystem application.

3.2 XCF Modes of Operation

There are three sysplex modes in MVS/ESA SP Version 5. The sysplex mode can be changed only by an IPL. You specify the sysplex mode in the IEASYSxx member in SYS1.PARMLIB. The three modes are:

XCFLOCAL A system in XCFLOCAL mode runs as a single-system sysplex. This mode is the nearest equivalent to the way systems run in MVS systems prior to MVS/ESA SP Version 4. With XCFLOCAL mode, it is not possible to have more than one system in the sysplex. XCF does not provide any XCF inter-system signalling services. However, multisystem applications can use signalling services to communicate (exchange data, programs, and so on) between members in the same system. The members can be in the same address space or in different address spaces. The system does not use couple data sets or XCF CTCs. GRS global serialization is not required in this mode. You would probably choose this mode if you are not planning to use XCF facilities or as the first stage of a migration to a full sysplex.

MONOPLEX The system runs as a single-system sysplex. MONOPLEX mode is also a single-system sysplex mode in which XCF does not provide any XCF inter-system signalling services. Multisystem applications can use signalling services to communicate (exchange data, programs, and so on) between members in the same system. The members can be in the same or different address spaces. XCF does not allow any other systems to enter this sysplex. The difference in configuration between XCFLOCAL and MONOPLEX modes is that MONOPLEX mode requires a couple data set. The couple data set makes it possible for applications using XCF monitoring services to use permanent status recording. Just as in XCFLOCAL mode, XCF CTCs are not used in MONOPLEX mode. GRS global serialization is also not required for MONOPLEX mode. MONOPLEX mode might be chosen as part of an intermediate sysplex migration stage to familiarize operators and systems programmers with sysplex before installing CTCs and to allow application programmers to experiment with multisystem applications.

MULTISYSTEM MULTISYSTEM mode is the multisystem sysplex mode. In MULTISYSTEM mode, the sysplex consists of one or more MVS systems. All systems in the sysplex must use the same couple data set, and if they are not all on one processor, they must all be connected to the same Sysplex Timer. At least two intersystem connections (CTC links) are required between every pair of systems in the sysplex. Multisystem applications can use XCF signalling services to communicate among members in the same system, as well as among members that reside on different systems. Systems running in MULTISYSTEM mode must also specify global resource serialization (that is, must specify either TRYJOIN, JOIN, or START on the GRS= keyword in IEASYSxx).

3.3 JES3 5.1.1 Enhancements for Sysplex

The following list contains the enhancements made to JES3 5.1.1 in support of an up to 32-way sysplex:

- 32-way sysplex enablement - the changes that are required to allow more than eight JES3 main processors in a sysplex.

  The size of some initialization intermediate spool records may span a spool buffer because of sections that are repeated for the maximum number of processors in a JES3 complex. The ITREAD and ITWRITE services are enhanced to process initialization files that span a spool buffer.

- Availability - improved JES3 restart for Main Device Scheduler (MDS) by adding multiple secondary SETUP (MDS) FCTs.

  Main connect processing does not begin until after MDS restart processing is complete. MDS restart processing currently runs under a single FCT. JES3 Version 5 creates multiple secondary FCTs (for MDSREST DSP) during MDS restart to read, rebuild, and write MDS control blocks.

- Systems Management - usability improvements to several JES3 operator commands and migration aids assist in the definition of I/O devices in the JES3 initialization stream.

3.4 JES3 5.2.1 Enhancements for Sysplex

The following list contains the enhancements made to JES3 5.2.1 in support of an up to 32-way sysplex:

- Multisystem console support is enabled when running JES3 5.2.1 and you have sysplex-wide control from any console on any system:
  - MCS
  - NetView (and similar products)
  - System Console
  - Extended MCS consoles
  - TSO (CONSOLE command)
- Transport of messages and commands between systems in the sysplex is controlled by MCS:
  - For command routing within the sysplex, the MVS ROUTE command replaces the JES3 *T command.

  - Command Prefix Facility (CPF) is used to register JES3 command synonyms (1-8 characters).
  - *T continues to support NJE command routing.
- JES3 *does not* control the flow of messages and commands across systems in the JES3 complex.
- Message Processing Facility (MPF) processing, including MPF exits, occurs only on the originating system:
  - JES3 provides an option to pass all message traffic *received* on the JES3 global through MPF processing on the global.
- MVS subsystems see messages on SSI on all systems eligible to display the message.
- JES3 command processors support 4-byte console IDs and preserve command and response tokens (CART).
- JES3 *INQUIRY and *MODIFY command processors identify command responses using the appropriate descriptor code.
- MVS and JES3 hardcopy log may be maintained using Operations Log. MVS operations log provides sysplex-wide services using the MVS system logger.
- Optionally JES3 maintained DLOG may be used for logging.
- JES3 RJP and NJE console support redesigned to exploit extended MCS consoles.
- Deleted JES3 functions:
  - Support for JES3 managed operator and MLOG (JES3 log format written to a hardcopy media or a JES3 managed display console) consoles is removed, as they are no longer required to manage the sysplex.
  - JES3 managed action message retention facility (JMRF) - totally replaced by MCS AMRF.
  - Support for entry of JES3 commands from a BDT session, along with the corresponding command authorization exit IATUX56.

The JES3 system environment of one global processor and many (up to 31) local processors is maintained.

3.5 JES3 Version 5 and Sysplex

Figure 13 shows the complexity of managing a very large sysplex containing a 32-way global/local configuration. For a 32-way sysplex to be a workable system, JES3 must allow up to 32 members to be connected in the configuration and allow the systems to be managed easily by the operators.

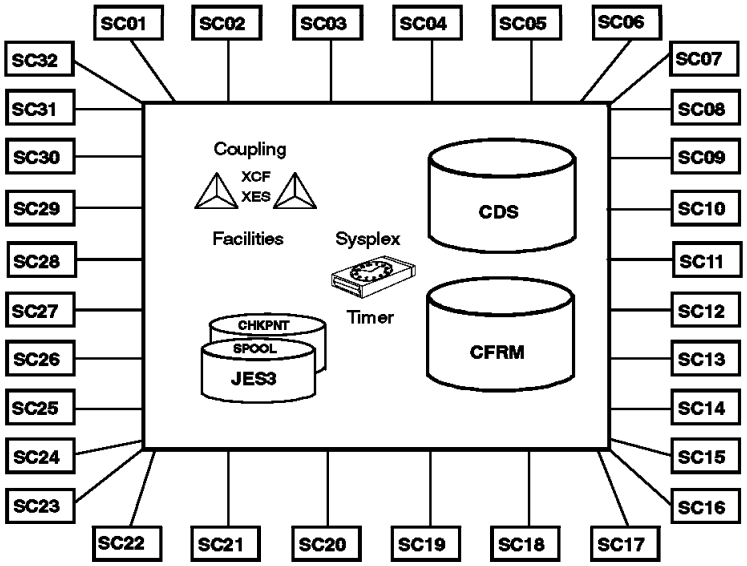


Figure 13. JES3 32-Way Sysplex

Beginning with JES3 Version 5, JES3 mains must join the MVS sysplex. Communications between JES3 mains are through XCF signalling. For XCF communications, JES3 uses whatever connections are supported by MVS/ESA, which could be the coupling facility or ESCON CTCs.

3.5.1 JES3 and JESXCF Services

After converting to JES3 Version 5 and with more than one processor in the JES3 complex, a sysplex environment is required. In prior releases, JES3 used CTC connections for inter-processor and intra-processor communication. JES3 Version 5 uses the JESXCF component to communicate between JES3 mains.

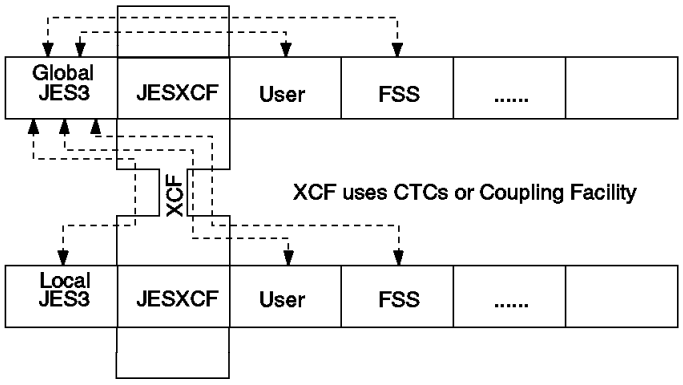


Figure 14. JES3 Configuration in a Sysplex

JES common coupling services (JESXCF) is an XCF multisystem application and provides, based on XCF coupling services, common inter-processor and intra-processor communication services for both JES3 and JES2 subsystems. The JESXCF address space, shown in Figure 14, is created in each system

during the first JES3 initialization. As each JES3 is initialized, it joins the JES3 XCF group. JESXCF can be described in the following ways:

- A component of the MVS/BCP.
- Runs as a server address space with one server address space on each MVS image.
- Does not start until JES starts.
- Data is kept in a separate dataspace.

The JESXCF services are tailored to meet JES specific requirements and are provided through macros that enable communication among JES members in a sysplex. The macros are available to either JES3 or JES2 and can be used *only in JES environments* except for the send message service, which can be used in all environments. For a more detailed description, see Chapter 2, "JES3 Version 5 and JES Common Coupling Services" on page 17. JES3 Version 5 uses JESXCF services for:

- Joining and leaving the sysplex.
- Staging area transport between the JES3 global and other address spaces (JES3 local, user, FSS) in the JES3 complex.

3.6 Planning for Sysplex

A JES3 Version 5 complex is required to be contained within one sysplex. The following considerations are necessary for a migration to sysplex:

Sysplex name A name specified for the sysplex in the COUPLExx member of SYS1.PARMLIB.

Group name A group in XCF is defined as the set of related members defined to XCF by a multisystem application. A member of an XCF group resides on one system in the sysplex and can use XCF services to communicate with other members of the same group. Each group must have a unique XCF group name in the sysplex. With JES3 Version 5, a JES3 global address space, each JES3 local address space, and each FSS address space become members of the same XCF group.

member name Each member of the XCF group must have a unique member name. JES3 provides the member name upon attaching to JESXCF.

3.6.1 COUPLExx Parmlib Member

There are two SYS1.PARMLIB members that contain parameter values when defining a sysplex, IEASYSxx and COUPLExx. Figure 15 on page 42 shows a minimum COUPLExx member definition for a sysplex and PLEXCFG=MULTISYSTEM mode configuration. Not shown here are other couple data sets and other users of the coupling facility. The sysplex name defined is WTSCPLX1, and the sysplex couple data sets are defined via the PCOUPLE and ACOUPLE keywords.

```
/*********************************************************************/
/*                                                                   */
/*  MEMBER = COUPLE00                                                */
/*                                                                   */
/*  DESCRIPTION = THIS PARMLIB MEMBER IS ACCESSED THROUGH THE        */
/*                IEASYSXX MEMBER BY SPECIFYING COUPLE=00            */
/*                                                                   */
/*********************************************************************/

 COUPLE SYSPLEX(WTSCPLX1)
        PCOUPLE(SYS1.XCF.CDS10)
        ACOUPLE(SYS1.XCF.CDS20)
        INTERVAL(85)
        OPNOTIFY(85)
        CLEANUP(30)
        MAXMSG(500)
        RETRY(10)
        CLASSLEN(1024)

 /* PATH DEFINITIONS FOR DEFAULT SIGNALLING */

    PATHIN  DEVICE(4010,4020,4030,4040,4050,4130,4150)
    PATHIN  DEVICE(4018,4028,4038,4048,4058,4138,4158)
    PATHOUT DEVICE(5010,5020,5030,5040,5050,5130,5150)
    PATHOUT DEVICE(5018,5028,5038,5048,5058,5138,5158)

    PATHOUT STRNAME(IXC_DEFAULT_1,IXC_DEFAULT_2)
    PATHIN STRNAME(IXC_DEFAULT_1,IXC_DEFAULT_2)
```

Figure 15. SYS1.PARMLIB Member COUPLExx Example

3.6.2 JES3 XCF Group Name

The XCF group name is required for each system when it joins the sysplex. The group name can be defaulted or you can choose the XCF group name on the OPTIONS statement.

```
OPTIONS   XCFGRPNM=groupname
```

JES3 determines the XCF group name to be used by making the following decisions:

1. Use the groupname from the OPTIONS XCFGRPNM= initialization parameter. The XCFGRPNM= parameter is optional.
2. Use the nodename from the NJERMT NAME= specification.
3. Use the default nodename of *N1* when no NJE networking is defined.

Note: The recommendation is to let the group name default to the node name. Due to syntax constraints, this may not be possible (the allowable node names in JES3 are a superset of allowable XCF group names). See 4.1, “JES3 Version 5.1.1 Initialization Changes” on page 53 for a description of the OPTIONS statement XCFGRPNM group name keyword restrictions.

3.6.3 JES3 XCF Member Names

Each member of an XCF group is identified by name. This member name is either the JES3 main name defined on the MAINPROC statement for JES3 global and local address spaces or the FSS name on the FSSDEF statement.

Note: The MAINPROC name *must* match the SYSNAME parameter specified in the IEASYSxx parmlib member.

3.6.4 Defining System Names

Table 1 and Table 2 show where the specifications can be made to choose a naming convention in which the SYSNAME, JES3 main name, and SMF SID are the same.

Table 1. SYS1.PARMLIB Definitions

PARMLIB Member	**Parameter Specification**
IEASYSxx	SYSNAME=SC50
SMFPRMxx	SID(&SYSNAME(1:4))

Table 2. JES3 Initialization Statements

Initialization Statement	**Parameter Specification**
MAINPROC	NAME=SC50
MAINPROC	NAME=SC49

Appendix B, “SYS1.PARMLIB Definitions for a Sysplex” on page 159 contains sample parmlib members for defining the system name and the SMF SID.

3.7 JES3 Version 5 Configurations

After migrating to JES3 Version 5 and with one or more than one processor in the complex, a sysplex environment is required. The MVS multisystem sysplex is only required if the JES3 complex has more than one JES3 main.

You must specify the type of sysplex configuration using the PLEXCFG parameter in the IEASYSxx member, as shown in Figure 16 on page 44, Figure 17 on page 45, and Figure 18 on page 45. You must also specify a COUPLExx parmlib member to identify the sysplex couple data sets and the signalling paths between systems using the PATHIN and PATHOUT statements, as shown in Figure 15 on page 42, when running in PLEXCFG=MULTISYSTEM mode.

3.7.1 Global Only JES3 Configuration

With a single MVS/JES3 complex, you can define the XCF configuration as PLEXCFG=XCFLOCAL or PLEXCFG=MONOPLEX, as shown in Figure 16.

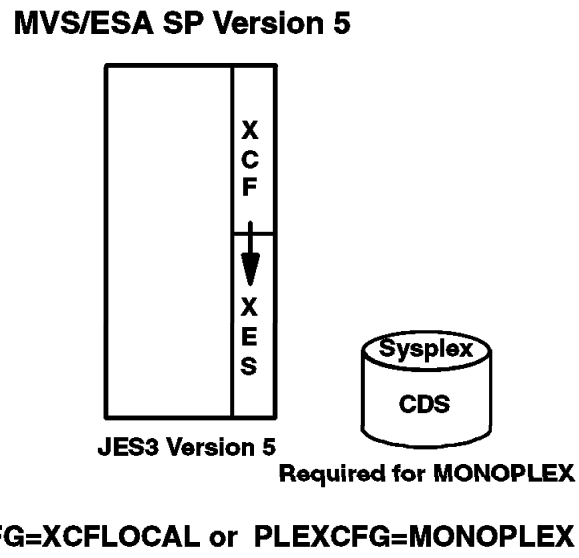


Figure 16. JES3 Global Only Configuration

Note: A Sysplex Timer is not required for any of the XCF modes specified in this configuration.

Consider using PLEXCFG=MONOPLEX in a single system environment because MVS/ESA Version 5 functions you may want to use require sysplex couple data sets. The advantage of specifying MONOPLEX mode is that it allows the MVS/ESA SP Version 5 workload manager (WLM) to be used in goal mode. WLM goal mode requires a function couple data set, and therefore, MONOPLEX mode is the minimum XCF mode of operation. See 3.2, “XCF Modes of Operation” on page 37. In MVS/ESA Version 5.2, the automatic restart management function also requires a couple data set. See 7.2, “Automatic Restart Management” on page 116, for JES3 support of this function.

Note: It is possible to have a coupling facility in the single system environment, which also requires couple data sets making it necessary to use PLEXCFG=MONOPLEX mode.

3.7.2 Global-Local JES3 PR/SM Configuration

With multiple MVS images in a single CPC environment using PR/SM, PLEXCFG=MULTISYSTEM is the XCF mode required. A Sysplex Timer is not required as PR/SM provides a common time reference, as shown in Figure 17. The JES3 systems communicate using XCF services, and the signalling services can be through either CTCs or the coupling facility.

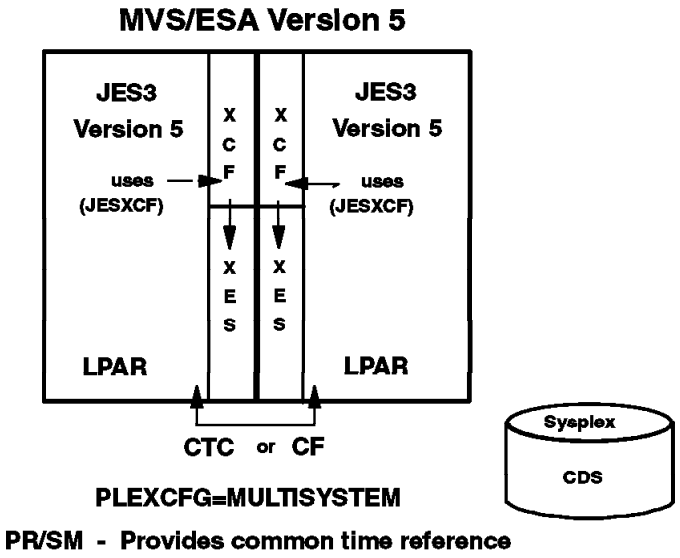


Figure 17. Global-Local JES3 PR/SM Configuration

3.7.3 Global-Local JES3 Multiple CPC Configuration

With multiple MVS images in multiple CPCs, PLEXCFG=MULTISYSTEM is the XCF mode required. A Sysplex Timer is required, as shown in Figure 18. The two JES3 systems communicate using XCF, and the signalling services can be through either CTCs or the coupling facility. See 3.8, “Planning for Coupling Facility Usage” on page 47 for more details.

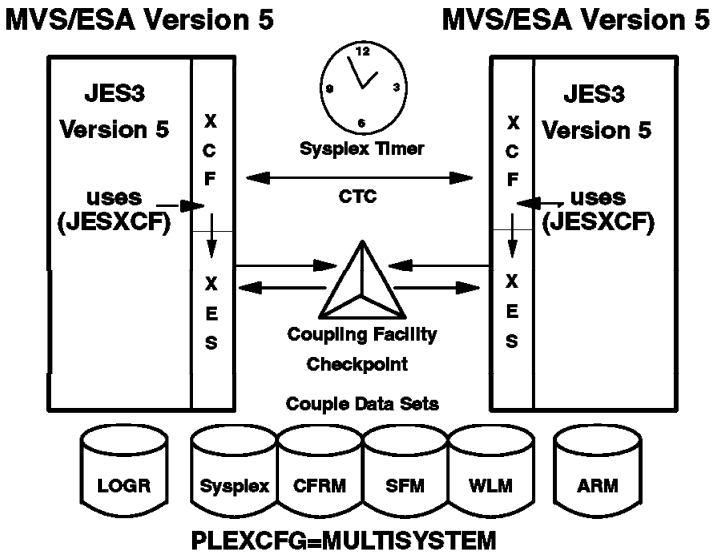


Figure 18. Global-Local JES3 Multiple CPC Configuration

Note: A Sysplex Timer is required in this configuration.

3.7.4 Sysplex Operator Commands

Table 3 summarizes the display XCF operator commands. For additional information, see *MVS/ESA SP V5 System Commands*.

Table 3. Display XCF Command Capabilities

Command	Action
D XCF,SYSPLEX	Display the sysplex name and the names of the systems in the sysplex.
D XCF,S,ALL	Display the active systems in the sysplex and their last update time.
D XCF,COUPLE,TYPE=CFRM	Display the in-use CFRM policy.
D XCF,PATHIN	Display what signalling paths are defined and being used by PATHIN. This includes both device and structures signalling path.
D XCF,PATHOUT	Display what signalling paths are defined and being used by PATHOUT. This includes both device and structures signalling path.
D XCF,GROUP	Display information about multisystem application groups.
D XCF,CLASSDEF	Display information about defined transport classes.
D CF (see note)	Display the coupling facilities connected (or that have been) connected to the system, the CHPIDs and the device status, and the CF space if the CF is connected.
D XCF,CF	Display information about all coupling facilities defined in the CFRM policy.
D XCF,STRUCTURE	Display information about all the structures.
D XCF,POLICY	Display the in-use policy.
Note: The D CF Command is in the table for completeness	

Use the D XCF commands to display the sysplex name, the XCF groups, and the specific member names. In the following example, WTSCPLX3 is the sysplex name from the COUPLExx parmlib member. The JES3 XCF group name shown in Figure 19 is defaulted from the node name on the NJERMT initialization statement:

```
NJERMT,NAME=WTSCPLX3,HOME=YES
```

```
D XCF,SYSPLEX
IXC334I  17.32.04  DISPLAY XCF 202
   SYSPLEX WTSCPLX3:   SC50              SC49
D XCF,GROUP
IXC331I  15.45.46  DISPLAY XCF
   GROUPS(SIZE):   AOFSMGRP(2)       COFVLFNO(6)       DFHIR000(16)
                   EJESEJES(2)       SYSDAE(11)        SYSGRS(6)
                   SYSIGW00(7)       SYSIGW01(7)       SYSIKJBC(6)
                   SYSMCS(9)         SYSMCS2(5)        SYSRMF(6)
                   SYSWLM(6)         WEGELE(1)         WTSCPLX3(4)
                   XCFCONS(6)
D XCF,GROUP,WTSCPLX3
IXC332I  15.46.08  DISPLAY XCF
   GROUP WTSCPLX3:  CI5              CI7              SC50
                    SC49
```

Figure 19. Sample DISPLAY XCF Command Output

The D XCF,GROUP,WTSCPLX3 command lists the active members of the JES3 complex. SC50 and SC49 are the JES3 mains, and CI5 and CI7 are the CI FSS names.

3.8 Planning for Coupling Facility Usage

To use the coupling facility in place of XCF CTCs for JES3 global to local communication, the following steps are necessary:

- Format CFRM couple data set - *IXCL1DSU*

  If a CFRM couple data set does not exist, one must be created and formatted. See 3.8.1.1, “Format CFRM Couple Data Set” on page 48.

- Define CFRM policy - *IXCMIAPU*

  A coupling facility structure must be predefined in the coupling facility resource management (CFRM) policy. See 3.8.1.2, “Define Structure in CFRM Policy” on page 49.

- Define a transport class in the COUPLExx member. See 3.8.1.3, “Modify the COUPLExx Member” on page 49.

  Note: IBM recommends using the XCF default transportation class for JES3 messages.

- Activate the policy See 3.8.1.4, “Activate the CFRM Policy” on page 50.

  SETXCF START,POLICY,POLNAME=CFRM01,TYPE=CFRM

A sample configuration is shown in Figure 20 on page 48. In the CFRM policy, there are two coupling facilities, which are identified as CF01 and CF02. The coupling facilities are in two of the three central electronic complexes (CPCs) of an IBM 9672-E03. The example shows a 6-way JES3 global/local sysplex.

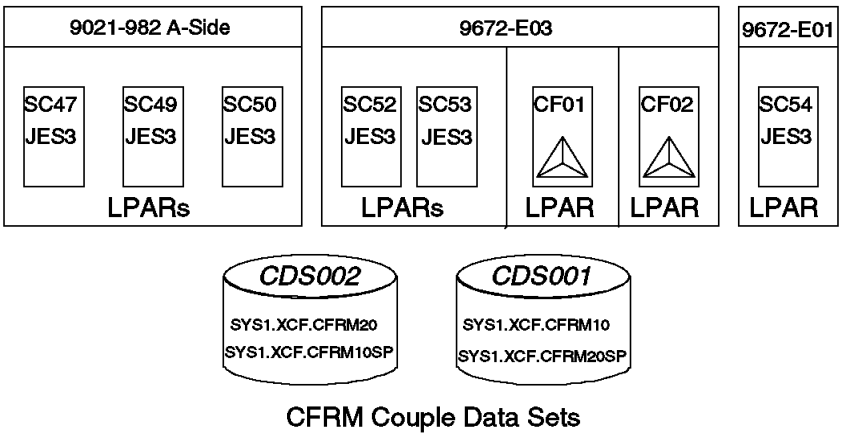


Figure 20. JES3 Sysplex Configuration

3.8.1.1 Format CFRM Couple Data Set

The CFRM couple data set must be defined and formatted. The example in Figure 21 shows the format definitions for two CFRM couple data sets. Each CFRM couple data set, as shown in Figure 20, should have an alternate defined.

```
//STEP1    EXEC PGM=IXCL1DSU
//STEPLIB  DD   DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
     DEFINEDS SYSPLEX(WTSCPLX1)
              DSN(SYS1.XCF.CFRM10) VOLSER(CDS001) MAXSYSTEM(16) CATALOG
          DATA TYPE(CFRM)
               ITEMNAME(POLICY) NUMBER(5)
               ITEMNAME(CF) NUMBER(2)
               ITEMNAME(STR) NUMBER(20)
               ITEMNAME(CONNECT) NUMBER(32)
     DEFINEDS SYSPLEX(WTSCPLX1)
              DSN(SYS1.XCF.CFRM10SP) VOLSER(CDS002) MAXSYSTEM(16) CATALOG
          DATA TYPE(CFRM)
               ITEMNAME(POLICY) NUMBER(5)
               ITEMNAME(CF) NUMBER(2)
               ITEMNAME(STR) NUMBER(20)
               ITEMNAME(CONNECT) NUMBER(32)
     DEFINEDS SYSPLEX(WTSCPLX1)
              DSN(SYS1.XCF.CFRM20) VOLSER(CDS002) MAXSYSTEM(16) CATALOG
          DATA TYPE(CFRM)
               ITEMNAME(POLICY) NUMBER(5)
               ITEMNAME(CF) NUMBER(2)
               ITEMNAME(STR) NUMBER(20)
               ITEMNAME(CONNECT) NUMBER(32)
     DEFINEDS SYSPLEX(WTSCPLX1)
              DSN(SYS1.XCF.CFRM20SP) VOLSER(CDS001) MAXSYSTEM(16) CATALOG
          DATA TYPE(CFRM)
               ITEMNAME(POLICY) NUMBER(5)
               ITEMNAME(CF) NUMBER(2)
               ITEMNAME(STR) NUMBER(20)
               ITEMNAME(CONNECT) NUMBER(32)
```

Figure 21. Format of a CFRM Couple Data Set

3.8.1.2 Define Structure in CFRM Policy

Coupling facility structures are defined through a CFRM policy, which is used to select a coupling facility, as shown in Figure 22.

Note: An installation may have other structures to define besides the JES3 signalling structure.

If a CFRM policy already exists, you could modify it with the JES3 signalling structure and modify the COUPLExx parmlib member to specify a transport class for JES3.

Add the *IXC_JES3* structure definition to the CFRM job, as shown in Figure 22.

```
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
     DATA TYPE(CFRM) REPORT(YES)
  /* DSN(SYS1.XCF.CFRM10) */
     DEFINE POLICY NAME(CFRM01) REPLACE(YES)
        CF NAME(CF01) TYPE(009672) MFG(IBM) PLANT(02) DUMPSPACE(2000)
           SEQUENCE(000000040104) PARTITION(1) CPCID(00)
        CF NAME(CF02) TYPE(009672) MFG(IBM) PLANT(02) DUMPSPACE=(2000)
              SEQUENCE(000000040104) PARTITION(1) CPCID(01)
        STRUCTURE NAME(IXC1_GRS) SIZE(1024)
              PREFLIST(CF01,CF02)
        STRUCTURE NAME(IXC1_DEFAULT)
              SIZE(4096)
        STRUCTURE NAME(IXC_CICS) SIZE(4096)
              PREFLIST(CF02,CF01)
        STRUCTURE NAME(IXC_JES3) SIZE(1024)
               PREFLIST(CF01,CF02)
```

Figure 22. Example of Modifying a CFRM Policy

The STRUCTURE statements in Figure 22 define the name, the size, and the location of the structure, where:

NAME The NAME is the one that is used to access the structure. This name must begin with *IXC* for XCF signalling.

SIZE The SIZE is the size of storage requested for this structure in 1K units.

PREFLIST The PREFLIST specifies a preferred coupling facility to contain this structure.

3.8.1.3 Modify the COUPLExx Member

To segregate the JES3 XCF signalling to a coupling facility list structure, modify the COUPLExx parmlib member for the next IPL.

Figure 23 shows the modified COUPLExx parmlib member with the JES3 signalling paths defined through a coupling facility.

```
/*********************************************************************/
/*  MEMBER=COUPLE00                                                  */
/*********************************************************************/
COUPLE SYSPLEX(WTSCPLX1)
       PCOUPLE(SYS1.XCF.CDS10)
       ACOUPLE(SYS1.XCF.CDS20)
       INTERVAL(30)
       CLEANUP(30)
       MAXMSG(500)
       RETRY(10)
       CLASSLEN(1024)

/* DEFINITIONS FOR CFRM POLICY */
DATA TYPE(CFRM)
     PCOUPLE(SYS1.XCF.CFRM10)
     ACOUPLE(SYS1.XCF.CFRM20)

/* DATA SETS FOR SFM POLICY */
DATA TYPE(SFM)
     PCOUPLE(SYS1.XCF.SFM10)
     ACOUPLE(SYS1.XCF.SFM20)

/* DEFINITIONS FOR JES3 SIGNALLING */
CLASSDEF CLASS(JES3) GROUP(WTSCPLX3)
PATHOUT STRNAME(IXC_JES3) CLASS(JES3)
PATHIN STRNAME(IXC_JES3)

/* DEFAULT CTC PATHS FOR SIGNALLING */
PATHIN  DEVICE(4010,4020,4030,4040,4050,4060,4070,4080)
PATHIN  DEVICE(4018,4028,4038,4048,4058,4068,4078,4088)
PATHOUT DEVICE(5010,5020,5030,5040,5050,5060,5070,5080)
PATHOUT DEVICE(5018,5028,5038,5048,5058,5068,5078,5088)
```

Figure 23. COUPLExx Member of SYS1.PARMLIB

3.8.1.4 Activate the CFRM Policy

The following command is used to activate the CFRM policy:

```
SETXCF  START,POLICY,POLNAME=CFRM01,TYPE=CFRM
```

3.8.1.5 Activate the Signalling Paths

Once the updated CFRM policy is activated, JES3 XCF signalling can be dynamically switched to use the coupling facility structure with the following SETXCF operator commands:

```
SETXCF START,CLASSDEF,CLASS=JES3,GROUP=WTSCPLX3
SETXCF START,PATHOUT,STRNAME=IXC_JES3,CLASS=JES3
SETXCF START,PATHIN,STRNAME=IXC_JES3
```

When these commands complete, JES3 communication between processors is through the coupling facility.

If the coupling facility should fail, XCF automatically switches the JES3 signalling to the DEFAULT transportation class, which is defined through a set of CTC paths and allows JES3 communication between the processors to continue.

Resource Measurement Facility Version 5 (RMF) generates XCF activity reports. These reports include XCF usage by system, XCF usage by member, and XCF path statistics sections. You should use these reports to verify that JES3 XCF signalling performs properly.

3.8.2 JES3 Global Resource Serialization Considerations

The disposition for all data sets defined in the JES3 and JES3CI procedures should have a DISP=SHR specified. This lets the JES3 global main, local mains and CIFSS address spaces start even if the installation has specified in the GRSRNL parmlib member generic inclusion RNLDEF entry for QNAME(SYSDSN).

Note: RACF should be used to control access to the JES3 checkpoint and spool data sets rather than the DISP=OLD specification on the JES3 JCL.

JES3 uses ENQs with QNAME=SYSZIAT to synchronize internal events. Some of these ENQs are at STEP level and are used to serialize activity within the address space among multiple tasks. Others are SYSTEM level and serialize within an MVS image.

During JES3 initialization, IATINTK issues an exclusive ENQ with QNAME=SYSZIAT RNAME=JES3ACTIVE at the SYSTEM level. If you convert this request to global (SYSTEMS) level, then none of the locals will be able to start. Or if a local is active, the global will not start as it cannot obtain the ENQ resource.

JES3 mains and JES3 CIFSSs serialize access to the checkpoint data sets through a RESERVE request. The RESERVE QNAME is SYSZIAT, and the RNAME is the checkpoint data set VOLSER followed by the data set name. The reserve activity against the checkpoint data set is not great, nor is it of long duration.

The IBM recommendation is to have the generic QNAME(SYSZIAT) in the RNL exclusion list.

Chapter 4. JES3 Version 5.1.1 Migration

This chapter summarizes the MVS/ESA JES3 Version 5 migration actions:

- JES3 Version 5.1.1 requires MVS/ESA SP Version 5 Release 1 The JES common coupling services is part of the MVS/ESA SP 5.1 BCP.
- All the JES3 Version 5.1.1 systems in a JES3 complex must be in the same multisystem sysplex. If the JES3 complex consists of only one JES3 instance (such as MAINPROC,CPUID=ONLY), the MVS XCF configuration can specify PLEXCFG=XCFLOCAL.
- JES3 Version 5.1.1 does not coexist in a JES3 complex with any previous JES3 releases.
- Migration from JES3 Releases 3.1.3 and 4.2.1 requires a warm start.
- Fallback to JES3 Releases 3.1.3 and 4.2.1 requires a warm start.
- Migration from JES3 Releases 2.2.1 and 3.1.2 requires a cold start. Dump job (DJ) must be used to carry jobs over the cold start.
- Fallback to JES3 Releases 2.2.1 and 3.1.2 requires a cold start. Dump job (DJ) must be used to carry jobs over the cold start.

Refer also to *MVS/ESA Conversion Notebook for JES3 Version 5*.

4.1 JES3 Version 5.1.1 Initialization Changes

JES3 Version 5.1.1 accepts an unchanged initialization stream created for JES3 Releases 3.1.3 and 4.2.1. Warning messages are issued for the obsolete initialization keywords, and defaults are supplied for the added keywords.

4.1.1 CPUID Removal from Initialization

- MAINPROC initialization statement:
  - The CPUID= and MODEL= keywords are removed and ignored if present. The JES3 system name must match the MVS system name; that is, the MAINPROC,NAME= must be the same as that specified by SYSNAME= in the IEASYSxx member of parmlib. JES3 main processor names “SHORT,” “S,” or “ALL” are not accepted. These names are reserved for JES3 commands.
  - The STXTNT= keyword is removed and ignored if present.

    Note: Initially it is recommended you keep these removed keywords and avoid the maintenance of two initialization decks for the case where a fallback to a previous level of JES3 is required. Message IAT3256 keyword KEYWORD IGNORED, NO LONGER SUPPORTED is issued when any of these keywords are encountered on the MAINPROC statement. JES3 initialization continues after the message.
- DEVICE initialization statement:
  - CTC device numbers on the JUNIT= keyword of the statements with DTYPE=SYSMAIN are ignored if present.
  - JUNIT= and XUNIT= may specify 4-digit device numbers.

- Change all device names that have a slash (/) as the first character. To avoid confusion with device numbers, change also all JNAME= names that consist of only 0-9, A-F characters and are only 3- or 4-characters long.

4.1.2 OPTIONS Statement

- OPTIONS initialization statement:
  - The XCFGRPNM= keyword is added to specify the JESXCF group name for the JES3 complex.

    The nodename of the homenode (NAME= on NJERMT statement, which indicates HOME=YES) is used as the default JESXCF group name JES3 Version 5.1.1. The nodename must conform to the JESXCF group naming conventions (first character alphabetic and remaining characters alphabetic, numeric or the special characters @, #, $). If the node name does not conform to these restrictions, the following can be done:
    - Change the node name (not a very practical solution).
    - Use the XCFGRPNM= keyword to assign a valid JESXCF group name for the JES3 complex.

  See 3.6.2, “JES3 XCF Group Name” on page 42 for an explanation of the rules.

4.1.3 NJERMT Statement

- NJERMT initialization statement:
  - Change all NJERMT,NAME= names that are called “ALL.” To avoid confusion with device numbers, change also all node names that consist of only 0-9, A-F characters and are only 3- or 4-characters long.

4.1.4 4-Digit Device Support

The following is an overview of JES3 Version 5.1.1 changes for 4-digit device support:

- The stage-1 records used by the initialization stream checker (IATUTIS) are changed to accommodate 4-digit device numbers. For compatibility, the 4-byte device number is added to the end and the 3-digit device number is also set whenever possible:

Device Number	Stage-1 Record	
82A	G,128,82A,3800	,082A
182A	G,128,***,3800	,182A

4.1.4.1 Initialization Statement Changes

CONSOLE For MCS-managed consoles, a 3-digit or 4-digit device number may be specified on the UNIT= parameter.

DEVICE (CTC) JES3 Version 5.1.1 does not use CTCs for inter-processor communication; therefore, the specification of CTC device numbers on the JUNIT= parameter is not required. The DEVICE statement with DTYPE=SYSMAIN specified continues to be used to define the initial status of the mains. Initially it is recommended to leave the CTC device numbers on the JUNIT= keyword and thus avoid

maintaining two initialization decks for the case where a fallback to a previous level of JES3 is required. A warning message IAT3184 DEVICE NUMBERS IGNORED FOR DEVICE STATEMENT WITH DTYPE=SYSMAIN is issued when the CTC device numbers are left on the DEVICE statement. JES3 initialization continues after the message.

DEVICE (Network) A 3-digit or 4-digit device number may be specified on the JUNIT= parameter.

DEVICE (I/O) A 3-digit or 4-digit device number may be specified on the JUNIT= and XUNIT= parameters.

DYNALLOC A 3-digit or 4-digit device number may be specified on the UNIT= parameter.

GROUP A 3-digit or 4-digit device number may be specified on the EXRESC= and DEVPOOL= parameters.

JES3LIB A 3-digit or 4-digit device number may be specified on the UNIT= parameter.

RJPLINE A 3-digit or 4-digit device number may be specified on the A= parameter.

4.2 DSP Customization

JES3 user exits and user DSPs may require modifications to support 4-digit device numbers. The following is a list of potential changes:

- References to UCBNAME should be changed to use the UCBDEVN service.
- The UCB services should be used to access UCB information.
- References to control block fields that are modified to support 4-digit device numbers.
- JES3 and PSF exits that reference the device number in JSPADEVA.

User DSPs that process staging areas can be changed to use the checkpoint message service macro (IXZXIXCM) to maintain modifications over a JES3 start.

Most JES3 user exits require a reassembly due to changes to the JES3 mapping macros.

4.3 Operations

The following operator commands have been modified or deleted:

- Status of CTCs that are used to communicate between processors in a JES3 complex cannot be interrogated or modified with JES3 commands.
- The *S,main,REDRIVE and *S,main,SWITCH commands are deleted.
- The *INQUIRY,D,D command has a new parameter SHORT that displays device status only on a specified main.
- The *SEND command has a new parameter ALL that sends MVS commands to all active mains in the complex.
- The *VARY and *MODIFY,V commands recognize ALL as a valid main parameter.

- 4-digit device numbers are displayed in messages and can be entered on commands.

4.4 Automation

As a result of many JES3 message changes, installations have to investigate whether the automation packages have to be updated.

4.5 Accounting

The communication mechanism between the JES3 address space and the user address spaces has been changed from asynchronous (SRB) to synchronous (PC). Some of the SRB CPU usage time that was previously charged to the JES3 address space will now be charged to the user's address space.

SMF type 48 record and SMF type 84 record are changed.

4.6 DFSMS

If the DFSMS/MVS Version 1.2 configuration is defined using sysplex names, JES3 does not provide data set awareness and scheduling services for the SMS data sets. If the SMS configuration is defined using a combination of both system names and system group names, JES3 SMS data sets services are available on those processors whose name matches the system names defined in the SMS configuration. JES3 SMS data set services are available on seven processors if there is more than eight MVS systems in the SMS complex.

4.7 JES3 SSI 54 Support

Subsystem interface function code 54 is now supported by JES3. This function code returns subsystem information when the caller issues the IEFSSREQ macro. The caller is a program or subsystem operating in a different address space other than JES3, and the caller can issue the request from either the global or local processor. The caller may be non-authorized. The SSI support provides a calling program with the following types of data:

- Fixed information about the release, such as:
  - The FMID and version number
- Functional information about the subsystem; for example:
  - Is dynamic output supported?
  - The node name.
- Functional information about the subsystem that the installation supplies:
  - The installation can supply a variable string of information via a new JES3 user exit, IATUX63.

4.7.1 User Exit IATUX63

JES3 user exit IATUX63 may be used to establish installation specific information that is to be returned to the caller of SSI function code 54. IATYUX63 is the user exit parameter list that contains the information sent and received from the user exit.

4.8 ENF Signal 40

JES3 when either initializing or terminating issues ENF 40 signals. The listener is a program or subsystem operating in a different address space than JES3. A listener can listen for ENF 40 from either the global or local processor. Listeners hear only when JES3 on the same processor initializes or terminates. The caller or listener must be authorized.

ENF 40 listening is independent of the SSI 54 support. A caller of SSI 54 should listen for ENF 40 signals that JES3 has initiated or terminated. This allows the caller to reissue SSI 54 calls to obtain potentially any changed subsystem version information.

4.9 Started Job Support

Job support for started tasks allows you to include a job statement with a started tasks (STC). With MVS/ESA Version 5, when you issue the START command, you start a JCL stream that can be either a job or a procedure; so you can now have “started jobs” or “started tasks.”

This support allows you to specify for your STCs the following:

- A JOB statement with a logical subset of parameters (for example, those parameters related to input processing are excluded)
- Job level statements, for example OUTPUT, JCLLIB, SYSIN DD

A new parameter has been added to the START command: *JOBNAME*. This new parameter allows you to specify a job name for the STC or job that you are starting.

```
START procname.identifier,keyword=option,keyword=option....,
                JOBNAME=jobname,SUB=subsystemname
```

Job level characteristics can now be associated with started tasks, such as:

- Job names
- Output control such as a job-level default OUTPUT JCL statement
- Accounting information

  IATUX29 has access to the accounting information through field *ISACCCNT*.

In place of the two card stream for started tasks as in the past, the stream can now contain many JCL images. Previously, JES3 had a limit of 15 JCL statements. JES3 5.1.1 is changed to allow many JCL images in a started task. The JCT flag byte, JCTDSEL, still indicates a demand select job.

Note: JES3 JECL *cannot* be used within the started task input stream.

Approximately 700 JCL images can now be used. The new limit for the number of JCL cards allowed is due to the maximum size of a staging area, which is

60,000 bytes minus the size of the staging area header. If a job is submitted with an input stream that exceeds the staging area size, the following message is issued:

```
IAT1614 SUBMIT FAILED FOR STARTED TASK STCJOB5  REASON=02
IEE485I START FAILED, SUBSYSTEM JES3 HAD AN I/O ERROR
```

4.9.1 Using Started Job Support

Started job support gives more flexibility in managing STCs, and the need for setting up and maintaining different procedures for multiple systems is eliminated. If you need to have the same procedure started on each system with a different name, you can now maintain one member containing the procedure you want to start instead of maintaining different members all containing the same JCL stream. You can start different jobs invoking the same procedure, or you can start the procedure giving it a different job name on each of your systems.

If you want to keep the jobs and procedures separate, you can use the *IEFJOBS* data set. You can update your MSTJCLxx to include the IEFJOBS DD as follows:

```
//MSTJCL00 JOB MSGLEVEL=(1,1),TIME=1440
//         EXEC PGM=IEEMB860,DPRTY=(15,15)
//STCINRDR DD SYSOUT=(A,INTRDR)
//TSOINRDR DD SYSOUT=(A,INTRDR)
//IEFPDSI  DD DSN=SYS1.PROCLIB,DISP=SHR

//IEFJOBS  DD DSN=SYS1.STCJOBS,DISP=SHR

//IEFPARM  DD DSN=SYS1.PARMLIB,DISP=SHR
//SYSUADS  DD DSN=SYS1.UADS,DISP=SHR
//SYSLBC   DD DSN=SYS1.BRODCAST,DISP=SHR
```

You can concatenate several data sets to the IEFJOBS ddname, exactly as you can do for the data sets concatenated to the IEFPDSI ddname.

If you do not want to update MSTJCLxx, you can still take advantage of this support by placing the jobs you want to start in one of the IEFPDSI data sets. However, it is recommended to define the IEFJOBS ddname in MSTJCLxx for several reasons:

- By defining the ddname IEFJOBS, you can keep jobs and procedures separated. This allows jobs and the procedures they invoke to have the same name.
- Existing procedure libraries need not be modified.
- You need not worry about products that ship procedures that are placed in procedure libraries. If you modify a procedure to add the job card and a new version of the procedure is received, you do not lose your modifications.

With the introduction of the new IEFJOBS DD, it is important to note the following rule:

- The *IEFJOBS* DD data sets must contain *only* jobs.
- The *IEFPDSI* DD data sets can contain both jobs and procedures.
- The JES3 procedure libraries must contain *only* procedures.

4.10 Issuing a START Command

When a START command is issued, the first task of the start command processing is to determine whether a job or a procedure is being started. Figure 24 shows the data sets where a procedure can be placed.

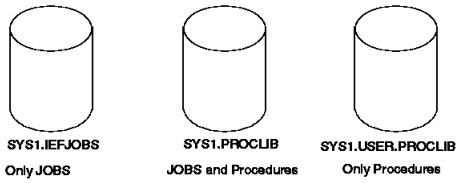


Figure 24. Procedure Libraries

The search of these libraries is as follows:

- The IEFJOBS DD is opened first, if it exists, and searched for the member name specified in the START command, for example:

  ```
  START MYJOB
  ```

  If MYJOB is not present in the IEFJOBS DD, or if this ddname does not exist in the MSTJCLxx, then:

  - The IEFPDSI DD is opened and searched for member MYJOB. If member MYJOB is not present in the IEFPDSI DD, then the JES procedure libraries are searched for a procedure. In this case the JCL image that is created by the system is exactly the same as used in releases earlier than MVS/ESA SP Version 5.

    ```
    //jobname   JOB MSGLEVEL=1
    //stepname  EXEC PROCNAME
    ```

- If member MYJOB is found either in IEFJOBS or in IEFPDSI, its first record is read. If this record contains a valid job card, the remaining records are read and used to build the JCL image that is started. The parameters passed in the START command for symbolic substitution in the JCL are copied to a SET JCL statement placed in the JCL image immediately before the EXEC statement and after the JOB statement.

  ```
  //jobname   JOB  MSGLEVEL=1
  //          SET  SYM1=VALUE1,SYM2=VALUE2
  //stepname  EXEC PROCNAME
  ```

- If the first record of MYJOB member is not a job card, then the start command processing assumes that MYJOB is a procedure.
- After having determined if a started task or a started job was the target of the START, the start command processing determines the job name:
  - If the JOBNAME keyword is not specified on the START command, and the member being started is a JOB, the job name is the one that is specified in the JOB card read from the member.
  - If the JOBNAME keyword is not specified on the START command, and the member being started is a procedure, the job name is the member name, exactly as for releases earlier than MVS/ESA SP Version 5.

- If the JOBNAME keyword is specified on the START command, and the member being started is a job, the job name from the START command will replace the job name read from the job card.
- If the JOBNAME keyword is specified on the START command, and the member being started is a procedure, the job name is the one specified on the JOBNAME keyword of the START command.

- Finally, start command processing passes the JCL image to start task control processing (that actually lets MYJOB execute).

Note: If the member to be started is found in IEFJOBS and it contains a procedure instead of a job, the START command fails with message IEE404I.

4.11 JES3 Version 5.1.1 Changes

This section is a summary of the many changes made in JES3 5.1.1.

4.11.1 Operator Commands

The following operator commands are modified in this version of JES3:

- ***INQUIRY,D,D (CTCs)** - as JES3 no longer manages CTCs for inter-processor communication, the specification of CTC device numbers on this command is no longer accepted.
- ***INQUIRY,D,D,main** - parameter (SHORT) is added allowing an operator to display the status of a device on the specified main.
- ***INQUIRY,D,V=,main** - parameter (SHORT) is added allowing an operator to display the status of a volume on a specified main.
- ***MODIFY,VARY,devname ONLINE/OFFINE,ALL** or ***VARY,devname ONLINE/OFFINE,ALL** - parameter (ALL) is added allowing an operator to change the online or offline status of a device on a specified processor or on all the processors in the JES3 complex to which the device is defined.
- ***MODIFY,VARY,ALL** or ***VARY,ALL** - JES3 does not accept the ALL parameter pertaining to “all devices” on the command. This command is not valid anymore.
- ***MODIFY,VARY,ddd (CTC)** or ***VARY,ddd (CTC)** - as JES3 no longer manages CTCs for inter-processor communication, the specification of CTC device numbers on this command is no longer accepted.
- ***SEND** - parameter (ALL) is added allowing an operator to send a system command to all processors in the JES3 complex.
- ***START,main,REDRIVE/SWITCH** - as JES3 no longer manages CTCs for inter-processor communication, this command has been deleted.
- **A 3-digit** or **4-digit device number** may be specified on the following JES3 operator commands:

```
*CALL,CR,IN=            *CANCEL,TL              *START
*CALL,DC,OUT=           *CANCEL,TP              *START,CNT
*CALL,DJ,IN=            *CANCEL,TT              *START,CR
*CALL,DJ,OUT=           *CANCEL,WTR             *START,DC
*CALL,DISPDJC,OUT       *FAIL                   *START,DJ
*CALL,DISPLAY,OUT       *MODIFY,F D=            *START,TD
*CALL,TD                *MODIFY,S M=            *START,TL
*CALL,TL                *MODIFY,S U=            *START,TP
*CALL,TP                *MODIFY,VARY            *START,TR
*CALL,TR                *MODIFY,W               *START,TT
```

```
*CALL,TT                *INQUIRY,D D=           *START,VARYL
*CANCEL                 *RESTART                *VARY
*CANCEL,CR              *RESTART,WTR
```

4.11.2 Executable Macros

The ILOCUCB service has been replaced with the MVS UCBLOOK service. In previous releases, the ILOCUCB service was used by the JES3 initialization modules to find the address of a UCB for a given the device number or a device name.

4.11.3 Operator Messages

The 4-digit device numbers are displayed in JES3 Version 5.1.1 messages. As a result, over 350 messages have been changed. *MVS/ESA JES3 Messages* provides a detailed list of changed messages in the “Summary of Changes” section.

4.11.4 Mapping Macros

The bulk of the mapping macro changes are to accommodate 4-digit device numbers:

```
IATYAWA     IATYDUM     IATYISD     IATYRSQ
IATYCNC     IATYDYN     IATYJST     IATYSEL
IATYCND     IATYD759    IATYMDS     IATYSET
IATYCNS     IATYEQU     IATYMEM     IATYSTA
IATYCSR     IATYFCT     IATYMLB     IATYSUP
IATYCTC     IATYFSA     IATYMOD     IATYSVT
IATYDDL     IATYFSCB    IATYMPC     IATYSYS
IATYDSP     IATYFSS     IATYMPE     IATYTVT
IATYDSQ     IATYINC     IATYOSD     IATYT35
IATYDJS     IATYINT     IATYPCD     IATYWTR
IATY6FB
```

4.11.5 Installation Exits

JES3 provides one new exit:

IATUX63 JES3 user exit IATUX63 may be used to establish installation specific information that is to be returned to the caller of SSI function code 54.

JES common coupling services provides three new exits:

IXZXIT01 The transport exit allows you to view, modify, add to, or reroute messages prior to the messages being delivered to the receiving JES3 member. This exit gets control for IXZXIXSM (send message) and IXZXIXAC (acknowledge message) macro invocations.

IXZXIT02 The receive exit allows you to view or modify messages before they are retrieved from a mailbox. The exit can also retrieve any message extents that IXZXIT01 might have added. This exit gets control for IXZXIXRM (receive message) macro invocations.

IXZXIT03 The ATTACH/DETACH exit receives control during JES3 initialization and termination.

During JES3 initialization, exit IXZXIT03 enables you to:

- Obtain a storage area during attach processing and make that area available to exits IXZXIT01 and IXZXIT02.
- Invoke the attach macro IXZXIXAT (This is necessary only if performance degradation is experienced when using the IBM-defined attachment).

During JES termination, exit IXZXIT03 enables you to:

- Free the storage obtained during attach processing.
- Invoke the detach macro service IXZXIXDT (This is necessary only if IXZXIT03 issued the IXZXIXAT macro during attach processing).

4.11.6 Diagnostic codes

The following JES3 and MVS diagnostic codes have been added or modified:

DM759 A new dump code for interface problems between JES3 and the JESXCF.

6FB All the reason codes have been changed for this completion code. The ABEND X'6FB' is issued during JES3 processing when the JES3 subsystem communication services (module IATSSCM or IATSSRE) detects an error. See *MVS/ESA System Codes* for details.

DFB New reason codes have been added for this completion code. The ABEND X'DFB' is issued during processing in a functional subsystem (FSS) address space when a JES3 module detected an error. A hexadecimal reason code in register 15 explains the error. See *MVS/ESA System Codes* for details.

4.11.7 SMF Records

The following SMF records have modifications in this release:

SMF43 Record type 43 is written during JES3 initialization and the converter/interpreter functional subsystem (C/I FSS) initialization. This record contains an indicator for the type of JES3 start, JES3 initialization deck origin type and contents, and JES3 procedure name. The change in this record accommodates 4-digit device numbers.

SMF48 JES3 creates SMF48 records to report on remote job processing (RJP) work station activity work. The record is changed to accommodate 4-digit device numbers.

SMF84 JES3 creates SMF84 records to report on the JES3 Monitoring Facility (JMF). The changes accommodate 4-digit device numbers and the JES3 staging area usage for JESXCF.

4.12 Updating DEVICE Initialization Statements

To make initialization changes easier, new ISPF edit macros are added to allow initialization stream DEVICE statement updates when adding main processors to the complex.

4.12.1 IATDEVA and IATDEVC ISPF Edit Macros

Two ISPF edit macros are provided that allow initialization stream DEVICE statements to be updated:

- **IATDEVA**

  For IATDEVA the following rules apply:

  - Both 3-digit and 4-digit device numbers are accepted.
  - A model DEVICE statement must be indicated by a “MOD” line command on the line of the DEVICE statement to use as a model. “A” or “B” line command specifies the destination of the IATDEVA operation.

- The model DEVICE statement is copied for each of the requested devices specified in the list or range of device numbers.
- If the model DEVICE statement contains a device number in the JNAME parameter, the device numbers for the new DEVICE statements being added are substituted in the new DEVICE statement's JNAMEs.
- If the model DEVICE statement does not contain a device number in the JNAME parameter, the device numbers for the new DEVICE statements being added are used as the new DEVICE statement's JNAMEs.
- All other fields are copied directly from the model DEVICE statement.

• **IATDEVC**

For IATDEVC the following rules apply:

- Both 3-digit and 4-digit device numbers are accepted.
- The DEVICE statements identified are modified to reflect all the processors defined by MAINPROC statements, which might include both the addition and deletion of some processors in the DEVICE statements. A warning message is issued to warn the user that by deleting or adding processors, jobs already in the system may be adversely affected when this change takes place.
- The DEVICE statements to be changed are identified either by specifying the list of device numbers on the command line, specifying the device number range on the command line, or by putting characters "CC" on the first and last line of the device statements to be changed.
- If the DESTCODEs for the DEVICE's current processor definitions are sequential (for example, S1, S2, S3,...), the new processor definitions for this DEVICE statement continue this pattern (if we run out of sequential DESTCODEs for any reason, the last DESTCODE used on this DEVICE statement is used for all the remaining processor definitions for this DEVICE statement).
- If ALL the DESTCODEs for the DEVICE's current processor definitions are the same, the new processor's definitions for this DEVICE statement use the same DESTCODE.
- All other parameters for the new processor definitions for this DEVICE statement are be copied from this DEVICE statement's last processor definition.

The edit macros are distributed in the SYS1.SAMPLIB data set, and the ISPF messages for the macros are in the SYS1.SBLSMSG0 data set.

4.12.1.1 IATDEVA Example

Add tape device BB3A:

```
File Edit Confirm Menu Utilities Compilers Test Help
VIEW       SYS1.PARMLIB(JES3INTS)  01.03                    Columns 00001 00080
Command ===> IATDEVA BB3A                                     Scroll ===> HALF
****** ***************************** Top of Data ********************************
                                                      54 Line(s) not Displayed
MOD055 DEVICE,DTYPE=TA33490,                                                  X
000056 JNAME=TAB3A,JUNIT=(B3A,MVS1,S1,OFF,B3A,MVS2,S2,OFF),                   X
A00057 XTYPE=(DI3490,TA),XUNIT=(B3A,MVS1,S1,OFF,B3A,MVS2,S2,OFF)
                                                      2607 Line(s) not Displayed
****** ***************************** Bottom of Data *****************************
```

Result:

```
File Edit Confirm Menu Utilities Compilers Test Help
VIEW       SYS1.PARMLIB(JES3INTS)  01.03                    Columns 00001 00080
Command ===>                                                  Scroll ===> HALF
****** ***************************** Top of Data ********************************
                                                      54 Line(s) not Displayed
000055 DEVICE,DTYPE=TA33490,                                                  X
000056 JNAME=TAB3A,JUNIT=(B3A,MVS1,S1,OFF,B3A,MVS2,S2,OFF),                   X
000057 XTYPE=(DI3490,TA),XUNIT=(B3A,MVS1,S1,OFF,B3A,MVS2,S2,OFF)
000058 DEVICE,DTYPE=TA33490,JNAME=TABB3A,
000059    JUNIT=(BB3A,MVS1,S1,OFF,BB3A,MVS2,S2,OFF),
000060    XTYPE=(DI3490,TA),XUNIT=(BB3A,MVS1,S1,OFF,
000061    BB3A,MVS2,S2,OFF)
                                                      2607 Line(s) not Displayed
****** ***************************** Bottom of Data *****************************
                    IATDV900 Command completed successfully.
```

4.12.1.2 IATDEVC Example

Add main processor MVS3:

```
File Edit Confirm Menu Utilities Compilers Test Help
VIEW       SYS1.PARMLIB(JES3INTS)  01.03                    Columns 00001 00080
Command ===> IATDEVC BB3A                                     Scroll ===> HALF
****** ***************************** Top of Data ********************************
                                                      32 Line(s) not Displayed
000033 MAINPROC,NAME=MVS1,SYSTEM=JES3,.......
000035 MAINPROC,NAME=MVS3,SYSTEM=JES3,.......
000034 MAINPROC,NAME=MVS2,SYSTEM=JES3,.......
                                                      23 Line(s) not Displayed
000059 DEVICE,DTYPE=TA33490,JNAME=TABB3A,
000060    JUNIT=(BB3A,MVS1,S1,OFF,BB3A,MVS2,S2,OFF),
000061    XTYPE=(DI3490,TA),XUNIT=(BB3A,MVS1,S1,OFF,
000062    BB3A,MVS2,S2,OFF)
                                                      2606 Line(s) not Displayed
****** ***************************** Bottom of Data *****************************
```

Result:

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
VIEW       SYS1.PARMLIB(JES3INTS)  01.03                    Columns 00001 00080
Command ===>                                                 Scroll ===> HALF
****** ********************************* Top of Data **********************************
                                                          1 Line(s) not Displayed
=NOTE= IATDV800W WARNING:  ORDER OF THE MAINPROC STATEMENTS MAY HAVE CHANGED.
                                                         31 Line(s) not Displayed
000033 MAINPROC,NAME=MVS1,SYSTEM=JES3,.....
000034 MAINPROC,NAME=MVS3,SYSTEM=JES3,.....
000035 MAINPROC,NAME=MVS2,SYSTEM=JES3,.....
                                                         23 Line(s) not Displayed
000059 DEVICE,DTYPE=TA33490,JNAME=TABB3A,
000060   JUNIT=(BB3A,MVS1,S1,OFF,
000061   BB3A,MVS3,S3,OFF,
000062   BB3A,MVS2,S2,OFF),
000063   XTYPE=(D13490,TA),
000064   XUNIT=(BB3A,MVS1,S1,OFF,
000065   BB3A,MVS3,S3,OFF,
000066   BB3A,MVS2,S2,OFF)
                                                       2606 Line(s) not Displayed
****** ******************************** Bottom of Data ********************************

                        IATDV900 Command completed successfully.
```

Note: When adding new main processors, the sequence of MAINPROC statements must not change across JES3 warm starts. However, new main processors can be added or existing processors can be deleted from the end of the MAINPROC statement sequence. Deleting and adding processors may adversely affect jobs already in the system.

4.12.2 Subgeneric Groups

MVS divides generic device types into subgeneric groups. Subgeneric groups allow MVS allocation to serialize a subset of units within a generic name. As a result, more than one allocation can process the same generic device type, as long as the allocations require different subgeneric groups within that generic.

JES3 devices must also be grouped by using generic and esoteric names. Caution should be exercised to avoid subgeneric splits when adding new devices to the JES3 initialization stream. For example, a subgeneric split is detected when some, but not all, devices in a subgeneric group are defined on DEVICE statements, or devices are defined to JES3 as belonging to one subgeneric group but are defined to MVS as belonging to a different subgeneric group.

JES3 initialization stream checker (IATUTIS) detects logical errors in the DEVICE, HWSNAME, RJPLINE, and SETNAME statements by comparing the JES3 initialization data with the configuration data that is generated by the Hardware Configuration Definition (HCD).

Chapter 5. MCS Sysplex Operations

JES3 console support is merged with MVS console support and improves systems management and automated operations by enabling the sysplex operations features of MVS/ESA.

JES3 5.2.1 improves systems management and automated operations by enabling the sysplex operations features of MVS/ESA. System and subsystems can use these features without special considerations to the alternative console implementations in the previous JES3 environments.

Sysplex-wide operation control is 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 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 the many new functions of MCS console support and the changes made to JES3 in support of the MCS multisystem mode of operation for a sysplex environment.

Enhancements to MCS support for the sysplex environment began in MVS/ESA Version 4. Figure 25 describes the many enhancements for a sysplex that JES3 Version 5.2.1 utilizes.

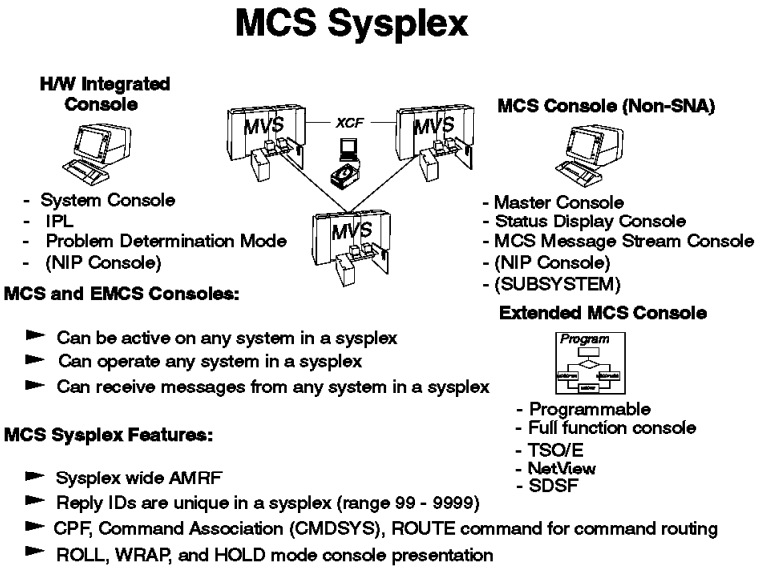


Figure 25. MCS Sysplex Environment

Three types of MCS consoles are shown in Figure 25:

MCS MCS consoles (DIDOCS and SUBSYSTEM) - up to 99 defined through CONSOLxx parmlib member.

EMCS Extended MCS (EMCS) consoles - programmable consoles defined and activated through an intended programming interface (the MCSOPER macro with REQUEST=ACTIVATE).

System System console in problem determination mode - automatically defined by MVS during system initialization.

Consoles can be active on any system in a sysplex and provide sysplex-wide control. MCS uses XCF services for command and message transportation between systems to provide a single system image for the console environment. MCS multisystem support provides:

- Sysplex-wide action message retention facility (ARMF)
- Sysplex-wide unique reply IDs

Command routing is done through:

- ROUTE operator command
- Command prefix facility (CPF)
- CMDSYS setting from the CONSOLE statement in the CONSOLxx parmlib member or the K V operator command.

Message presentation is controlled by a console's ROUTCDE and MSCOPE settings, and the UD (undeliverable message) attribute. SYNCHDEST parameter on the DEFAULT statement in CONSOLxx member defines the alternate console group whose members can receive a synchronous message.

- DIDOCS consoles display messages in either ROLL mode or WRAP mode.
- HOLDMODE specification on the DEFAULT statement of CONSOLxx member controls the temporary suspension or holding screens when in roll, roll-deletable, or wrap mode.

For console switching, consoles may be associated with an alternate console group or an alternate console. When a failing console is switched to an alternate (or when an operator switches a console as a result of the SWITCH command), MCS merges console attributes with those of the alternate. Note that the attributes are added to those of the alternate console and do not replace the existing attributes. Thus, the command authority, message scope, and UD status of the alternate console are not permanently affected by the addition of the failing console's attributes.

5.1 Consoles in a Sysplex

MCS consoles may be attached to any system in a sysplex. The physical attachment does not limit their span of control. A single MCS console can operate any system in the sysplex. An operator can control activity in the sysplex from any console regardless of the type of the console (DIDOCS, extended MCS or system console) or where it is attached. A single master console can control all systems in a sysplex. Multiple consoles may have master authority.

5.1.1 System Console

The system console is automatically defined during MVS initialization. How the system console receives messages after initialization can be controlled by defining values in the CONSOLxx parmlib member. 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 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

The system console should be used only for initialization of MVS and for backup and recovery purposes. The system console should not be used alone to operate the system because system performance can be seriously affected. For normal operation of the system, MCS consoles or extended MCS consoles, or subsystem consoles, like NetView consoles, should be used.

5.1.2 MCS Consoles

The CONSOLxx parmlib member defines an MCS console configuration (up to 99 consoles in a MCS sysplex) and MCS processing options. Some of the CONSOLxx definitions have sysplex scope and are only processed by the first system entering the sysplex:

- RLIM (on the INIT statement)
- AMRF (on the INIT statement)
- CNGRP (on the INIT statement)
- ROUTTIME (on the INIT statement)
- RMAX (on the DEFAULT statement)

MCS consoles are assigned both 4-byte and 1-byte console identifiers. The console IDs are used by command processors to route command responses back to the issuing console. EMCS consoles normally get only the 4-byte console ID unless at activation time a 1-byte migration ID is also 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 when only a 4-byte console ID is available.

5.1.3 Extended MCS Consoles

An extended MCS console (EMCS) is defined through MCSOPER service by the program that uses it. TSO/E, NetView, and SDSF are examples of programs that use EMCS consoles.

Programs can issue commands from an extended MCS console through the MGCRE macro service and include a command and response token (CART), which associates a response with a command. Messages that are queued to EMCS consoles may be selectively retrieved through MCSOPMSG macro service by specifying a CART value or CART mask. Once again, “old” command processors, for example JES3 releases prior to JES3 5.2.1, were not programmed to preserve the CART from the command and return the response through WTO macro service with the CART.

Extended MCS consoles provide flexibility in the number of consoles that can be used in an MVS system. Defining such consoles allows you to increase the number of consoles beyond the MCS limit of 99.

An installation can assign an authorized user of an extended MCS console many of the attributes of MCS consoles, for example, the commands that can be issued and the messages that will be received. The attributes for extended consoles are controlled through the OPERPARM segment in the user's RACF profile.

Currently, extended MCS consoles can be used by:

TSO/E TSO/E users with the appropriate authority 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.

The GETMSG function allows TSO/E users to write REXX programs to handle console output.

NetView Beginning with NetView Version 2 Release 3, the option of communicating with MVS via extended MCS consoles was introduced.

Specifying the appropriate parameters ensures that NetView uses extended MCS consoles on MVS/ESA systems. In addition, individual NetView operators and autotasks can allocate themselves an extended MCS console by issuing the GETCONID command.

Applications Application programs can be authorized to establish extended MCS console sessions, during which commands can be issued and messages (both command responses and unsolicited messages) can be received.

Three assembler macros are needed to use extended consoles:

MCSOPER The MCSOPER macro activates and deactivates extended console sessions. The console authorization is defined either using OPERPARM data from RACF or the OPERPARM parameter list on the MCSOPER macro.

MCSOPER can specify a parameter called MSGDLVRY. This determines whether messages will be delivered to this extended MCS console. If MSGDLVRY=NONE is specified, no messages will be delivered to the console.

MGCRE The MGCRE macro enables the user to issue operator commands.

MCSOPMSG Messages are returned to extended MCS consoles in a Message Data Block (MDB). The program that issues the MCSOPMSG macro must include the MDB mapping macro (IEAVM105) in order to gain access to the MDB fields.

See *MVS/ESA Version 4 Implementation Guide* for a discussion on using extended consoles, including an example of using the three macros described above.

5.2 MCS Command and Message Flow

Commands and messages flow through a system in a similar way as shown in Figure 26.

5.2.1 Command Flow

When an operator command is entered through the MGCRE macro service (see Figure 26), it is first processed by the installation 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. A return code set by the exits indicates whether the command was processed by the exit, MCS should process the command, or the user is not authorized to issue the command.

The command is then broadcast on the Subsystem Interface (SSI) to all active subsystems. Each subsystem, beginning with the primary job entry subsystem, may inspect the command and decide whether to process it. The subsystems make this decision based on the command prefix characters of the command string. For example, by default, NetView looks for a percent sign (%) as the first character. If a subsystem decides that it is 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.

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

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 one does not exist, an error message is issued.

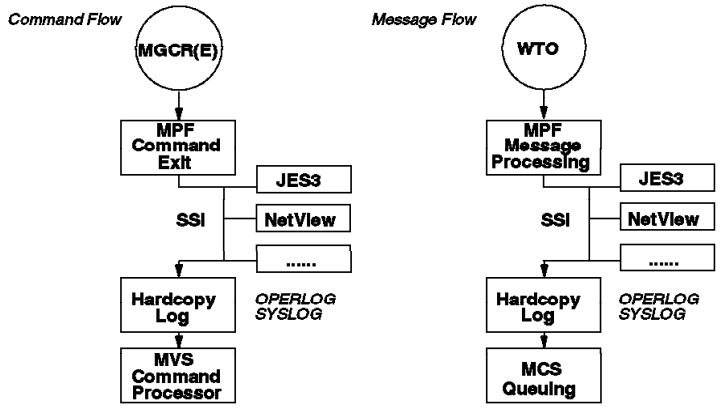


Figure 26. MCS Command and Message Flow

5.2.2 Message Flow

A message is entered to the system either through a WTO (Write to Operator) or a WTOR (Write to Operator with Reply) macro service. The message can be:

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

Solicited Solicited messages are responses to commands issued by an operator.

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

Figure 26 shows a typical message flow within a single system. First, the message is processed by MPF (Message Processing Facility). This processing is based on entries in the MPFLSTxx parmlib member.

MPF allows an installation to influence how WTO and WTOR messages are to be processed. Through the MPFLSTxx member, you can specify four 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. WTOR messages and command responses cannot be suppressed.
- 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, specifically 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.

For full details of MPFLSTxx options and parameters, see *MVS/ESA SP V5 Initialization and Tuning Reference*.

Following MPF processing, the message is broadcast to all active subsystems. The message is presented to each subsystem in turn. Each subsystem may inspect the message and process it as appropriate. A subsystem can alter WQE fields, in which case later subsystems on the SSI will see the changed WQE. A WQE (Write-to-Operator Queue Element) is an internal control block that contains the message text and all related information for that message.

After the message has been inspected by all active subsystems, it is written to the hardcopy log (SYSLOG and/or OPERLOG) unless hardcopy logging is suppressed by an exit.

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 the receiving consoles may not be physically attached to the system where the message was issued.

5.3 Command Routing in a Sysplex

Several console definitions affect command routing in a sysplex. Command routing with JES3 5.2.1 is an important consideration due to the fact that the **T* command no longer can be used to send commands to a main processor. 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)
- The IEECMDPF program in SYS1.SAMPLIB
- The ROUTE command

MCS transports, using XCF services, the command directly to the command processing system, where the command is passed through MPF command user exits, the SSI loop, hardcopy logging, and finally through MVS command processing. This is the same process as would be taken for commands issued locally on that system.

Figure 27 shows command routing in a sysplex.

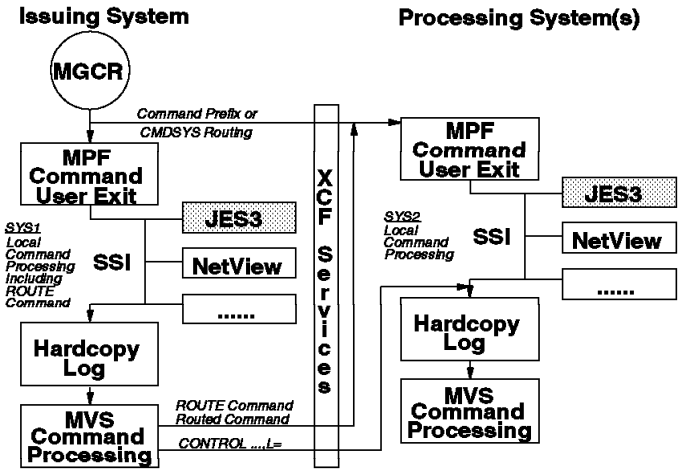


Figure 27. Command Routing in a Sysplex

5.3.1 CMDSYS Parameter and Routing

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 method of allowing a console that is physically attached to one system to be logically associated with another system.

For MCS consoles, CMDSYS is specified in CONSOLxx. For EMCS consoles, it is part of the RACF OPERPARM segment.

Implicit command routing is done when:

- A console physically attached to one system, say SYS1, as shown in Figure 27 on page 73, is logically associated with another system, say SYS2. That is, CMDSYS(SYS2) is in effect. This means that commands issued from this console are to be executed on another system.

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

Note also that there are some commands - LOGON/LOGOFF, TRACK/STOPTR, ROUTE and some variations of CONTROL - that are not affected by the CMDSYS values. These commands are always processed on the issuing system.

5.3.2 Command Prefix Routing

The command prefix facility (CPF) allows subsystems, such as JES3, to register command prefixes with MCS. These prefixes are defined through the CPF macro service and are held in the CPF table, which is propagated to all systems in a sysplex.

CPF allows any operator or any authorized application to enter a command from any system in a sysplex and route that command to the appropriate system for execution. The command responses will come back to the originating console. The application can be an installation exit, a subsystem, or an installation-written program.

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

SYSPLEX The command issued is routed to the system for which the prefix is defined.

SYSTEM The command issued is executed on the system on which the command is entered.

Note: To avoid conflicts with 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*. This is shown in Figure 28 on page 75.

A console physically attached to one system, say SYS1 as shown in Figure 26 on page 71, issues a command using a designated command prefix, indicating that the command is to be executed on another system, say SYS2.

5.3.3 IEECMDPF Program

Another use for the CPF could be through the IEECMDPF program provided in SYS1.SAMPLIB. 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.

The command prefixes in use can be displayed by issuing the MVS operator command DISPLAY OPDATA,PREFIX.

```
D OPDATA,PREFIX
  IEE603I 11.41.01 OPDATA DISPLAY 288
   PREFIX     OWNER      SYSTEM     SCOPE     REMOVE   FAILDSP
   *          JES3       SC50       SYSPLEX   NO       PURGE
   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
```

Figure 28. D OPDATA Operator Command

Note: The * is the JES3 defined CPF prefix in the initialization deck. See 6.7.1, “Defining the JES3 Command Prefix” on page 104.

5.3.4 ROUTE Command

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

The format of the ROUTE command is:

ROUTE sysname,text

Where:

sysname The name of the system that is to process the command

text The command itself

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

5.3.4.1 L= Command 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.

Such commands go through all the normal processing stages on the issuing system - installation command exit processing, SSI broadcasting, logging and execution by an MVS command processor before being transported to the receiving system.

A flag is set to indicate that installation command exit processing and broadcasting on the SSI are to be bypassed on the receiving system. The command is, however, logged on the receiving system and is then processed by the appropriate MVS command processor.

Note that in this case the command is logged on both the issuing and the receiving system.

If the console identified by the L= operand is on a different system than the one that issued the command, the command has to be transported to that system for processing.

The ROUTE command, as shown in Figure 27 on page 73, is processed in two stages; the ROUTE command is processed on the issuing system, and the routed command, which is transported to the receiving system, is processed on the receiving system as a locally issued command.

The installation command exit and the active subsystems on the issuing system see the ROUTE command and its operands. The ROUTE command is logged in the hardcopy log on the issuing system. The ROUTE command processing on the issuing system then causes the routed command to be transported to the receiving system.

The routed command is then treated as though it originated on the receiving system for execution there.

In summary, the full ROUTE command is processed on the issuing system, and the routed command is processed on the receiving system.

5.4 Message Routing in a Sysplex

Figure 29 shows the message flow across systems in a sysplex. The message goes through all the steps of message processing on the issuing system. These steps were shown in Figure 26 on page 71 and 5.2.2, “Message Flow” on page 72.

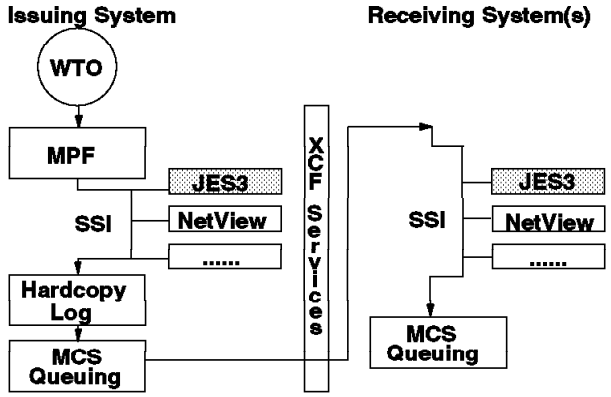


Figure 29. Message Flow in a Sysplex

If there are active consoles receiving this message or active subsystem allocatable consoles on other systems in the sysplex, the message is transported to these systems. On the receiving systems the message goes through the SSI loop, but it is not logged, and finally the message is processed by the message queuing tasks.

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 displayed on consoles with the UD attribute (by default, the master console of the sysplex).

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

Chapter 6. JES3 5.2.1 Sysplex Operations

JES3 operations support is significantly enhanced to exploit the MCS multisystem operations capabilities. JES3 managed operator 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.

Figure 30 describes the new JES3 5.2.1 environment with MCS multisystem support as follows:

MCS and EMCS consoles: 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. As shown in Figure 30 on page 80, 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.

JES3 supported MCS sysplex features: 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 that must execute on a local processor, such as RETURN.

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.

Extended MCS consoles: JES3 exploits MCS facilities. Extended MCS consoles are used to implement JES3 RJP and NJE console support. MCS becomes responsible for queuing messages to all remote consoles.

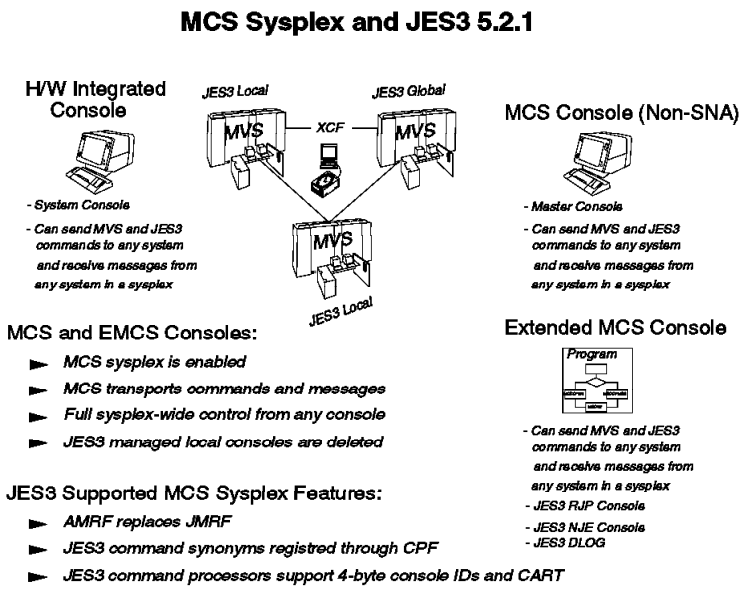


Figure 30. JES3 5.2.1 and MCS Sysplex

6.1 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. For example, these functions and products include:

- MVS operations log
- Display RESERVE message
- ROUTE command
- DB2 use of sysplex command prefixes
- RACF use of sysplex command prefixes
- HCD sysplex ACTIVATE

The following is a list of the major functional changes introduced in the JES3 operations enhancements:

- JES3 operator consoles no longer supported.
- JES3 no longer requires the suppression of the MCS multisystem processing, but instead, relies on the 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.

Currently, a JES3 or non-JES3 (MVS or some other subsystem) command can be sent to a targeted processor using JSERV or SSISERV macro services. The modifier codes used in the MOD= parameter on the JSERV or SSISERV macro are used to request different functions for the specified destination queue. 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.

6.2 JES3 5.2.1 Command Processing

Figure 31 illustrates the command processing flow in JES3 5.2.1. All commands go through the MCS MGCR(E) interface, which serves as the single point of the command entry and all commands are presented to MPF command exits. The MCS MGCR(E) processing invokes the MPF command exits prior to entering the SSI loop for the command processing. JES3 SSI processing (IATSI34) examines input commands to see if they are valid JES3 commands. Valid JES3 command are sent to JES3 through the SSISERV service and further MGCR(E) processing is terminated.

Commands can be entered from the same sources as in prior JES3 releases except for the JES3 managed operator consoles, which are deleted. In addition, support for JES3 command entry through a BDT session is deleted. The JES3 commands authorization exit (IATUX56) for commands entered through BDT is also deleted.

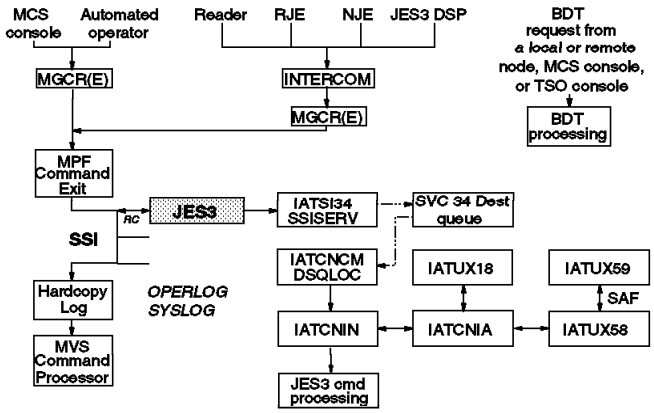


Figure 31. JES3 5.2.1 Command Processing

6.2.1 Command Stacking

The processing for commands entered from an MCS console or an automated operator interface through MGCR(E) remains the same. When multiple commands are entered through a single input from an MCS console, MCS unstacks commands and sends one command at a time to the SSI. When multiple commands are entered through a single MGCR(E) by an automated operator, the whole stack of commands is sent by MCS 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. The JES3 command processing continues to provide the unstacking service as in prior releases of JES3.

6.2.1.1 Command Stacking in CONSOLxx

JES3 no longer uses the delimiter defined by the EDIT keyword in the CONSTD initialization statement to unstack the commands as *newline* has been removed.

```
CONSTD,EDIT=({escape,bkspace,newline,linedl})
```

Instead, it uses the MCS command delimiter that is defined by the CMDDELIM parameter on the INIT statement in the CONSOLxx parmlib member.

```
INIT  CMDDELIM(c)
```

6.2.2 JES3 DSPs and INTERCOM

A JES3 DSP can internally issue a JES3 command through INTERCOM macro (simulate input of operator command). The INTERCOM macro is converted into an MCS MGCRE.

JES3 command processing (IATCNCM) processes staging areas on the SVC 34 destination queue and enters the commands contained in the staging areas for execution. IATCNIN receives control from IATCNCM to perform input command processing.

6.3 JES3 5.2.1 Command Authorization and Exits

JES3 5.2.1 enters all commands, independent of their source, to the system through an internally issued MGCRE macro. The MGCRE processing may invoke one or more MPF command installation exits to affect command processing. The command exits receive control every time a command is entered.

6.3.1.1 Command Authorization with IATUX18

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.

As shown in Figure 31 on page 82, for JES3 commands, IATCNIN calls the JES3 console authorization checking module (IATCNIA) to validate command authority. IATCNIA calls user exit IATUX18. This exit routine allows you to modify a JES3 command and validate the console′s authority to enter the command. If the operator enters a JES3 command at a console that has been defined as not having a high enough authority level for that command, the command is rejected. You could use this exit to allow a particular command to be issued from a console whose definition would reject the command.

Note: IATUX18 no longer sees every command entered from JES3 sources. Installation exit code related to non-JES3 commands entered from JES3 sources must be moved to an MPF command exit.

A new return code (16) is provided which may be used when a command is completely processed within the exit. When this return code is used, JES3 performs no further processing for the command. Previously, JES3 would issue an error message for any unrecognized command processed within the exit.

Changes in the exit interface and return convention will likely necessitate updates to exiting exit code as part of the migration to JES3 5.2.1

Return codes from IATUX18: Indicates how the command is to processed:

0 The command is to be processed; bypass JES3 console authority default checking

4 The command is to be rejected; bypass JES3 console authority default checking

8 The decision is to be made by IBM; use JES3 console authority default checking

12 The decision is to be made by IBM; use JES3 console authority default checking. Also, the exit should be made a dummy exit and should not be called again.

16 The command has been processed; bypass JES3 console authority default checking

The JES3 command authorization process includes invocation to the security authorization facility. 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.

Exit IATUX18 continues to be called for *all* JES3 commands independent of their source. In summary, it is recommended to continue to have all installation JES3 commands processing in MPF command installation exits or in the JES3 command exit, and move all installation non-JES3 commands processed into MPF command installation exits (or other facilities).

6.3.1.2 Command Authorization with SAF/RACF

MCS consoles and MGCRE provide SAF command authorization processing, which became available with MVS/SP Version 3 Release 1.3 when RACF Version 1 Release 9 and JES3 Version 3 Release 1.3 were also installed. MCS uses a SAF-based command authorization scheme for controlling and auditing operator command execution. All operator commands are controlled through SAF and the security product (RACF) on a command by command basis. Of course, this support is optional and must be activated through the use of RACF profiles.

Each MVS and JES3 command has an entity name and an access level associated with it. Each command issuer has an associated UTOKEN (an encapsulation or representation of the security characteristics of the user). RACF uses the UTOKEN of the command issuer, the entity name and the access level of the command to determine if the command issuer is authorized to issue the command.

If SAF command authorization checking is not active, JES3 command authorization checking is used. JES3 command authorization checking uses the authorization level of the console issuing the command and the authorization level required to execute the command to verify whether the command should be executed.

6.3.1.3 Command Authorization Checking with MGCR(E)

MGCR(E) processing allows one or more MPF command exits to affect command processing. You specify on the MPFLSTxx parmlib member the MPF command exits. SET MPF operator command lets you dynamically change the MPFLSTxx member configuration and the exit specifications. Up to six command installation exits can be defined. These exits can:

- Change the text of commands.
- In a sysplex, change the destination of commands by routing them to a different system for execution.
- Modify a console′s authority to use a particular command. That is:
  - Authorize the command from a console that normally would not have the authority to issue the command
  - Reject the command from a console that normally would have the authority to issue the command

- Execute commands.
- Suppress commands.

6.4 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 operator 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.

6.4.1 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.

```
CONSTD,EDIT=({escape,bkspace,,linedel}),
       GLOBMPF={YES|NO},
       SYN={(syn1,...syn6)|8}
       PLEXSYN={(syn1,...syn6)|*},
       CIFSS={FSSDEF|MSGROUTE},
       DLOG={ON|OFF}
```

Figure 32. Global MPF Processing on CONSTD Statement

GLOBMPF=YES Indicates that all messages routed by MCS to the global processor should also be made available to MPF processing on the global processor in addition to MPF processing on the originating system.

GLOBMPF=NO The default. Indicates that messages routed to the global should not be presented to MPF on the global processor. In this case, MPF processing is only possible on the system that a message originates from.

It should be noted 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 will result 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.

6.4.2 Exit 69

Exit 69 is called from the JES3 WTO SSI processing and allows you to examine a message and decide whether it should be sent to the JES3 global address space for additional processing. If the exit indicates that the message should be sent to the global, the WTO SSI sends the message through the JES3 SSISERV service. This exit is called on the system where the message is issued for all messages regardless of the origin of the message (for example: messages that originate on a local, messages that originate on the global, and messages issued by JES3 DSPs through the JES3 MESSAGE macro). In addition to calling exit 69, the WTO SSI processing continues to call IATUX57 to allow the installation to select a single routing code for JES3 message routing processing.

The following is a summary of the types of WTOs and WTORs that are presented to exit 69 and any special considerations that exist for the type of message. Flags in the exit parameter list indicate the type of request being passed to the exit:

Multi-Line WTO The first line of a multi-line WTO (the major line). Subsequent minor lines are *not* passed to the exit. Using the major line, the exit determines whether the message should be sent to the global.

If the exit determines that the message should be sent to the global, the major line and all minor lines are presented to exit 70 on the global. A separate call to exit 70 is made for each minor line.

Commands The text of an operator command.

WTORs The text of a WTOR message.

WTOR responses The full text of a WTOR response.

Exit 69 is managed through the MVS dynamic exit facility. JES3 defines this exit to the MVS dynamic exit facility with the name IAT_EXIT69. By default, JES3 does not define any exit routines to this exit. A sample exit module IATUX69 is provided in the SYS1.AJES3SRC data set.

You can use the EXIT statement of the PROGxx parmlib member, the SETPROG EXIT operator command, or the CSVDYNEX macro to control this exit and its exit routines. JES3 allows multiple exit routines to exist for this exit.

6.4.3 Exit 70

Exit 70 is called in the global JES3 address space when the installation exit 69 has sent a message that requires further processing in the JES3 global address space. This exit cannot influence any routing, presentation or retention attributes of the message.

Exits 69 and 70 allow you to implement JES3-specific processing in the JES3 global address space under CONSERV FCT. Note that the exits are not intended to be a replacement for MVS MCS message exits.

JES3 defines this exit to the MVS dynamic exit facility with the name IAT_EXIT70. By default, JES3 does not define any exit routines to this exit point. You can use the EXIT statement of the PROGxx parmlib member, the SETPROG EXIT operator command, or the CSVDYNEX macro to control this exit and its exit routines. JES3 allows multiple exit routines to exist for this exit. Note that the exit routines are *not* invoked with ASAVE linkage. A sample exit module IATUX70 is provided in the SYS1.AJES3SRC data set.

The pre-JES3 5.2.1 installation exit IATUX31 is deleted. This exit was called from the JES3 global address space prior to the display of a message on JES3-managed consoles and allowed many of the message attributes to be changed.

6.4.4 Functional Message Routing

JES3 functional message routing is currently a way to specify routing information for messages that pertain to functional areas or for messages that pertain to a specific device. For messages issued by JES3 DSPs through the JES3 MESSAGE macro, the routing information is provided directly on the MESSAGE macro. For messages that are not directly issued by JES3 functions the message routing is provided during the JES3 WTO SSI processing based on MSGROUTE initialization statement specifications.

An installation specifies the functional message routing information on the JES3 initialization stream parameters. *S1* and *S2* are JES3 destination classes that are the functional routing for messages issued for device B3A.

```
DEVICE,DTYPE=TA33490,JNAME=TAB3A,JUNIT=(B3A,MVS1,S1,OFF,B3A,MVS2,S2,OFF),
XTYPE=(D13490,TA),XUNIT=(B3A,MVS1,S1,OFF,B3A,MVS2,S2,OFF)
```

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.

6.4.5 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) replaces the function provided by JMRF, and the MCS DISPLAY command must be used to display outstanding messages requiring operator action. AMRF also retains action messages issued when JES3 is not initialized, and action messages issued from programs running under the MASTER subsystem.

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
```

6.4.5.1 Deleted Action Message Commands

Outstanding WTOR requests are always retrievable using the D R command with JES3 5.2.1. Figure 33 shows the deleted JES3 commands and the MCS command equivalents:

Deleted JES3 Command	MCS DISPLAY Equivalent	Differences
*IR - Displays all outstanding action messages including WTORs. SETUP-related messages are not displayed.	DR,L	SETUP-related messages are displayed by the DR command. The DR command also displays messages that were issued before JES3 was fully
*IR,main - Displays all outstanding action messages associated with the named processor. Messages that are issued by a specific JES3 DSP via the JES3MESSAGE macro are not displayed by this command.	DR,L,SYS=main	All messages issued from the named system are displayed by the DR command.
*IR,dspname - Displays all outstanding action messages associated with the named JES3 DSP.	DR,L,KEY=dspname	No major differences.
*IR,SETUP - Displays all outstanding action messages issued by the JES3 Setup DSP (for example, mount messages for a job).	DR,I,KEY=MOUNT	No major content differences. Change in the KEY name from prior releases.
*IR,SETUP,J=jobno - Displays all outstanding action messages issued by the JES3 Setup DSP for a specific job.	DR,I,KEY=MOUNT, JOB=jobname	The DR command uses job name rather than job number.
*IR,SETUP,C=Snn - Displays all outstanding action messages for the designated setup message destination class.	DR,I,ROUT=nnn	The route code equivalent of the JES3 destination class must be used on the display command.
*IR,J=? - Displays the job numbers of the three jobs with the most action messages currently being retained.	none	No equivalent display command
*IR,J=jobno - Displays the number of action messages currently being retained for the specified job	none	No equivalent display command
	DR,U	Return device numbers with unfulfilled mount requests and units requiring intervention are to be displayed.

Figure 33. *I R and DISPLAY R Command Equivalency Summary

The following examples show the commands and message output for some commands shown in Figure 33.

```
 IRR010I  USERID VAINI    IS ASSIGNED TO THIS JOB.
 IAT6100 (JOB06188) JOB VAINITA  (JOB19898), PRTY=01, ID=VAINI
 ICH70001I VAINI    LAST ACCESS AT 09:37:47 ON TUESDAY, MARCH 28, 1995
 IAT4842 LOCATE SUBTASK TERMINATION COMPLETE
 IAT5110 JOB VAINITA  (JOB19898) GET     TIGNORE,SL NOT.REAL.DATA.SET
 IAT5200 JOB VAINITA  (JOB19898) IN SETUP ON MAIN=SC50
*IAT5210 JOB VAINITA  (JOB19898) SC50    MOUNT TIGNORE ON 0B3F,SL,NORING NOT.REAL.DATA.SET

*I S
 IAT5619 ALLOCATION QUEUE    = 00000  BREAKDOWN QUEUE = 00000
 IAT5619 SYSTEM SELECT QUEUE = 00000  ERROR QUEUE     = 00000
 IAT5619 SYSTEM VERIFY QUEUE = 00000  FETCH QUEUE     = 00000
 IAT5619 UNAVAILABLE QUEUE   = 00000  RESTART QUEUE   = 00000
 IAT5619 WAIT VOLUME QUEUE   = 00000  VERIFY QUEUE    = 00001
 IAT5619 ALLOCATION TYPE = AUTO
 IAT5619 CURRENT SETUP DEPTH - ALL PROCESSORS = 00001
 IAT5619 MAIN NAME    STATUS      SDEPTH     DASD        TAPE
 IAT5619 SC50         ONLINE      IPLD 020,001 01048,00000 00080,00000
 IAT5619 SC49         ONLINE      IPLD 020,000 01048,00000 00080,00000

*I S V
 IAT5634 VERIFY      QUEUE
 IAT5636 JOB VAINITA  (JOB19898)      VFYCNT=001

DR,L,KEY=MOUNT
 IEE112I 17.35.50 PENDING REQUESTS 774
RM=7   IM=8     CEM=3     EM=31   RU=0    IR=0    AMRF
ID:R/K    T SYSNAME  JOB ID   MESSAGE TEXT
      29032 I SC50     VAINITA  *IAT5210 JOB VAINITA  (JOB19898) SC50
                                  MOUNT TIGNORE ON 0B3F,SL,NORING
                                NOT.REAL.DATA.SET

DR,L,KEY=MOUNT,JOB=VAINITA
 IEE112I 17.38.04 PENDING REQUESTS 778
```

```
RM=7   IM=8    CEM=3    EM=31   RU=0   IR=0   AMRF
ID:R/K    T SYSNAME  JOB ID   MESSAGE TEXT
     29032 I SC50    VAINITA  *IAT5210 JOB VAINITA  (JOB19898) SC50
                                MOUNT T IGNORE ON 0B3F,SL,NORING
                              NOT.REAL.DATA.SET

DR,I,SYS=SC50
 IEE112I 17.38.33 PENDING REQUESTS 780
RM=0   IM=2    CEM=2    EM=0    RU=0   IR=0   AMRF
ID:R/K    T SYSNAME  JOB ID   MESSAGE TEXT
     29032 I SC50    VAINITA  *IAT5210 JOB VAINITA  (JOB19898) SC50
                                MOUNT T IGNORE ON 0B3F,SL,NORING
                              NOT.REAL.DATA.SET

*IEC501A M 0B3F,IGNORE,SL,,VAINITA,E1,REAL.DATA.SET

DR,I,SYS=SC50
 IEE112I 17.55.18 PENDING REQUESTS 858
RM=0   IM=2    CEM=2    EM=0    RU=1   IR=0   AMRF
ID:R/K    T SYSNAME  JOB ID   MESSAGE TEXT
     30032 I SC50    VAINITA  *IEC501A M 0B3F,IGNORE,SL,,VAINITA,E1,R
                              EAL.DATA.SET
DR,U
 IEE112I 17.58.06 PENDING REQUESTS 881
RM=7   IM=7    CEM=3    EM=31   RU=1   IR=0   AMRF
PENDING UNITS:
 READY UNITS: 0B3F
```

6.5 JES3 5.2.1 and the MVS Operations Log

The MVS system logger is a set of services that allow an application to write, browse, and delete log data. 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. Using the system logger, MVS provides a sysplex-wide consolidated SYSLOG called the OPERLOG or operations log:

- MCS logs the hardcopy message set into the operations log in an MDB format. The SYSLOG has an MVS record format.
- The OPERLOG is maintained by the MVS system logger.

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.

JES3 5.2.1 provides an option to activate a DLOG, where records are written in the JES3 format rather than the MVS format. See 6.6, “JES3 5.2.1 DLOG” on page 97.

6.5.1 MVS System Logger

The MVS system logger executes in its own MVS address space and provides a set of MVS 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

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.

6.5.1.1 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 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)
```

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

In your CFRM policy, you must provide 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 and a shared log stream structure.

```
DATA TYPE(CFRM) REPORT(YES)
DEFINE POLICY NAME(CFRM05) REPLACE(YES)
     ::: other structure definitions :::
     STRUCTURE NAME(SYSTEM_OPERLOG)
            SIZE(1024)
            PREFLIST(CF02,CF01)
     STRUCTURE NAME(SYSTEM_LOGSTREAM)
            SIZE(16128)
            PREFLIST(CF01,CF02)
     ::: other structure definitions :::
```

Once the definitions are in place, use the following command to activate the new policy:

```
SETXCF  START,POLICY,TYPE=CFRM,POLNAME=CFRM05
```

6.5.2 Defining the MVS Operations Log

The OPERLOG is operationally independent of the SYSLOG; that is, an installation can choose to run with either or both. A failure of the OPERLOG when the OPERLOG is active and SYSLOG (or the hardcopy message log) is not active causes SYSLOG to be activated automatically in an effort to reduce the impact to the operations and problem determination tasks. The SYSLOG records are written in the MVS format. The OPERLOG can be defined in the CONSOLxx parmlib member as follows.

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

Figure 34. Defining OPERLOG in SYS1.PARMLIB

If OPERLOG is not defined in parmlib, the operator can activate the OPERLOG with the following command, providing the MVS system logger is activated:

```
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 effect 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.

6.5.2.1 OPERLOG Messages

The messages that are logged in an OPERLOG are equivalent, so far as it is practical, to those that appear in the aggregate of the SYSLOGs for the same systems. Messages in an OPERLOG come from two sources:

1. The hardcopy message set, as provided to an extended MCS console that has the HARDCOPY=YES attribute.

   The hardcopy message set includes messages with one or more of the following characteristics:

   - Have the "hardcopy only" message delivery attribute
   - Are WTOR messages
   - Have descriptor codes of 1, 2, 3, 11, or 12
   - Have no routing codes
   - Have an installation-specified routing code
   - Are command responses of the installation's specified command level
   - Have a message type specified

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

2. Messages created as a result of issuing the WTL macro.

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.

6.5.2.2 OPERLOG EMCS Console

The OPERLOG function internally activates an extended MCS console to receive messages for transcription to the log. 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
```

The response shows six systems in the sysplex.

6.5.3 Defining the Operations Log

You use the IXCMIAPU utility to create or update the LOGR inventory couple data set log stream definitions. Figure 35 on page 94 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 35. Output from the IXCMIAPU Utility for the Operations Log

6.5.3.1 Operations Log Stream Definitions

As shown in Figure 35, the following definitions and rules apply to the operations log definitions:

- The log stream NAME must be SYSPLEX.OPERLOG for the 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.

  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 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.
- 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. See 6.5.8, “Operations Log Stream Data Analysis” on page 96.
- 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 data set names format for the OPERLOG log stream data sets is:

```
hlq.SYSPLEX.OPERLOG.Annnnnnn
-  where nnnnnnn is a running sequence number
```

Note: System logger maintains for each log stream an inventory of logs stream DASD data sets in the LOGR inventory couple data set. This inventory can hold only a limited number of logs stream DASD data sets for each log stream. It is strongly advised to use the IEAMDGLG sample program 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*.

6.5.4 Starting the LOGR Subsystem

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)

- Use the SETSSI command to add the subsystem dynamically:

  SETSSI ADD,SUBNAME=LOGR,INITRTN=IXGSSINT

6.5.5 Activating the LOGR Subsystem

Once the OPERLOG is defined into the LOGR inventory couple data sets, 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 the LOGR inventory couple data set.

The OPERLOG status on each system can be displayed with the D C,HC operator command. The VARY OPERLOG,HARDCPY command activates and the VARY OPERLOG,HARDCPY,OFF command deactivates the OPERLOG on a system.

```
D C,HC
 IEE889I 20.59.14 CONSOLE DISPLAY 149
 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
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 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.

6.5.6 IEAMDGLG 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 services of the MVS system logger to retrieve and delete records from the OPERLOG stream. It reads the records created in a given time span, converts them from Message Data Block (MDB) format to Hard-copy Log format (HCL or SYSLOG), and writes the SYSLOG-format records to a file. It also has an option to delete from the stream all the records created prior to a given date.

6.5.7 Sample OPERLOG Program

Appendix C, “Sample ISPF Dialog to Interface Operlog” on page 161, shows a sample program that reads records from an OPERLOG log stream, converts them to SYSLOG format, and passes the records to ISPF browse for online viewing. It also accepts operator commands from the browse panel and attempts to execute them using the TSO CONSOLE command. The program is based on the IEAMBDLG sample program found in SYS1.SAMPLIB.

The ISPF panel to request the OPERLOG data is shown in Figure 36.

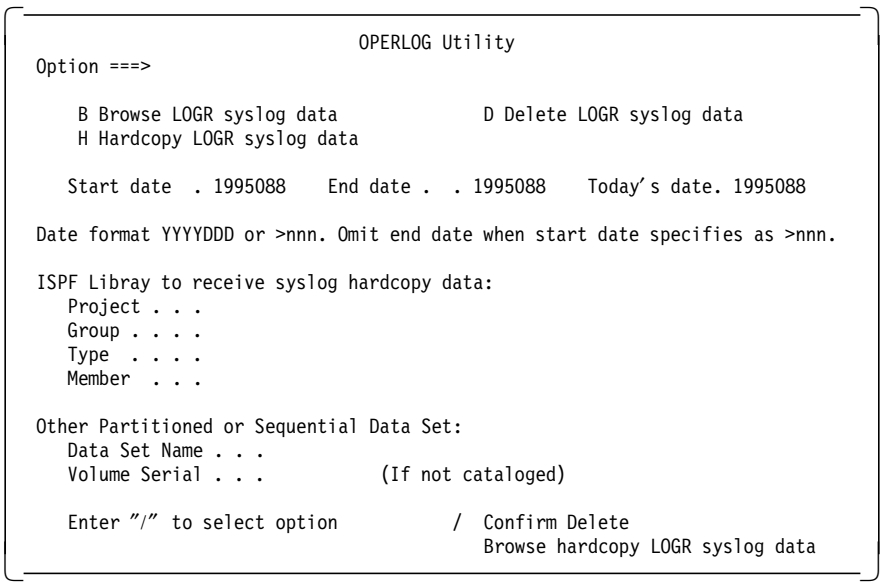


Figure 36. ISPF Panel to Browse the OPERLOG

6.5.8 Operations Log Stream Data Analysis

To determine the MDB size in the operation log stream, data was extracted from the OPERLOG log stream using the LOGR subsystem data set interface. You can use the LOGR subsystem for existing eligible applications that need to access log stream data in data set format. The JCL for the MDB extract job was:

```
//VAINIJ JOB (999,POK),EXPERT,MSGLEVEL=1,MSGCLASS=X,NOTIFY=&SYSUID,
//  CLASS=A
//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
```

From the IEBGENER output, the average MDB size was 338.21 bytes. The MDB size distribution was:

```
200> <=300> <=400> <=500> <=600> <=700> <=800> <=900> <=1000> <=1100> <1200> <=1300> <1400>
 78998 65102 32308 1001   478   142   359   114   156     95     82     59     997

Maximum MDB size 60333          Minimum MDB size 234          Number of samples 179891
```

Given the above definitions, each OPERLOG log stream DASD data set holds roughly 6000 MDBs, which is an average of 11,700 *SYSLOG format lines* as they are generated by the IEAMDBLG program.

The IEBGENER utility program SYSUT1 DD-statement in the previous JCL allocates the OPERLOG log stream through the subsystem LOGR.

6.6 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 -S008= VAINI    DT
743587  S008= VAINI    IEE136I LOCAL TIME=17.43.58 DATE=1994.271 GMT TIME=21.43.58 DATE=
743587  S008= VAINI    1994.271
```

as opposed to the MVS format:

```
NC0000000 SC47    94271 17:39:05.03 VAINI    00000290  DT
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.

6.6.1 Operations Log and JES3 DLOG

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 37. 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).

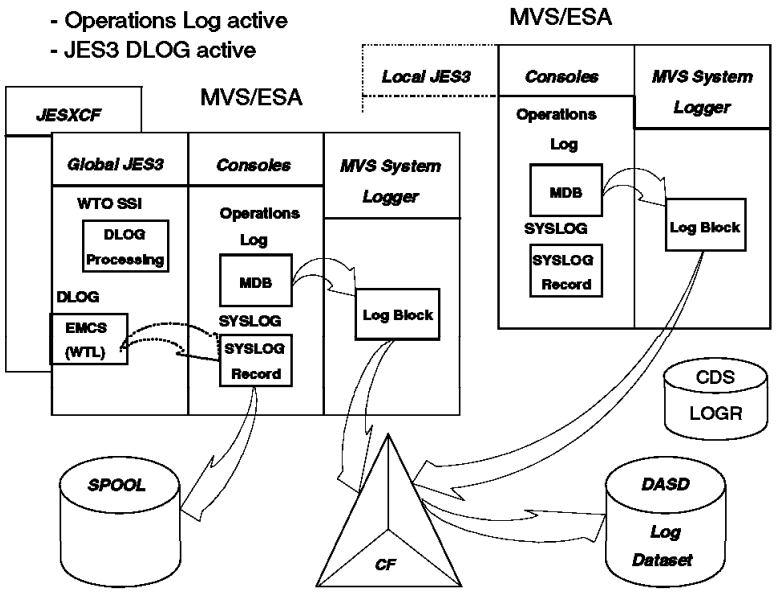


Figure 37. JES3 5.2.1 and MVS Operation Log

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.

6.6.2 Defining the JES3 DLOG

Prior to JES3 5.2.1, JES3 initialization assigns SYSLOG unconditionally to be the MVS hardcopy medium (VARY SYSLOG,HARDCPY) on all systems. If a hardcopy log to a device is required (MLOG is requested in the initialization stream), it is written by JES3 to a JES3 managed device. In addition, if DLOG is specified in the initialization stream, and the JES3 global is restarting without IPL, JES3 ensures that the log is active on the global processor (WRITELOG START). During local connect processing, JES3 causes 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 continues to ensure that SYSLOG is assigned the hardcopy media and that a SYSLOG task in the console address space is active.

The initial state for DLOG is defined on the CONSTD initialization statement with the parameter DLOG=ON or DLOG=OFF, as shown in Figure 38. The old CONSTD keyword HARDCOPY= is deleted.

```
CONSTD,EDIT=({escape,bkspace,,linedel}),
        GLOBMPF={YES|NO},
        SYN={(syn1,...syn6)|8}
        PLEXSYN={(syn1,...syn6)|*},
        CIFSS={FSSDEF|MSGROUTE},
        DLOG={ON|OFF}
```

Figure 38. DLOG Specification on the CONSTD Statement

6.6.3 Activating the JES3 DLOG

The *I O,DLOG displays the current DLOG status, and *F O,DLOG=ON or OFF activates or deactivates the DLOG function.

```
*I O,DLOG
 IAT8617 DLOG STATUS - ACTIVE
```

If the SYSLOG is activated as a result of the DLOG initialization stream specification, no attempt is made to deactivate it when DLOG is deactivated. SYSLOG remains active and MVS format log records are inserted into it.

6.6.4 JES3 DLOG Extended MCS Console

The JES3 DLOG runs in its own address space, as shown in Figure 39. 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. Figure 39 shows console messages for a DLOG activation/deactivation.

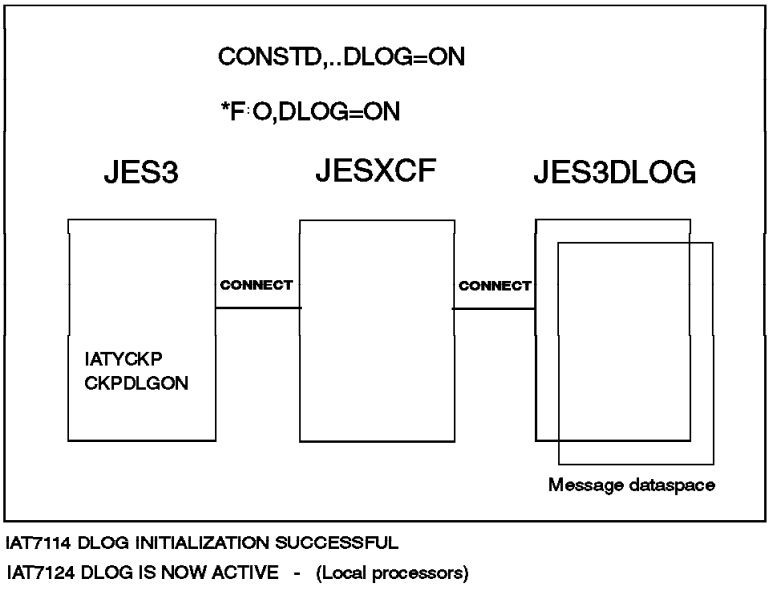


Figure 39. Address Spaces for JES3, JESXCF, and DLOG

The following operator commands show the deactivation and activation of the JES3 DLOG.

```
D C,HC
 IEE889I 12.53.43 CONSOLE DISPLAY 603
 MSG: CURR=11   LIM=1500 RPLY:CURR=7    LIM=999  SYS=SC50      PFK=00
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS ---------------
  SYSLOG                 COND=H      AUTH=CMDS         NBUF=1    UD=Y
                         ROUTCDE=ALL
  OPERLOG                COND=H      AUTH=CMDS         NBUF=N/A  UD=Y
                         ROUTCDE=ALL
```

```
*F O DLOG=OFF
 IEA631I  OPERATOR SYSJ3D01 NOW INACTIVE, SYSTEM=SC50    , LU=SYSJ3D01
 IXZ0002I CONNECTION TO JESXCF COMPONENT DISABLED,
          GROUP WTSCPLX3 MEMBER JES3DLOG
 IEF404I IEESYSAS - ENDED - TIME=12.53.29
 IAT8020 DLOG FACILITY DISABLED
```

```
*F O DLOG=ON
 IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS USED
         TO START JES3DLOG WITH JOBNAME IEESYSAS.
 IEF196I         1 //IEESYSAS JOB MSGLEVEL=1
 IEF196I         2 //JES3DLOG EXEC IEESYSAS,PROG=IATCNDTK
 IEF196I  STMT NO. MESSAGE
 IEF196I         2 IEFC001I PROCEDURE IEESYSAS WAS EXPANDED USING SYSTEM
 IEF196I LIBRARY SYS1.PROCLIB
 IEF196I         3 XXIEESYSAS PROC PROG=IEFBR14
 IEF196I         4 XXIEFPROC  EXEC PGM=&PROG
 IEF196I           XX* THE IEESYSAS PROCEDURE IS SPECIFIED IN THE
 IEF196I           XX* PARAMETER LIST TO IEEMB881 BY MVS COMPONENTS
 IEF196I           XX* STARTING FULL FUNCTION SYSTEM ADDRESS SPACES.
 IEF196I           IEFC653I SUBSTITUTION JCL - PGM=IATCNDTK
 IEF403I IEESYSAS - STARTED - TIME=12.53.57
 IEA630I  OPERATOR SYSJ3D01 NOW ACTIVE,   SYSTEM=SC50    , LU=SYSJ3D01
 IAT7114 DLOG INITIALIZATION SUCCESSFUL
 IAT8020 DLOG FACILITY ENABLED
 IXZ0001I CONNECTION TO JESXCF COMPONENT ESTABLISHED,
          GROUP WTSCPLX3 MEMBER JES3DLOG
```

```
D C,CN=SYSJ3D01
 IEE889I 13.34.51 CONSOLE DISPLAY 764
 MSG: CURR=10   LIM=1500 RPLY:CURR=6    LIM=999  SYS=SC50      PFK=00
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS --------------
  SYSJ3D01          ---  COND=A      AUTH=MASTER       UD=N
   WTSCPLX3              MFORM=M         LEVEL=ALL
   SC50                  ROUTCDE=NONE
                         CMDSYS=SC50
                         MSCOPE=SC50,SC49
```

6.6.5 JES3 5.2.1 NJE Console

JES3 5.2.1 uses internally created extended MCS consoles for NJE console implementation. 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.

The NJE extended MCS console name is SYSJ3Nxx with a KEY=xcfgroupname. The xx is a generated number that starts with 01. The NJE console is created by JESXCF and is used for network commands and command responses on the JES3 homenode node. For example, suppose a command, CP SM RSCS CMD WTSCPLX3 *I Q, is sent from VM to JES3. When the command is received at node WTSCPLX3, it is issued with an INTERCOM. The CNDB (the console destination block or CNDB encapsulates console routing information) for the INTERCOM has a unique CART value assigned identifying the command instance and SYSJ3N01 as the console. Later, the CART value is used to filter the command response from the SYSJ3N01 console's message queue. Once the response is retrieved from the console, it is sent back through the network to the remote node.

DSI processing switches the console name from SYSJ3N01 to the next available console name (e.g. SYSJ3N02).

6.6.6 JES3 5.2.1 RJP Consoles

JES3 5.2.1 internally creates an extended MCS console for RJP console implementation. The RJP extended MCS console name is SYSJ3Rxx with a KEY=xcfgroupname. 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 or is associated with a location where a group of print devices reside.

The extended MCS console named SYSJ3Rxx, where xx is a number starting with 01, is activated by JESXCF. This console is used to deliver messages to *all* RJP consoles. JES3 messages issued in response to commands entered from RJP consoles are directed to the SYSJ3R01 console. JES3 retrieves messages queued on the SYSJ3R01 console for a specific RJP workstation by using the RJP terminal name as the CART value and delivers these messages to the remote console.

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 SYSJ3R01 console, not the actual remote work station.

6.6.6.1 RJP Console Initialization

The CONSOLE statement is only used in JES3 5.2.1 for defining RJP consoles. The RJP CONSOLE initialization statement is changed:

```
CONSOLE,JNAME=name,TYPE=RJP,DEST=(msgdest,...msgdest),LL=nnn,
        LEVEL=nn,SAVEMSG={YES|NO}
```

The new parameters and changes are:

SAVEMSG The SAVEMSG= parameter specifies whether messages should be received and spooled while the console is logged off.

DEST DEST= now accepts MVS routing codes in addition to JES3 destination classes.

The DEPTH= parameter is deleted and is ignored if specified.

6.6.6.2 RJP Console Commands

With JES3 5.2.1, the following commands may be used to affect message queueing to RJP consoles:

***SWITCH RJP01,RJP02** Redirects message traffic from one RJP console to another. This command must be issued from an operator (non-RJP) console. Only a single level of switch is allowed.

***SWITCH RJP01,NONE** Stops message queueing to the target RJP console. This command must be issued from an operator (non-RJP) console.

***FREE RJP01** Frees all messages currently queued to the target RJP console.

***FREE** Frees all messages currently queued to the RJP console from which the command is entered.

The following commands apply only to RJP consoles and the DLOG function:

***I O** command displays RJP console and DLOG status
***F O** command modifies RJP console and DLOG status

The **Z* command to send a message to another RJP console or to a message destination is still supported.

To display the number of spooled messages queued to a signed off RJP console, enter:

```
*I D,T=RJP01
  IAT8618 .. gives  SPOOLED MSG=xx
```

For signed on workstations, this command displays the number of messages queued in JES3 storage for the workstation.

```
*I D,T=RJP02
  IAT8622 .. gives  CHAINED MSG=xx
```

To display a RJP console status:

```
*I O=*
 IAT8589 CONSOLE DISPLAY
 NAME      COUNT    SWITCH      LL     AUTH    SAVEMSG
 RJP01    00000010             0120     15       YES
   ROUTE CODE=(BROADCAST)
   DEST CLASS=(ALL)
   SWITCHED CONSOLES=(RJP06)
```

When displaying console RJP06 whose messages have been switched to console RJP01, RJP01 console status is also displayed:

```
*I O=RJP06
 IAT8589 CONSOLE DISPLAY
 NAME      COUNT    SWITCH      LL     AUTH    SAVEMSG
 RJP06    00000000   RJP01     0120     15       YES
   ROUTE CODE=(HARDCOPY,1-128)
   DEST CLASS=(TOTAL)
 RJP01    00000010             0120     15       YES
   ROUTE CODE=(BROADCAST)
   DEST CLASS=(ALL)
```

6.7 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. See 5.3, “Command Routing in a Sysplex” on page 73 for command routing considerations. JES3 registers the SYSPLEX (PLEXSYN=) and SYSTEM (SYN=) scope command prefixes that are defined on the JES3 CONSTD initialization statement, as shown in Figure 40.

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*.

6.7.1 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 40. 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.

6.7.2 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.

6.7.3 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.

The MVS “D OPDATA” command displays currently registered CPF prefixes:

```
D OPDATA
 IEE603I 22.08.03 OPDATA DISPLAY 444
          PREFIX     OWNER      SYSTEM     SCOPE     REMOVE   FAILDSP
          *          JES3       SC50       SYSPLEX   NO       PURGE
          SC49       IEECMDPF   SC49       SYSPLEX   YES      SYSPURGE
          SC50       IEECMDPF   SC50       SYSPLEX   YES      SYSPURGE
```

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 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
```

6.8 JES3 5.2.1 Enhanced Console Processing User Considerations

The console destination block (CNDB) is a control block that encapsulates console routing information for a command or a message. JES3 5.2.1 intercepts JES3 commands from the SSI and saves the command origin including 4-byte console ID and related response processing information into a CNDB. CNDBs are used in MESSAGE macros to direct messages to a specific destination. CNDBs are also used in INTERCOM macros to identify the origin of an internally issued command. JES3 5.2.1 *INQUIRY and *MODIFY command processing flags the CNDBs to indicate that all messages in response to the command are to be marked as “command responses.” When the JES3 MESSAGE to WTO Converter detects from the CNDB that a message is in response to a command, the DESC=5 parameter is included on the invocation of the WTO macro.

MCS uses 4-byte console IDs to identify a message receiver or command issuer. JES3 5.2.1 is also enhanced to support the 4-byte MCS console IDs.

MCS sysplex assigns symbolic console names for all MCS consoles. JES3 5.2.1 supports MCS console names in the command processing that are carried in the CNDB throughout the command processing.

JES3 5.2.1 continues to define console names for remote consoles through the JNAME keyword in the CONSOLE initialization statement.

JES3 5.2.1 supports the CART to insure accurate presentation of command responses to the extended MCS console. The CART is carried in the CNDB throughout the command processing.

6.8.1 Console Destination Block

The console destination block (CNDB - mapped by IATYCNDB) is a control block that maps and reserves space for origin information for a command, or destination information for a message. Only JES3 console support can directly reference fields in this control block. DSPs which have to copy or update fields in the CNDB should use the IATXCNDB macro service. The IATXCNDB macro can be used both inside and outside the JES3 address space. Most JES3 control blocks that contain console destination information are changed in JES3 5.2.1 to include a CNDB to hold the information.

The IATXCNDB macro allows you to:

- Initialize a console destination block by supplying the console name, console identifier, routing code mask, command and response token (CART), and command indicator.
- Transfer a copy of the specified console destination block to a specified area.
- Update the console identifier, routing code mask, command and response token (CART), and command indicator of the specified console destination block.
- Set to 0 (reset) the console identifier, routing code mask, command and response token (CART), and command indicator of the specified console destination block.
- Verify that the specified console destination block is at the correct level for the current JES3 release.
- Extract the console identifier, routing code mask, command and response token (CART), or command indicator of the specified console destination block.

JES3 5.2.1, during the MCS command processing SSI phase, intercepts JES3 commands, saves the command origin including 4-byte console ID and related response processing information into a CNDB, and finally routes (SSISERV) the command and the CNDB to the JES3 address space.

6.8.2 MESSAGE Macro Changes

The JES3 MESSAGE macro is used by DSPs to issue operator messages (action messages, command responses, and informational messages). In JES3 5.2.1, the following changes are made to the MESSAGE macro parameters:

CNDB= This parameter is added to specify the CNDB, which contains routing and destination information for the message. The CNDB also contains an indicator as to whether the message being issued is a command response and the CART of the command.

MLWOLST= This parameter is added and specifies the list of a multi-line message list (IATYMLWO).

ROUTCDE= This parameter is added and specifies the routing codes to be assigned to the message.

CONS= This parameter accepts a 4-byte console ID.

ROUT= This parameter may be specified at all times. Prior to JES3 5.2.1, ROUT= is only valid during JES3 initialization.

CLASS= This parameter may be specified at all times. Prior to JES3 5.2.1, CLASS= is not valid until JES3 initialization is complete.

BUSY= Certain BUSY return conditions are obsolete because of the deletion of JES3-managed consoles and JMRF.

PRTY= This parameter no longer affects the queuing of messages.

Note: The MPF=, DOMID=, and SYSNAME= parameters are deleted.

6.8.3 JES3 Multi-Line Messages

JES3 5.2.1 DSPs can issue true multi-line WTOs rather than a series of single line messages. Multi-line WTOs keep message lines together for display and provide definitive end line indication for a message.

Issuing a multi-line message in JES3 is a multi-step process:

1. Use a series of IATXMLWO REQUEST=BUILD,TEXT=text,LINETYPE=C|L|D,TOKEN=tkn to construct a multi-line message line-by-line.

```
IATXMLWO ...,TOKEN=0
ST   R1,MSG
IATXMLWO ...,TOKEN=MSG
IATXMLWO ...,TOKEN=MSG
IATXMLWO ...,TOKEN=MSG
```

2. Use MESSAGE MLWOLST= to issue the message:

```
MESSAGE MLWOLST=MSG,...
```

3. Use IATXMLWO REQUEST=CLEANUP to free resources associated with the message.

   The IATXMLWO multi-line WTO message setup macro has two functions for multi-line WTO message processing:

   - Building a single line of a multi-line WTO message - an IATXMLWO macro invocation creates a IATYMLWO token that represents one line of a multi-line WTO message. These tokens are chained together. Once the message is complete, it is issued with the MESSAGE MLWOLST= macro. The MESSAGE macro processing converts the message text elements into a multi-line WTO.

     Each IATYMLWO token is chained to the next IATYMLWO token in the chain for the multi-line WTO message. This is done by passing in the IATYMLWO token address of the first line of the multi-line WTO message through the TOKEN parameter. The TOKEN parameter must be zero for the first line of the multi-line WTO message. The address of the first IATYMLWO in the chain must be passed to the MESSAGE macro through the MLWOLST parameter to issue the multi-line WTO message. The number of lines in a multi-line WTO message is limited to 999.

   - Cleaning up a multi-line WTO message - the IATXMLWO REQUEST=CLEANUP frees the storage of all the IATYMLWO tokens in a chain for a multi-line WTO message. The IATYMLWO token address of the first line of the multi-line WTO message is passed in via the TOKEN parameter.

6.8.4 INTERCOM Macro Changes

The JES3 INTERCOM macro simulates operator input of a console command. The INTERCOM service is used by DSPs to internally issue commands. The INTERCOM macro processing enters the command into the system with an MCS MGCRE macro. As a result, all commands go through the MCS interface, which serves as the single command entry into the MCS sysplex.

In JES3 5.2.1, the INTERCOM macro continues to be the primary means for JES3 DSPs to enter commands into the MCS sysplex. Some changes have been made to the INTERCOM macro:

- An optional CNDB= parameter is added to the INTERCOM macro. It addresses a CNDB that contains a 4-byte console ID, a console name, and a CART of the command issuer. If the CNDB= parameter is not specified, the INTERCOM macro uses the default dummy CNDB in the TVTX.
- Since the console ID is included in the CNDB, the CONS= parameter is deleted. The BUFFER= parameter is also deleted.

The MCS SVC 34 processing invokes the command exits prior to issuing the SSI call for the command processing. If a command is issued through MGCRE with the user data indicating to bypass the authority check (originally specify CHK=NO in the INTERCOM macro), this indicator is passed to the JES3 SSI command processing and is carried over to JES3 command processing to indicate “bypass authority checking.”

See Appendix A, “Sample Program Using JESXCF Services” on page 153, for a sample DSP, which shows both the old and new format INTERCOM and MESSAGE macros.

6.8.5 DEQMSG Macro Changes

The DEQMSG macro has been changed to support the following requests:

- Remove a specific input buffer for an FCT
- Remove all input buffers for an FCT
- Delete a specific action message for an FCT
- Delete all action messages for an FCT
- Remove all input buffers and delete all action messages for an FCT

Note: The TYPE=CONSOLE option, the CONS= parameter, and the DOM= parameter have been deleted. An MCS DOM is implicitly issued for all action messages.

6.8.6 4-Byte Console IDs

A console ID identifies a command issuer or a message receiver. Prior to JES3 5.2.1, JES3 used 2-byte console IDs for its command and message processing. The 2-byte console IDs of JES3 managed operator consoles and remote consoles (BSC/RJP, SNA/RJP, NJE, and BDT) were assigned during JES3 initialization. The IDs were assigned sequentially based on the order of the CONSOLE statements in the initialization stream. All MCS 1-byte console IDs were converted to the JES3 2-byte console IDs that were used internally by JES3. On the global, JES3 could receive commands from MCS consoles with a 4-byte console ID. However, JES3 could not send command responses back to the issuing console unless the console had also a 1-byte console ID.

Because MCS uses unique 4-byte console IDs for all consoles in the entire sysplex, JES3 5.2.1 is enhanced to support 4-byte console IDs. JES3 5.2.1 can receive and respond to commands from any console in the sysplex. The 4-byte console ID is also accepted by the MESSAGE macro.

User DSPs must be updated to use 4-byte console IDs and CNDBs.

6.8.7 Symbolic Console Names

Prior to JES3 5.2.1, all JES3 managed consoles had console IDs and console names associated with them. The console name was defined in the JNAME field of the CONSOLE statement and was used in JES3 commands, messages, and MLOG/DLOG. MCS consoles did not have JES3 names and were referenced only by 1-byte console IDs. The console names in JES3 messages or MLOG/DLOG were in the form CN(1-byte_console_ID).

MCS sysplex assigns symbolic console names for all MCS consoles. MCS WTO, WTOR, and MGCRE macro services accept symbolic console names in lieu of console IDs as the destination of a message or the issuer of the command. JES3 5.2.1 supports MCS console names in command and message processing. The console name is carried in the CNDB throughout processing.

JES3 5.2.1 also continues to define console names for remote consoles through the JNAME keyword in the CONSOLE initialization statement.

6.8.8 Command and Response Token Support

MCS sysplex introduced the command and response token (CART) for extended MCS consoles to allow a command response to be associated with the command. The CART is supplied by the command issuer. The command processor includes the CART in the command response message(s), thus allowing the issuing program to uniquely associate a command instance with its response.

Note: Prior to JES3 5.2.1, JES3 did not have CART support.

In JES3 5.2.1, JES3 also uses the CART to insure accurate presentation of command responses to the NJE and RJP extended MCS console. The CART is carried in the CNDB throughout the command processing.

When a command with a CART is issued from an EMCS console, JES3 obtains the CART from the SVC 34 subsystem interface and saves it in a CNDB. Later, when JES3 command processors issue command responses through the MESSAGE macro, the macro points to the CNDB that contains the CART that associates the command response with the command. The MESSAGE macro service routine then issues a WTO macro with CART= parameter to present the message for MCS processing.

6.8.8.1 Command Response Support

Prior to JES3 5.2.1, JES3 did not provide any indication that a specific message was issued in response to a command.

JES3 5.2.1 *INQUIRY and *MODIFY command processing flags the CNDB to indicate that all messages in response to the command are to be marked in the MESSAGE to WTO conversion as “command responses.”

When the JES3 MESSAGE to WTO Converter detects from the CNDB that a message is in response to a command, the DESC=5 parameter is included on the invocation of the WTO macro. If descriptor code 5 is not specified, command responses are considered unsolicited messages. Descriptor code 5 causes command responses to be considered as solicited messages. Extended MCS consoles (such as a TSO operator session), which may filter the message reception by selecting solicited messages only, can also receive JES3 command

responses. This also enables automated operations to identify messages that are command responses.

Chapter 7. New Function Considerations for JES3 5.2.1

The following functional changes made to MVS and JES3 should be evaluated when migrating to JES3 5.2.1:

- JES3 sysplex operations
  - See Chapter 5, “MCS Sysplex Operations” on page 67 and Chapter 6, “JES3 5.2.1 Sysplex Operations” on page 79.
- System symbols
- Automatic restart management (ARM)
- MVS shared tape allocation

7.1 System Symbols and JES3

The parallel sysplex environment is generally replicated with a similar setup on each of the MVS systems. However, despite this similarity, there are a number of unique definitions for each system. For example, each MVS system in the sysplex must have a unique system name. Job names for multi-system applications should be unique to simplify problem determination, particularly in the ARM environment.

To allow installations to use a single parmlib member, a single procedure or a single command to handle multiple MVS systems in a sysplex, MVS/ESA 5.2 provides system symbols substitution, which allows replication of definitions.

System symbols act like variables in a program. They can take on different values based on the input to the program. When you specify a system symbol in a parmlib member, the system symbol acts as a “place holder.” Each system that shares the definition replaces the system symbol with a unique value during initialization. Symbolic substitution can be used in:

- SYS1.PARMLIB member parameters
- MVS commands
- JES3 commands
- Started task and TSO logon JCL

The intent of this support is to enable the use of a single parmlib member, a single procedure, or a single command to handle multiple instances across the sysplex. System symbols have their names defined to the system. Your installation defines substitution texts or accepts system default texts for system symbols. The system symbols are the following:

- &SYSCLONE
- &SYSNAME
- &SYSPLEX

MVS/ESA 5.1 provided base support for system symbol variables by defining two system symbol variables (&SYSNAME and &SYSPLEX). MVS/ESA 5.2 defines additional symbolic variables:

&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 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.

user symbols Up to 100 system symbol variables that the user can define with a name of his own choosing.

The symbolic variables are defined at IPL time. The term “static system symbols” is also used to refer to the system symbolic variables because they are defined at IPL time and remain fixed for the life of the IPL.

Static system symbols in source JCL are supported for demand select jobs (started tasks and TSO logons). JCL rules are used for determining where the symbols can be used and how they are resolved. Both system symbols and JCL symbolic parameters are used to resolve the symbols in the source JCL for demand select jobs. If the JCL symbolic parameters redefine a system symbolic, the value associated with the JCL symbolic parameter is used. For batch jobs, only the JCL symbolic parameters are used.

Figure 41 shows the principles of how the global JES3 receives the system symbol tables from JES3 main processors and how they are passed to converter/interpreter processing for demand select jobs.

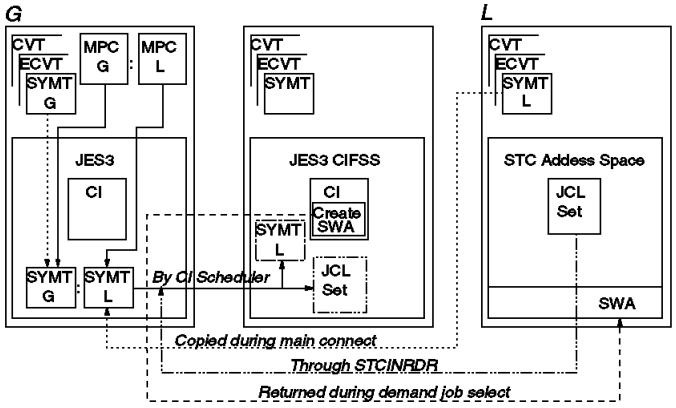


Figure 41. JES3 Support for System Symbols

JES3 Converter/Interpreter (CI) service controls the conversion of JCL statements to its internal representation. The three principal phases of CI service are:

- Converter/Interpreter phase - uses MVS services to convert and interpret (through the MVS interpreter) the JCL into scheduler control blocks (SWA) on the global main or in a CI functional subsystem address space.
- Prescan phase - creates JES3 job tables from the scheduler control blocks for use in the postscan phase. At the end of this phase, JES3 moves the scheduler control blocks from the SWA to the JES3 spool.
- Postscan phase - locates data sets and gathers information for use in JES3 device setup.

In the JES3 global address space, the CI dynamic support program (DSP) controls all three of the above phases. More than one copy of the CI DSP can be active at a time.

CI service can also take place in a CI functional subsystem (FSS) address space. A CI FSS address space contains many functions similar to a JES3 address space and can operate on the global or any local main.

In order for the replication of installation definitions (cloning) to work for demand select jobs, JES3 must use the correct system symbols table during the MVS C/I phase, even if the conversion occurs on a system other than where the demand select job is to execute. When the local connects, it passes the connecting system's system symbols to the global. When a demand select job is scheduled for CI, the symbol table of the system where the job executes is passed with the JCL to the processor that performs the JCL conversion. For ARM restarts, JES3 uses the symbol table that was used for the original execution of the job.

7.1.1 Defining System Symbols

The system symbols *&SYSNAME* and *&SYSCLONE* are part of the statements defined in SYS1.PARMLIB member IEASYMxx.

Figure 42 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(A3)
                SYSPARM(49)
                SYSNAME(SC49)
                SYMDEF(&SSNLST01='XX')
SYSDEF          HWNAME(P201)
                LPARNAME(A3)
                SYSPARM(50)
                SYSNAME(SC50)
                SYMDEF(&SSNLST01='XX')
SYSDEF          HWNAME(P101)
                LPARNAME(A1)
                SYSPARM(XX)
                SYSNAME(SC52)
                SYMDEF(&APPCLST1='XX')
                SYMDEF(&CMDLIST1='XX,00')
                SYMDEF(&SSNLST01='XX')
```

Figure 42. 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 43 on page 116 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 43. D SYMBOL Command for SYSNAME SC52

7.2 Automatic Restart Management

The purpose of automatic restart management (ARM) is to provide fast efficient restarts for critical applications when they fail. ARM is an MVS recovery function that improves the time required to restart an application by automatically restarting the batch job or started task (STC) when it unexpectedly terminates. These unexpected outages may be the result of an ABEND, system failure, or the removal of a system from the sysplex.

A primary availability requirement for all system environments is to reduce the duration and impact of an outage. The sysplex environment helps to achieve this objective through the following recovery strategies:

- Detection and isolation of failures - detect a failure and route work around it.
- Recovery of resources - release critical resources as soon as possible so that other programs and systems can acquire them. This reduces contention and speeds up the recovery or restart of work affected by the failure.
- Recover lost capacity - restart or resume processing that was affected by the failure. For example, in a database environment, the DB manager should be restarted quickly so that its log recovery can be processed.

ARM uses *event driven* failure recognition, such as *end-of-memory (EOM)* and *sysgone* processing, to trigger restart activities. It forms part of the integrated sysplex recovery, which includes:

- Sysplex failure management (SFM) - provides automated sysplex recovery for signalling failures, missing status updates and reconfiguring systems in a PR/SM environment.
- Workload manager (WLM) - provides statistics on the remaining processing capacity in the sysplex.

ARM provides an ongoing continuity with automation (AOC/MVS) and production control (OPC/ESA) products by augmenting their function. The design is simple and is intended to reduce the need to manually adjust systems, balance

workloads or provide special recovery instructions relating to abnormal situations. It makes contingency planning far more flexible.

For detailed information about ARM, see *MVS/ESA SP Version 5 Sysplex Migration Guide*.

7.2.1 ARM Environment

Automatic restart management is a function introduced in MVS/ESA SP 5.2.0. It runs in the cross-system coupling facility (XCF) address space and maintains its own dataspaces. ARM requires a primary couple data set to contain policy information as well as status information for ARM managed jobs. It supports both JES2 and JES3 environments. A started task or batch job that is a user of ARM services and unexpectedly terminates is restarted on the same system. Started tasks or batch jobs that are on a system that fails are restarted on another MVS image in the sysplex.

In a sysplex with a mixture of systems, some at lower release levels, ARM is operational *only* on those systems that have MVS/ESA SP 5.2.0 and JES3 5.2.1. The exception is a started task, started using the SUB=MSTR specification, on a MVS/ESA SP 5.2.0 running a lower level of JES3.

A pre-JES3 5.2.1 system cannot be a candidate for ARM restarts. Batch jobs and started tasks that run under JES3 can register successfully only on a system that is running MVS 5.2 and JES3 5.2.1. Figure 44 illustrates the various components in this environment.

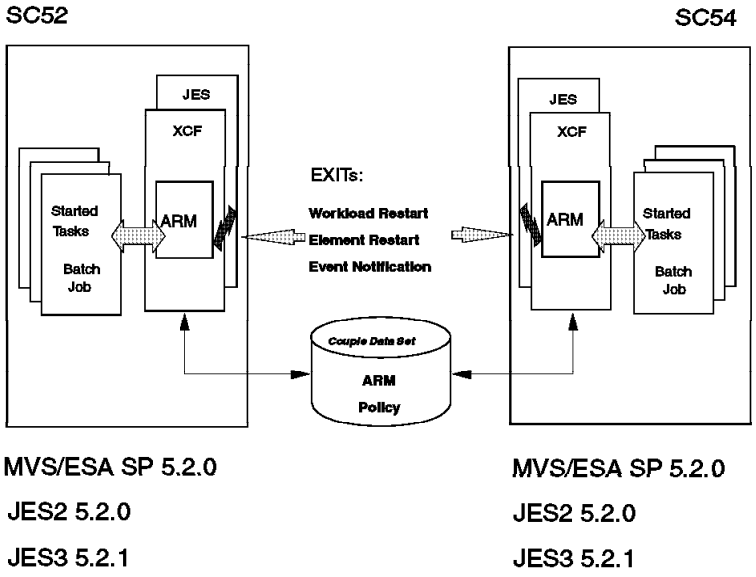


Figure 44. Automatic Restart Management Environment

7.2.2 Element Registration

For a batch job or started task to be ARM restartable, they must register with ARM using an authorized macro service IXCARM. The registration requires the specification of an “element” name. This element name must be unique in the sysplex as that is how ARM identifies jobs under its control for restart. The element name can be from one to sixteen characters. The three basic ARM services that a job must use as depicted in Figure 45 are:

REGISTER Early in the initialization processing for a job, a program that wants to use ARM for restart must issue the IXCARM REQUEST=REGISTER macro to register. Also specified on the macro request is the element name.

READY When a program that has issued the register request has completed initialization and is ready to process, it must issue the IXCARM REQUEST=READY request.

DEREGISTER Before a job completes its normal termination processing, the program must issue an IXCARM REQUEST=DEREGISTER request to remove all ARM restart possibilities.

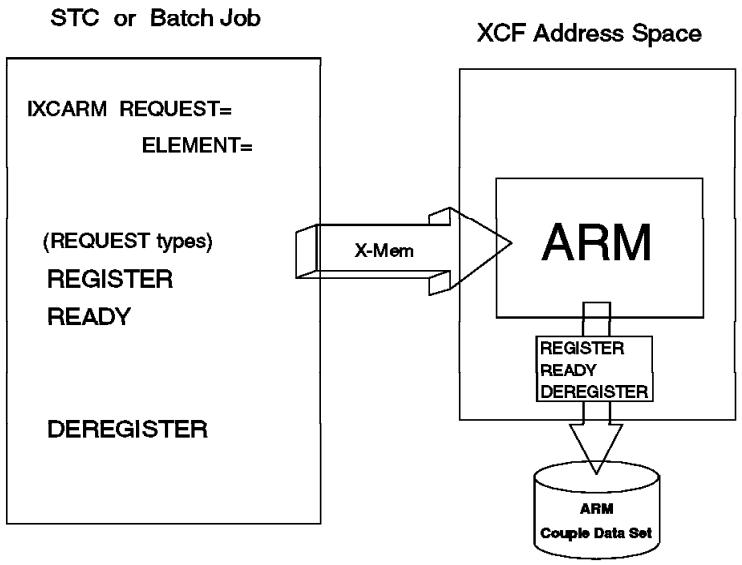


Figure 45. Registering with ARM

7.2.3 ARM Element States

ARM puts each element into one of five states to identify the last interaction that the element had with it. These states are:

Starting From the initial registration (IXCARM-Register) of a program as an ARM element to it subsequently becoming ready (IXCARM-Ready).

Available From the time the element becomes ready (IXCARM-Ready) until the element either deregisters from ARM or terminates.

Failed From when ARM detects the termination of an element, or termination of the system on which the element had been running, until ARM initiates a restart of the element.

Restarting From ARM's initiation of a restart until the subsequent reregistration of the element.

Recovering From the element's registration after an ARM restart to its subsequent issuing of IXCARM-Ready.

The state of a given element will be part of the information provided when ARMSTATUS is requested through the DISPLAY XCF command for one or more elements. Figure 46 shows an element's use of ARM services and how they affect the state of that element.

Initially, an element is unknown to ARM. The first interaction with ARM occurs when the program makes itself known to ARM by invoking the register service and specifying the element name under which it is to be registered. ARM sets the state of the element to *starting*.

When the element is ready to accept new work, it invokes the ready service, and ARM sets the state of the element to *available*. Once the element's state is available, there are no more interactions with ARM until the element terminates. Before terminating normally, the element invokes the deregister service, which again makes the element unknown to ARM.

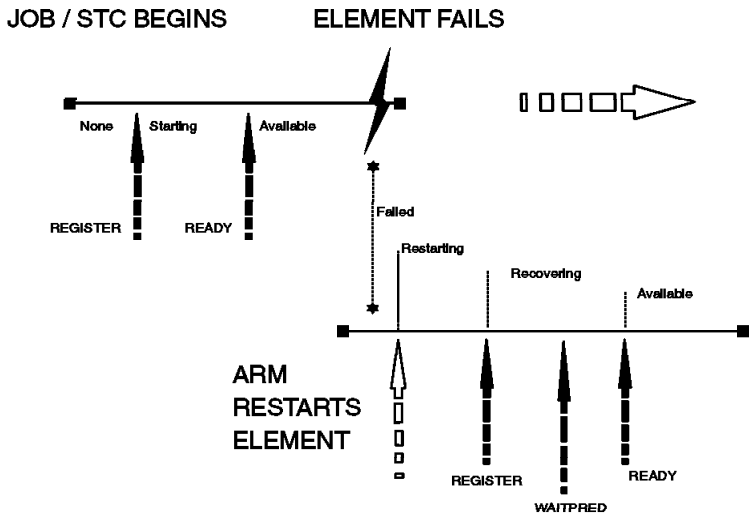


Figure 46. States of an ARM Element

When ARM detects that the element has unexpectedly terminated, that is, the program has terminated without invoking the deregister service, or that the system on which the element had been running has left the sysplex, ARM sets the element's state to *failed* as part of its processing to restart the element. An element will be in the failed state for a very brief interval.

When ARM restarts the element, it sets the state to *restarting*. When a restarting element subsequently invokes the register service, ARM sets the state to *recovering*. Once the element is prepared to accept new work, it notifies ARM by invoking the ready service, which causes ARM to set the element's state to *available*.

7.2.4 Element Registering with ARM and JES3

When a started task or batch job requests to be registered as an ARM element, ARM processing includes the registering of that element with JES3, as shown in Figure 47. JES3 returns a job token related to the registering job and the name of the JES XCF group in which that job is running.

ARM uses this job token and XCF group name as an identifier in all subsequent interactions with JES3 for this element. JES3 also uses the job token and XCF group name as an identifier when it subsequently asks or tells ARM something about an element.

STCs started under the master subsystem are supported because they can request a job ID from JES3. In this case, the only purpose of registration is to associate the STC with a JES XCF group if possible. If the STC has requested a job ID from JES3, the register function returns an actual group name. Otherwise, it returns a blank group name. In either case, the register function returns a dummy job token.

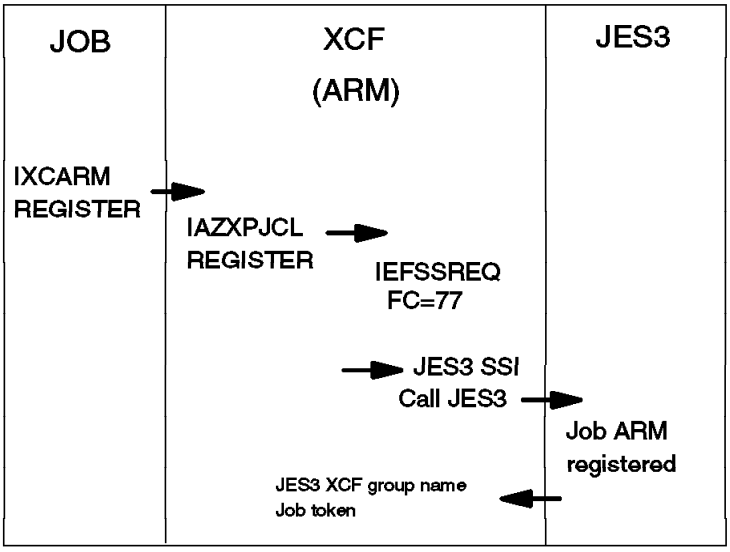


Figure 47. A Job Registering with ARM and JES3

The element registers with ARM by using the IXCARM macro request type of REGISTER. The request is queued to the XCF address space where ARM executes. ARM then passes the register request to a processing module that obtains the JES name and job ID of the job and issues an IEFSSREQ request to a JES3 SSI routine. The request is queued to the JES3 main task where JES3 locates the JQE of the job and flags the job as ARM registered. JES3 passes back to ARM the JES3 XCF group name for the sysplex and a job token.

7.2.5 ARM Element Termination Restarts

Besides internal changes, JES3 provides a new command extension to the cancel command to support ARM.

```
*F J=100,ARMR
*C mainname,100,ARMR
```

Cancelling with ARMR, an abbreviation for ARMRESTART, enables ARM to restart the element when it's registered in ARM.

For batch jobs or started tasks to be restarted because of system failures, they have to be registered with ARM. To register with ARM, the executing program has to do the registration.

ARM has two restart types: element termination (ELEMTERM) and system termination (SYSTERM). A *C SC50,100,ARMR command forces ARM to an ELEMTERM.

When ARM restarts an element that terminated with an ELEMTERM, this element is restarted on the same system where it was previously active. Even if the initiators of the system are all busy or not started, JES3 tries to restart the job on that system. This is done by setting the system affinity of the restarting job to the same system.

The system affinity and the status of the job can be displayed using:

```
*I Q,H,ARM
```

Other commands that display ARM status are:

***I J=** and ***X DISPLAY**

The job failure options are ignored when a job is ARM restarted. The failure options that can be specified in the initialization stream or by the //*MAIN statement are:

```
FAILURE={CANCEL|HOLD|PRINT|RESTART|}
```

Dump job ignores ARM jobs when dumping jobs from the job queue. Dependent job control (DJC) jobs cannot register with ARM. If a job is an ARM registered element, the output is as following:

```
*I Q,H,ARM
 IAT8674 JOB NET      (JOB00079) P=15 CL=A        HOLD=(ARM) MAIN
 IAT8674 JOB NET      (JOB00301) P=15 CL=A        HOLD=(ARM) MAIN
 IAT8698 HELD JOB SUMMARY:
 IAT8696     2 JOBS IN ARM HOLD
```

```
*I J=79
 IAT8674 JOB NET   (JOB00079) P=15 CL=A   HOLD=(ARM) R=(3072K,3072K),
                    MAIN(EXECUTING-SC50)
```

```
*X DISPLAY,J=79
 IAT6306 JOB (JOB00079) IS DISPLAY , CALLED BY ROGERS
 IAT7762 - 15 JOB00079 NET      SE=(CI-COMPLETE,MAIN-NOSTAT,
 IAT7762 - OUTSERV-COMPLETE,PURGE-NOSTAT),JCTMAINS=(SC50),
 IAT7762 - ORG=ANYLOCAL,CLASS=A,
 IAT7762 - GROUP=A,JTRACKS=00000004,IORATE=MED,
 IAT7762 - PROCLIB=IATPLBST,LINEST=020,PAGEST=00000500,
 IAT7762 - BYTEST=00999999,CRDEST=200,
 IAT7762 - OUSID=STC,HOLD=ARM,DJCNET=WTSCPLX3
 IAT7450 JOB DISPLAY  (JOB00623) PURGED
```

A job that is being ARM restarted when it fails with an ELEMTERM termination can never execute on a different system. This is because the job would try to register with ARM using the same element name on a different system than the restart system. This is not allowed, and the following return code and reason code are issued by ARM: (RC0008, RSN0150 of IXCARM REQUEST=REGISTER).

The job would run fine after the ARM RESTART_TIMEOUT is expired and ARM deregisters the element. However, this is not an ARM controlled restart anymore and is indicated by the following message.

```
IXC803I JOBNAME ARMTEST2, ELEMENT ARMTEST2STEP1 WAS DEREGISTERED. 869
THE RESTART TIMEOUT THRESHOLD HAS BEEN REACHED.
```

Operators may look at the ARM status of the job by issuing the following command:

```
D XCF,ARMSTATUS,DETAIL
```

The command response follows:

```
D XCF,ARMSTATUS,DETAIL
IXC392I  19.43.12  DISPLAY XCF 876
ARM RESTARTS ARE ENABLED
-------------- ELEMENT STATE SUMMARY --------------   -TOTAL-  -MAX-
STARTING  AVAILABLE  FAILED  RESTARTING  RECOVERING
       0          1       0           1           0        2    200
RESTART GROUP:ARMDRVR#DEFAULT  PACING  :    5    FREECSA:   10      20
ELEMENT NAME :$@$ARMTEST1STEP1 JOBNAME :N/A      STATE   :RESTARTING
  CURR SYS :SC55               JOBTYPE :JOB      ASID    :N/A
  INIT SYS :SC55               JESGROUP:WTSCPLX3 TERMTYPE:ALLTERM
  EVENTEXIT:IEFBR14            ELEMTYPE:XARMDRVR LEVEL   :  300
  TOTAL RESTARTS :        1    INITIAL START:04/27/95 19:15:21
  RESTART THRESH :   1 OF 3    FIRST RESTART:04/27/95 19:37:25
  RESTART TIMEOUT:      600    LAST  RESTART:04/27/95 19:37:25
RESTART GROUP:DEFAULT          PACING  :    0    FREECSA:    0       0
ELEMENT NAME :NET@SC47M        JOBNAME :NET      STATE   :AVAILABLE
  CURR SYS :SC47               JOBTYPE :STC      ASID    :00FC
  INIT SYS :SC47               JESGROUP:WTSCPLX3 TERMTYPE:ELEMTERM
  EVENTEXIT:ISTINCAR           ELEMTYPE:SYSVTAM  LEVEL   :    1
  TOTAL RESTARTS :        0    INITIAL START:04/25/95 09:53:22
  RESTART THRESH :   0 OF 3    FIRST RESTART:*NONE*
  RESTART TIMEOUT:      300    LAST  RESTART:*NONE*
```

7.2.6 ARM System Termination Restarts

On system termination, ARM also sets a system affinity for the failing jobs. However, the jobs are restarted on different systems in the sysplex according to the TARGET_SYSTEM definition in the ARM policy. Here again, changing the system affinity is not possible because ARM has already determined the system where the job should re-register.

The ARM restart is independent of the *R SC50,NET command issued by the operator to restart the job NET.

JES3 must process the SYSGONE for the failed member before ARM restarts jobs to that member.

When ARM restarts jobs to another JES3 main, this JES3 main communicates with ARM as follows:

- To query if a job is still registered
- To notify ARM when a queued job is canceled
- To notify ARM when a registered job ends

7.3 ARM Capability without Program Changes

In many installations, for many programs that are used, no source is available or should or could not be changed.

For some programs, or even some utility programs like IEBGENER, it would be nice to have them ARM capable.

For this purpose, a program has been written as a sample to avoid making code changes to existing programs. This program exists in two versions. The listings can be found in Appendix D, “Sample ARM Driver Program” on page 175.

To use the ARMDRVR program, you don′t need to change your programs to register and deregister with ARM. However, you have to change one line of your JCL, as shown in Figure 48. With ARMDRVR, you call the ARMDRVR program instead of your application program by providing the original program call as a PARM for the ARMDRVR program. The ARMDRVR program registers with ARM and calls the application program internally after this process. After the application finishes, the control passes back to ARMDRVR to do the deregistration from ARM.

A second way to register jobs with ARM, when using ARMDRVR for many jobs, is to code JES3 exit IATUX33 and make the JCL modification on the EXEC statement.

Figure 48 shows how to change the JCL.

```
Change from :
   //STEP1 EXEC PGM=IEFBR14,PARM='OTHER PARMS'
to
   //STEP1 EXEC PGM=ARMDRVR,PARM='IEFBR14/OTHER PARMS'

or from :
   //STEP2 EXEC PGM=IEBGENER
to
   //STEP2 EXEC PGM=ARMDRVR,PARM='IEBGENER'
```

Figure 48. JCL Modification for ARMDRVR

Figure 49 shows a job executing using the ARMDRVR program. The following lines are messages issued by the ARMDRVR program:

```
RC=0000 RSN=0000 - REGISTER REQUEST: ARMTEST1STEP1
```

```
IAT2000 JOB ARMTEST1  (JOB000123) SELECTED SC50     GRP=A
IEF403I ARMTEST1 - STARTED - TIME=15.29.28
RC=0000 RSN=0000 - REGISTER REQUEST: ARMTEST1STEP1
RC=0000 RSN=0000 - READY REQUEST:    ARMTEST1STEP1
   .
   .  ... Here the job does what it should do ...
   .
RC=0000 RSN=0000 - DEREGISTER RQUEST:ARMTEST1STEP1
IEF404I ARMTEST1 - ENDED - TIME=15.30.30
```

Figure 49. ARMDRVR Messages

Figure 50 shows an ARMDRVR job that calls a non-APF authorized program. The job runs because the authorization to register with ARM is done by the ARMDRVR program.

```
IAT2000 JOB ARMRSTR3  (JOB000125) SELECTED SC50     GRP=A
IEF403I ARMRSTR3 - STARTED - TIME=15.29.28
CSV019I REQUESTED MODULE WT1     NOT ACCESSED, IS IN NON-APF LIBRARY/CO
NCATENATION
CSV028I ABEND306-0C  JOBNAME=ARMRSTR3  STEPNAME=STEP3
RC=0000 RSN=0000 - REGISTER REQUEST: ARMRSTR3STEP3
RC=0000 RSN=0000 - READY REQUEST:    ARMRSTR3STEP3
   .
   .  ... Here the job does what it should do ...
   .
RC=0000 RSN=0000 - DEREGISTER RQUEST:ARMRSTR3STEP3
IEF404I ARMRSTR3 - ENDED - TIME=15.30.30
```

Figure 50. Non-APF Authorized Program Using ARMDRVR

7.3.1 Different ARM Driver Programs

There are two different samples of the ARMDRVR program in Appendix D, “Sample ARM Driver Program” on page 175. The difference is in the way they generate the ARM element name.

The first module (ARMDRVR1) generates an element name that is created by using the jobname (8 bytes) and the stepname (8 bytes). Having the job and the stepname present in the element name allows a method to define policies that

allows for step restarts. See 7.3.2, “Step Restarts Using ARM” on page 125, for details. The problem of having the jobname as the first part of an element name is that the policy must now be defined very specifically for the jobs.

The second module (ARMDRVR2) prefixes the element name (using prefix $@$). This allows the creation of a general policy for ARM driver jobs. Because the element name can only be 16 bytes, the stepname can only be the first 5 bytes.

The element type in both modules is called XARMDRVR.

Figure 51 shows how to set up a policy for jobs using the ARMDRVR2 module.

```
//ARMPOL1 JOB (POK,999),ROLAND,MSGCLASS=T,NOTIFY=&SYSUID
//STP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
//SYSIN    DD  DATA,DLM='||'
/* ######### DEFINE AUTOMATIC RESTART MANAGER POLICY ######### */
   DATA TYPE(ARM)

   DEFINE POLICY NAME(ITSOARM1) REPLACE(YES)

     RESTART_ORDER
       LEVEL(100)
         ELEMENT_NAME($@$*)
         ELEMENT_TYPE(XARMDRVR)
       LEVEL(200)
         ELEMENT_NAME($@$ARMTEST1STEP1)

     RESTART_GROUP(ARMDRVJOBS)
       TARGET_SYSTEM(*)
       FREE_CSA(10,20)
       RESTART_PACING(5)
       ELEMENT($@$*)
         RESTART_ATTEMPTS(3,600)
         RESTART_TIMEOUT(600)
         READY_TIMEOUT(600)
         TERMTYPE(ALLTERM)
         RESTART_METHOD(BOTH,PERSIST)

     RESTART_GROUP(ARMTEST1)
       TARGET_SYSTEM(SC54,SC55)
       FREE_CSA(10,20)
       RESTART_PACING(5)
       ELEMENT($@$ARMTEST2STEP1)
         RESTART_ATTEMPTS(3,600)
         RESTART_TIMEOUT(600)
         READY_TIMEOUT(600)
         TERMTYPE(ALLTERM)
         RESTART_METHOD(ELEMTERM,JOB,'TRAUNER.JCL.LIB(ATEST4A)')
         RESTART_METHOD(SYSTERM,PERSIST)

||
```

Figure 51. ARM Policy for ARMDRVR

7.3.2 Step Restarts Using ARM

For batch jobs it is possible to set up a step restart procedure using ARM. Assume that you have the a job, as shown in Figure 52, and you need to restart this job on a step basis.

7.3.2.1 JCL for Step Restart Job

This job is using the ARMDRVR program to register and deregister during each step of the job using the following element names in each step:

Step 1 PRODJOB1STEP1

Step 2 PRODJOB1STEP2

Step 3 PRODJOB1STEP3

The element name consists of the jobname and stepname, as shown in D.1, “ARM Driver Program 1” on page 175.

```
//PRODJOB1 JOB (POK,999),ROLAND,MSGCLASS=T,NOTIFY=&SYSUID
//*
//STEP1    EXEC PGM=ARMDRVR,PARM='IEBGENER'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,DSN=TRAUNER.JCL.LIB(A1)
//SYSUT2   DD DISP=SHR,DSN=TRAUNER.JCL.LIB(A2)
//SYSIN    DD *
//*
//STEP2    EXEC PGM=ARMDRVR,PARM='IEBGENER'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,DSN=TRAUNER.JCL.LIB(A2)
//SYSUT2   DD DISP=SHR,DSN=TRAUNER.JCL.LIB(A3)
//SYSIN    DD *
//*
//STEP3    EXEC PGM=ARMDRVR,PARM='IEBGENER'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,DSN=TRAUNER.JCL.LIB(A3)
//SYSUT2   DD DISP=SHR,DSN=TRAUNER.JCL.LIB(A4)
//SYSIN    DD *
```

Figure 52. Job for Step Restart

7.3.2.2 ARM Policy Statements for Step Restart Job

The ARM policy statements for the step restart job are the following:

```
RESTART_GROUP(PRODJOB)
  TARGET_SYSTEM(*)
  ELEMENT(PRODJOB1STEP1)
    TERMTYPE(ALLTERM)
    RESTART_METHOD(BOTH,PERSIST)
  ELEMENT(PRODJOB1STEP2)
    TERMTYPE(ALLTERM)
    RESTART_METHOD(BOTH,JOB,'TRAUNER.JCL.LIB(R1)')
  ELEMENT(PRODJOB1STEP3)
    TERMTYPE(ALLTERM)
    RESTART_METHOD(BOTH,JOB,'TRAUNER.JCL.LIB(R2)')
```

Figure 53. ARM Policy for the Step Restart Job

The job is processed for restart by ARM in the following way:

Job abends in step 1 If the job fails during the first step, the job should be restarted from the beginning.

```
ELEMENT(PRODJOB1STEP1)
  TERMTYPE(ALLTERM)
  RESTART_METHOD(BOTH,PERSIST)
```

The TERMTYPE specifies that on any terminations, SYSTERM or ELEMTERM, restart the job using the appropriate restart method. The RESTART_METHOD

specifies for either SYSTERM or ELEMTERM, use the original JCL to restart the job.

Job abends in step 2 If the job fails during the second step, the job should be restarted beginning with step 2.

```
ELEMENT(PRODJOB1STEP2)
  TERMTYPE(ALLTERM)
  RESTART_METHOD(BOTH,JOB,¢TRAUNER.JCL.LIB(R1)¢)
```

The TERMTYPE specifies that on any terminations, SYSTERM or ELEMTERM, restart the job using the appropriate restart method. The RESTART_METHOD specifies for either SYSTERM or ELEMTERM, use the JCL specified in the library “TRAUNER.JCL.LIB(R1).”

Job abends in step 3 If the job fails during the third step, the job should be restarted beginning with step 3.

```
ELEMENT(PRODJOB1STEP3)
  TERMTYPE(ALLTERM)
  RESTART_METHOD(BOTH,JOB,¢TRAUNER.JCL.LIB(R2)¢)
```

The TERMTYPE specifies that on any terminations, SYSTERM or ELEMTERM, restart the job using the appropriate restart method. The RESTART_METHOD specifies for either SYSTERM or ELEMTERM, use the JCL specified in the library “TRAUNER.JCL.LIB(R2).”

Restart Jobs JCL: The restart jobs are the following:

```
   TRAUNER.JCL.LIB(R1)

//PRODJOB1 JOB (POK,999),ROLAND,MSGCLASS=T,NOTIFY=&SYSUID
//*
//STEP2    EXEC PGM=ARMDRVR,PARM='IEBGENER'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,DSN=TRAUNER.JCL.LIB(A2)
//SYSUT2   DD DISP=SHR,DSN=TRAUNER.JCL.LIB(A3)
//SYSIN    DD *
//*
//STEP3    EXEC PGM=ARMDRVR,PARM='IEBGENER'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,DSN=TRAUNER.JCL.LIB(A3)
//SYSUT2   DD DISP=SHR,DSN=TRAUNER.JCL.LIB(A4)
//SYSIN    DD *

   TRAUNER.JCL.LIB(R2)

//PRODJOB1 JOB (POK,999),ROLAND,MSGCLASS=T,NOTIFY=&SYSUID
//*
//STEP3    EXEC PGM=ARMDRVR,PARM='IEBGENER'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,DSN=TRAUNER.JCL.LIB(A3)
//SYSUT2   DD DISP=SHR,DSN=TRAUNER.JCL.LIB(A4)
//SYSIN    DD *
```

Figure 54. JCL for Step Restart Jobs

7.3.3 Restrictions for ARM Processing

A started task that runs under the master subsystem can register and restart on systems running a downlevel JES3, with the following restrictions:

- The started task must be in a restart group that contains no batch jobs or started tasks running under JES3. By default, all jobs and STCs are placed in one restart group. Therefore, an installation policy is required to isolate any started task that runs under the master subsystem into its own restart group.
- A started task must not request a job ID from JES3 before it registers with ARM. If it does, it is treated as if it started under JES3.

7.4 MVS Shared Tape Allocation

MVS/ESA 5.2 allocation has changed the way it selects non-JES3 managed tape devices. It considers the type of request, unit information on the request, and characteristics of each available tape device, including the cartridge loader state and the automatically switchable attribute. Based on these factors, the optimal device is allocated. Also a new broadcast subsystem interface (SSI) function code 78 is introduced to allow subsystems to receive control at least once for each JCL job step or dynamic allocation to influence the way the system selects the tape devices to be allocated.

Another MVS/ESA 5.2 allocation enhancement is automatic tape switching, the ability for an installation to have a category of non-JES3 managed tape devices known as automatically switchable. These devices are not dedicated to any one system; rather they can be used by all systems that are connected through a coupling facility in a parallel sysplex. The coupling facility is used to maintain the sysplex wide allocation status of the automatically switchable tape devices. The automatically switchable tape drives must also have the ASSIGN/UNASSIGN feature. Allocation uses single system assign to control which system at any given time can access the tape device.

The MVS VARY ONLINE command, when it is issued for an automatically switchable device, brings the device online but does not assign it. The automatically switchable device is assigned at step allocation or dynamic allocation time and is unassigned at step termination or dynamic unallocation time. When the MVS VARY OFFLINE command is issued for an automatically switchable device, the device is taken offline from MVS.

The MVS “VARY dddd,AUTOSWITCH,ON or OFF” command allows you to dynamically change the AUTOSWITCH attribute of a tape device without using HCD. This command is accepted only for non-JES3 managed tape devices that are offline.

Note: When a JES3 managed tape device is brought online to a JES3 main, it is also automatically varied online to MVS on that system. If the device is assignable, JES3 uses multi-system assign that allows the device to be accessible from all the systems in the JES3 complex. When a JES3 managed tape device is varied offline from a JES3 main, it is also automatically varied offline from MVS on that system. If the device is assignable, JES3 unassigns the device on that system.

7.4.1 JES3 Coexistence with MVS Shared Tape Allocation

JES3 main device scheduler (MDS) on the global processor reserves resources (data sets, devices, and volumes) for jobs based on the global knowledge of resource status (maintained in the JES3 global address space). The resources are reserved for jobs before they are passed to MVS for execution. When jobs go through the MDS phase, it is not necessarily known on which processor they will be executing. This is one of the reasons why JES3 uses multi-system assign for assignable tape drivers. During job step initialization, MVS allocation calls JES3 to get the device numbers that MDS has reserved for the job. At MVS unallocation time, JES3 is informed about the released resources.

Some considerations when both JES3 managed and automatically switchable tape drives are used:

- JES3 managed devices and automatically switchable devices must form separate non-overlapping device groups.
- JES3 managed devices must be defined in the JES3 initialization stream such that subgeneric splits are avoided.
- The JES3 SETNAME initialization statements must define the UNIT names that will be managed by JES3.
- The automatically switchable devices must have UNIT names other than the ones defined on the JES3 SETNAME initialization statements.
- Input and output tape data sets must be directed to either JES3 managed devices or to the automatically switchable devices by using the proper UNIT definition on the JCL.

  Note: When the system obtains volume serial information from catalogs for a cataloged tape input data set, the retrieved device information must be overridden on JCL if JES3 manages the generic tape device type in order to direct the allocation to the automatically switchable devices or vice versa.

Chapter 8. JES3 5.2.1 Migration

The pre-JES3 5.2.1 initialization stream is accepted by JES3 5.2.1. Warning messages are issued for initialization statements and keywords that are not supported on JES3 5.2.1.

Fallback PTFs are made available for pre-JES3 5.2.1 systems.

Start JES3 5.2.1 positioning activities *immediately*. MVS/ESA Version 4 or MVS/ESA Version 5 systems provide a number of MCS or NetView facilities that help to replace the JES3-managed consoles and other console processing facilities with corresponding MCS or NetView facilities.

JES3 5.2.1 requires a JES3 warmstart and a complex-wide IPL with CLPA or refresh of MLPA to install.

All JES3 systems in the sysplex must be at the JES3 5.2.1 level. Coexistence of mixed levels of JES3 in the same JES3 complex is not supported.

Migration from JES3 Releases 3.1.3, 4.2.1, or 5.1.1 requires a warm start.

Fallback to JES3 Releases 3.1.3, 4.2.1, or 5.1.1 requires a warm start.

Migration from a JES3 release prior to 3.1.3 requires a cold start. Dump job (DJ) **cannot** be used to carry jobs over the cold start unless you first migrate to an interim release.

Fallback to a JES3 release prior to 3.1.3 requires a cold start. Dump job (DJ) **cannot** be used to carry jobs over the cold start unless you first fallback to an interim release.

Migration to JES3 5.2.1 release may be accomplished in the following steps:

- Positioning
- Planning
- Operational considerations
- Programming considerations
- Implementation activities

8.1 Positioning for JES3 5.2.1

For installations running MVS/ESA Version 4 or MVS/ESA Version 5 systems, it is a good idea to replace JES3-managed consoles with other facilities such as MCS consoles, NetView consoles, and increased automation on your current MVS level. Use the MVS MCS facilities instead of the JES3 facilities where possible. At the conclusion of this step, the desired environment for the installation should be as follows:

- Use of JES3-managed operator consoles is eliminated.
- Message-based automation and exit processing is performed using NetView (or a similar product), and MPF exits are used on the issuing system.

Installations migrating to this release from MVS/SP Version 2 or Version 3 may still achieve many of the outlined objectives; however, not all MVS MCS functions are available. For example, the enhancements to MCS consoles to

provide JES3-like features are not available as they are on MVS/ESA Version 4 and later systems. You may choose not to expend any effort on this aspect of the positioning activity. Other installations may choose to displace JES3 consoles with NetView consoles and proceed with the positioning.

8.2 Planning for JES3 5.2.1

The most important task is to understand the existing MCS sysplex support as provided in MVS/ESA Version 4. This support, suppressed by JES3 on pre-JES3 5.2.1 systems, is fully functional when JES3 5.2.1 is installed. *MVS/ESA Planning: Operations* is a good source of information on the MCS sysplex environment. Key changes introduced with MCS sysplex are:

- The ability to control multiple systems from any console in the JES3 complex, not just those consoles attached to the JES3 global processor.
- The transport of commands and messages between systems by MCS, rather than exclusively by JES3.

Specific JES3 considerations in planning for MCS sysplex are discussed in the following topics.

8.3 Checklist of Migration Actions for Initialization of JES3 5.2.1

__ Update your CONSOLxx and MPFLSTxx members of SYS1.PARMLIB due to the activation of the MCS sysplex environment with JES3 5.2.1.

__ Convert any JES3 operator or MLOG consoles to another console manager such as MCS or NetView since JES3 5.2.1 removes support for these consoles.

__ Update your MPF tables and exit routines such that you perform MPF processing only on the originating system whenever possible. When this is not possible, consider use of the JES3 global MPF function specified using the new GLOBMPF keyword of the CONSTD initialization statement.

__ Remove the MSGROUTE initialization statements from your initialization stream if you no longer need this JES3 function. If you retain these statements, update the MSGROUTE J specification if appropriate for your environment due to its changed function in sysplex message routing.

__ Consider using MVS routing codes in the initialization stream and in JES3 commands in addition to, or in place of, JES3 destination classes.

__ Define a single command delimiter using the CONSOLxx CMDDELIM keyword. JES3 5.2.1 ignores any new line character specified on the CONSTD EDIT keyword.

__ Add JES3 destination class ALL to any RJP console you want to receive broadcast messages, if it is not already specified.

__ Define the new JES3DLOG address space in the STARTED general resource class (RACF V2R1) or the RACF started procedures table (ICHRIN03) in the same manner in which you define your JES, and be certain to set on the trusted bit.

__ Add the new SAVEMSG=NO specification to those RJP CONSOLE statements in your initialization stream for which it is unnecessary that

JES3 save messages while the workstation is signed off. This function can also be controlled using the *MODIFY O command.

__ Do not define MCS consoles with names beginning with SYSJ3. These names are reserved for system use.

__ Continue to IPL JES2 systems using the NOJES3 IPL parameter when JES2 and JES3 are installed on the same system.

__ Do not use an ampersand (&) as a line editing character on the CONSTD EDIT keyword, MVS uses the ampersand to delimit system symbols in commands.

__ Remove module names IATSIDO and IATSIMS from any IEALPAxx members of SYS1.PARMLIB once migration to JES3 5.2.1 is complete.

8.4 JES3 5.2.1 - Initialization

JES3 5.2.1 initialization has changed in the following areas:

- Considerations for the MCS Sysplex Environment
- Converting JES3 Operator Consoles to MCS Consoles
- MPF Processing Considerations
- Defining MSGROUTE Processing
- Using MVS Routing Codes
- Defining the Command Delimiter
- Considerations for Destination Class ALL
- Defining the JES3DLOG Address Space
- Spooling of RJP Console Messages
- Reserved MCS Console Names
- Installing JES2 and JES3 on the Same System
- Sharing JES3 Commands and Source JCL for Demand Select Jobs
- Removal of JES3 LPA-resident Modules

8.4.1 Considerations for the MCS Sysplex Environment

JES3 no longer inhibits MVS from initializing with the MCS sysplex functions active. When JES3 5.2.1 is installed, these MVS/ESA functions are available to your installation.

JES3 5.2.1 allows extended MCS consoles and the Hardware Master Console to control the entire sysplex, rather than just the single system on which they are operating as was the case with prior JES3 releases.

8.4.1.1 Migration Actions

With the activation of the MCS sysplex environment, changes to your CONSOLxx parmlib member may be required. Changes are most likely to be required if:

- The total number of CONSOLE statements (including SUBSYSTEM consoles) from every system in your sysplex is greater than 99. In the MCS sysplex environment, there is a maximum of 99 MCS consoles for the sysplex. This may require that you convert programs using SUBSYSTEM consoles (such as automated operations) to use Extended MCS consoles.
- You choose to remove the SUBSYSTEM consoles previously used by JES3 from your CONSOLxx member. JES3 5.2.1 no longer needs SUBSYSTEM consoles. Removing these definitions may help you fit within the 99 console limit for the sysplex. The DISPLAY CONSOLES,SS command on your pre-JES3 5.2.1 system will identify the subsystem consoles used by JES3.

- You define multiple consoles using the same device number on different systems, and you do not use the NAME option in the console definition. In the MCS sysplex environment, this is considered an attempt to redefine the same device and will not be accepted.
- You choose to NAME your SUBSYSTEM consoles currently used by non-JES3 products. Naming the subsystem consoles is recommended as it avoids using more console IDs than are needed for the sysplex when systems are IPLed.
- You want to scope your MCS consoles to only receive messages from certain system(s), rather than all systems in the sysplex. Use the MSCOPE keyword on the CONSOLE statement. The default is MSCOPE(*ALL), meaning the console is eligible to receive messages from all systems in the sysplex.
- You want to associate your MCS console(s) with a specific system such that commands entered from the console will be processed by that system. Use the CMDSYS keyword on the CONSOLE statement. The default is CMDSYS(*), meaning the commands will be processed on the system where the console is defined.
- You want to change the maximum value of the WTOR reply id from its default of 99. Use the RMAX keyword on the DEFAULT statement. In the MCS sysplex environment, a single range of WTOR reply ids is shared by all systems.

Refer to *MVS/ESA Planning: Operations* for additional information.

8.4.2 Converting JES3 Operator Consoles to MCS Consoles

With the activation of the MCS sysplex environment, JES3 operator consoles are no longer required to manage the sysplex. JES3 5.2.1 does not support JES3 operator consoles.

8.4.2.1 Migration Actions

If you use JES3 operator consoles, they must be converted to another console manager (such as MCS or NetView) prior to or concurrent with the migration to JES3 5.2.1.

JES3 operator consoles are defined on the CONSOLE statement in the JES3 initialization stream. MCS consoles are defined in a CONSOLxx member of SYS1.PARMLIB. Table 4 shows the mapping of the JES3 CONSOLE statement into an MCS definition.

Table 4 (Page 1 of 2). Converting JES3 Operator Consoles to MCS Consoles

JES3 CONSOLE Statement	**MVS CONSOLE Statement**
JNAME={name }	NAME (conname)
TYPE={(devicetype,model)}	UNIT {(unittype)}
DEST={(destclass,destclass(,...))}	ROUTCODE={(nnn(,nnn-nnn)(,nnn). .)}
UNIT=(procname,{ddd} (,...))	N/A
MAIN={procname}	CMDSYS {(sysname)}
ALTCON={name}	ALTERNATE {(conname)} ALTGRP(groupname)
DEPTH={nn}	N/A

Table 4 (Page 2 of 2). Converting JES3 Operator Consoles to MCS Consoles	
JES3 CONSOLE Statement	**MVS CONSOLE Statement**
LL=nnn	N/A
TIME={nn}	RNUM {(nn)} RTME {(nnn)}
LEVEL={nn}	AUTH {(MASTER/INFO/SYS/IO/CONS/ALL)}
PFK=tblname	PFKTAB(tablname)
SP=tblname	N/A
IOWAIT={seconds}	N/A
3290={YES/NO}	N/A

When converting JES3 operator consoles to MCS management coincident with the migration to JES3 SP5.2.1, using the same console name will allow for consistent routing of messages directed by console name across the migration.

The MFORM=X and DEL=W specifications in the MCS CONSOLE definition cause MCS to display messages in a manner similar to that previously provided by JES3 operator consoles. Both options may also be selected using the CONTROL S command. Determine if you wish to use these MCS presentation formats when converting JES3 operator consoles to MCS management.

Use the PFKTABxx member of SYS1.PARMLIB to define PFK tables for MCS consoles similar to those previously used for your JES3 operator consoles.

JES3 operator consoles flag each message with a special character when JES3 encounters a minimal or marginal spool space condition, or a JSAM buffer shortage. MCS consoles do not display these characters. Instead, these conditions can be identified using existing messages IAT1017, IAT1018, IAT1101, IAT1102, and IAT1103.

8.4.3 MPF Processing Considerations

Performing MPF processing on the originating system is recommended. For installations unable to distribute their MPF processing as part of the installation of JES3 SP5.2.1, a facility is provided which causes all messages otherwise routed to the global processor to be presented to MPF on the global.

8.4.3.1 Migration Actions

Perform MPF processing on the originating system whenever possible. When this is not possible, consider use of the new JES3 global MPF function specified using the GLOBMPF keyword of the CONSTD initialization statement. Be aware that use of this facility may introduce sysplex overhead if you also choose to broaden the set of messages being received by the global processor. When using the GLOBMPF function, remember that MPF processing may have to be adjusted as part of a DSI. Ensure that the MPF options on the old and new global are set up correctly. The SET MPF operator command can be used to change the MPF options for a particular system.

8.4.4 Defining MSGROUTE Processing

The MSGROUTE initialization statement is now optional. MSGROUTE is a means to alter the routing of messages for each system on a routing code basis. If a MSGROUTE statement is omitted for a system, JES3 will not perform MSGROUTE processing for messages from that system. Other elements of JES3 message routing, such as device-related routing, may still be performed.

The interpretation of the MSGROUTE J option in the initialization stream is changed to be consistent with the new sysplex message routing. Prior to JES3 SP5.2.1, specifying the J option indicated that messages with the specified routing code(s) should only be subject to JES3 global routing processing. This option caused messages to be suppressed from display on MCS consoles attached to the originating local processor.

As of JES3 SP5.2.1, the J option no longer suppresses a message from display since MCS is now responsible for the routing and display of messages. Specifying the J option now causes the routing code equivalent of the JES3 destination class to be used for the message **in place of** the message′s original routing code(s). Not specifying J for a routing code causes the routing code equivalent of the associated destination class to be **merged** with the message′s original routing code(s).

8.4.4.1 Migration Actions

Remove the MSGROUTE initialization statements from your initialization stream if you no longer need this JES3 function.

Update the MSGROUTE J specification in your initialization stream if appropriate.

8.4.5 Using MVS Routing Codes

Prior to JES3 5.2.1 JES3 message destination information was specified in terms of JES3 destination classes. JES3 5.2.1 allows MVS routing codes to be used in place of, or in addition to, JES3 destination classes in the initialization stream and JES3 commands.

8.4.5.1 Migration Actions

Determine which means of expressing message destination information is appropriate for your installation. Use of routing codes allows for consistent routing specification with MVS commands. Use of destination classes allows for somewhat more meaningful named routings.

8.4.6 Defining the Command Delimiter

Beginning with JES3 5.2.1, JES3 will ignore any new-line specification from the CONSTD EDIT keyword of the initialization stream. In its place, JES3 will use the MVS command delimiter specification defined in the MVS CONSOLxx INIT CMDDELIM= keyword. IBM has always recommended that these definitions be identical to avoid inconsistencies in command parsing.

8.4.6.1 Migration Actions

Update your command delimiter specification(s) if necessary.

MCS consoles do not support an “escape” character associated with the MVS command delimiter. Thus, any use of the MVS command delimiter from an MCS console will be interpreted as a delimiter. JES3 consoles allowed the use of an “escape” character together with JES3′s command delimiter.

8.4.7 Considerations for Destination Class ALL

JES3 no longer adds destination class ALL to consoles receiving unsolicited messages. Prior to JES3 5.2.1, JES3 added destination class ALL to all JES3 operator and RJP consoles, except those defined in the initialization stream with DEST=NONE or DEST=MLG.

8.4.7.1 Migration Actions

With JES3 5.2.1, you must explicitly specify DEST=ALL for the RJP consoles you wish to receive broadcast messages. For MCS consoles, use the LEVEL parameter to control receipt of broadcast messages.

8.4.8 Defining the JES3DLOG Address Space

JES3 5.2.1 creates a new system address space (JES3DLOG) when the DLOG hardcopy log is requested.

8.4.8.1 Migration Actions

Define JES3DLOG in the STARTED general resource class (RACF V2R1) or the RACF started procedures table (ICHRIN03) in the same manner in which you define your JES, and be certain to set on the trusted bit.

8.4.9 Spooling of RJP Console Messages

Prior to JES3 5.2.1, console messages sent to an RJP workstation while the workstation was signed off were always saved on spool. The messages were then delivered to the workstation the next time the workstation signed on.

JES3 5.2.1 makes this saving of RJP messages optional so that the system overhead can be avoided when the function is not needed.

8.4.9.1 Migration Actions

Add the new SAVEMSG=NO specification to those RJP CONSOLE statements in your initialization stream for which it is unnecessary that JES3 save messages while the workstation is signed off. This function can also be controlled using the *MODIFY O command.

8.4.10 Reserved MCS Console Names

JES3 5.2.1 reserves certain MCS console names for its use.

8.4.10.1 Migration Actions

Console names beginning with SYSJ3 are reserved for system use. Do not use console names beginning with SYSJ3.

8.4.11 Installing JES2 and JES3 on the Same System

Prior to JES3 5.2.1, JES3 prevented the MCS sysplex functions from activating during MVS IPL. When JES2 and JES3 were installed on the same system specifying CON=(xx,NOJES3) as part of the IPL parameters allowed the MCS sysplex functions to be active in spite of the presence of JES3 on the system.

8.4.11.1 Migration Actions

If you have both JES2 and JES3 SP5.2.1 installed on the same system, you should continue to IPL the JES2 system with CON=(,NOJES3) specified. This is necessary in order to enable the use of enhanced short form reply (i.e. no delimiter required) when running the JES2 system.

8.4.12 Sharing JES3 Commands and Source JCL for Demand Select Jobs

JES3 5.2.1 allows two or more systems to share JES3 commands and source JCL for demand select jobs, while retaining unique values where required.

8.4.12.1 Migration Actions

If your installation plans to use system symbols in commands, do not specify an ampersand (&) on the console standard statement (CONSTD) as a special character to be used in editing commands processed by JES3 console services.

Also, see the chapter that describes migration actions for using system symbols in *MVS/ESA SP V5 Conversion Notebook*.

8.4.13 Removing JES3 LPA-resident Modules

JES3 5.2.1 deletes some LPA-resident modules that you may have included in an MLPA list.

8.4.13.1 Migration Actions

Remove module names IATSIDO and IATSIMS from any IEALPAxx members of SYS1.PARMLIB once migration to JES3 5.2.1 is complete.

8.5 Checklist of Migration Actions for Customization of JES3 5.2.1

__ After installing JES3 5.2.1 reassemble all your JES3 exits.

__ Review your user written DSPs and exits to determine if they can benefit from the JES3 multi-line message service, IATXMLWO.

__ Review any installation-written DSPs, exits, or other modifications that process commands or issue messages. Changes may be required to take advantage of the Console Destination Block (CNDB).

__ Move any installation-written code from JES3 message exit IATUX31 to MPF exits on the originating system. JES3 5.2.1 no longer calls IATUX31. New exits 69 and 70 may be used when there is no alternative to processing messages in a JES3 global address space exit.

__ Move any code related to non-JES3 commands from the JES3 command exit IATUX18, to an MVS command exit. Update any installation-written IATUX18 code due to the JES3 5.2.1 changes in the exit interface.

__ Delete any installation-written code contained in the JES3 BDT command authorization exit IATUX56. JES3 5.2.1 no longer calls IATUX56.

8.6 JES3 5.2.1 - Customization

JES3 5.2.1 introduces customization changes in the following areas:

- Issuing Multi-line Messages from JES3 Functions
- Changes Affecting Command Processors and Message Issuers
- Changes Affecting Message Exits
- Changes Affecting Command Exits
- Changes Affecting JES3 Command Entry from a BDT Session

8.6.1 Issuing Multi-line Messages from JES3 Functions

Prior to JES3 5.2.1, the JES3 MESSAGE service was limited to single line messages. When a JES3 function needed to issue a multiple line message, the message was issued as a series of single lines. This impacted readability since the system could intersperse other message traffic with the function′s multiple line message.

Beginning with JES3 5.2.1, JES3 provides a multi-line message service usable by JES3 and user-written DSPs, and exits.

8.6.1.1 Migration Actions

Review your user written DSPs and exits to determine if they can benefit from the JES3 multi-line message service, IATXMLWO.

8.6.2 Changes Affecting Command Processors and Message Issuers

JES3 uses Console Destination Blocks (CNDBs) to save console identification and message routing information. JES3 uses the IATXCNDB macro service to work with CNDBs. IATXCNDB includes services for creation, update, copy, and extraction. You should use the IATXCNDB service whenever you work with CNDBs.

When an operator issues a command, JES3 creates a CNDB containing information about the source of the command, such as the console identifier and console type. This CNDB is passed to the DSP′s console appendage along with the command text. When your DSP needs to communicate with the operator issuing the command, you may use the MESSAGE macro service, specifying the CNDB that was saved by your appendage. In addition, any commands that your DSP may internally issue on behalf of an operator may be issued using the INTERCOM service specifying a CNDB.

CNDBs may also be used for routing of unsolicited messages. For example, JES3 creates CNDBs for much of the message destination information specified in your initialization stream. When you need to issue a message to this destination, such as a message about a particular main processor, you use the MESSAGE service specifying the appropriate CNDB.

8.6.2.1 Migration Actions

Review your installation-written DSPs, exits, and other modifications and begin using CNDBs and the IATXCNDB service when appropriate.

INTERCOM You may need to specify a CNDB on the INTERCOM macro. Use the IATXCNDB service to set up the CNDB if necessary. If a CNDB is not provided, the dummy CNDB is used.

The JES3 INTERCOM service uses MGCRE to issue all commands beginning with JES3 5.2.1. Previously, JES3 commands were queued directly to JES3 console services. MGCRE translates any non-printable data found in the command text as part of its processing. If it is not practical to eliminate the passing of this data with a command, the non-printable data can be enclosed in quotation marks. MGCRE then passes the data unchanged.

MESSAGE You may need to specify a CNDB on the MESSAGE macro. Use the IATXCNDB service to set up the CNDB if necessary.

SSISERV Replace any modifications that use the JES3 SSISERV macro to direct commands to the JES3 global because sysplex command support allows these commands to be directly issued and transported to the global by MVS.

Beginning with JES3 5.2.1, buffer pools are only used for commands awaiting processing by a DSP. Therefore, user DSPs that process messages under the DSP rather than the console appendage have to be changed. Today the DSP must run the input chain until the first outstanding action message is found.

The TYPE=OUTPUT and TYPE=MLOG formats of the console buffer mapping (IATYCNS) are deleted. TYPE=DLOG provides a mapping for the JES3 format SYSLOG records.

8.6.3 Changes Affecting Message Exits

Message traffic for the sysplex is no longer unconditionally transported to the JES3 global address space. As a result, the JES3 message processing installation exit (IATUX31) is removed. JES3 SP5.2.1 provides new installation exits 69 and 70 to allow selected messages to be processed in the JES3 global address space. These exits exploit the MVS dynamic exit facility, allowing multiple exit routines to be defined for each exit, as well as update of exit code without the need for an IPL or JES3 outage.

8.6.3.1 Migration Actions

IATUX31 processing should be moved to MPF exits on the originating system whenever possible.

Use of the new exits 69 and 70 should be limited to those messages which must be processed in the JES3 global address space, since sysplex overhead is required to transport these messages to the global.

8.6.4 Changes Affecting Command Exits

JES3 5.2.1 changes the interface to the JES3 command exit IATUX18. For details, see *MVS/ESA SP V5 JES3 Customization*. With the removal of the TYPE=MCS and automation console definitions from the JES3 initialization stream, it is no longer possible to define JES3 command authority levels for these consoles in the initialization stream for use by IATUX18. In addition, installation exit code related to non-JES3 commands entered from JES3 sources must be moved to an MVS command exit. MVS command exits run in the address space from which the command is issued. Therefore, any code you move to an MVS command exit will still run in the JES3 address space, and JES3 address space data such as device group will continue to be available.

JES3 5.2.1 provides a new return code for IATUX18 to allow the exit to indicate that it has completely processed a command. Previously, JES3 would issue an error message for any unrecognized command processed within your exit code.

8.6.4.1 Migration Actions

Move any IATUX18 code related to non-JES3 commands to an MVS command exit.

Update your IATUX18 code due to the JES3 5.2.1 changes in the exit interface.

8.6.5 Changes Affecting JES3 Command Entry from a BDT Session

Support for JES3 commands entered from a BDT session is removed in JES3 5.2.1. As a result, the command authorization exit for JES3 commands entered from a BDT session (IATUX56) is deleted.

8.6.5.1 Migration Actions

Delete any installation code contained in IATUX56.

The TSO CONSOLE operator facility provides an alternative means of interacting with JES3 from a TSO session.

8.7 Checklist of Migration Actions for Operation of JES3 5.2.1

__ Plan your migration to the MVS operations log (OPERLOG). This migration should occur either concurrent with the migration to JES3 5.2.1, or staged following that migration. Hardcopy log post-processors may need to be changed to work with the additional information provided in the MVS operations log and SYSLOG log formats.

__ Replace your use of the *SEND command to direct commands to another system in the JES3 complex with use of the ROUTE command.

__ Replace your use of the *INQUIRY R command to retrieve outstanding action messages with use of the DISPLAY REQUESTS command.

__ Activate the Action Message Retention Facility (AMRF) using either the CONSOLxx INIT AMRF(Y) parmlib specification or the CONTROL M,AMRF=Y command. Select the set of action messages to be retained using the MPFLSTxx RETAIN specification.

__ Review automation scripts and provide operator training for the new sysplex command prefix support in JES3 5.2.1. If you wish to change the command prefixes used by JES3, or their scope, use the CONSTD SYN= and PLEXSYN= parameters in your initialization stream. Determine if your programs can now execute commands on the JES3 global processor through use of MGCRE with a sysplex-scoped command prefix, rather than relying on other means (such as SSISERV) to direct the command to the global.

__ Determine if automation script improvements are possible due to JES3 5.2.1 support for the Command and Response Token (CART) and command response descriptor code.

__ Review automation scripts and operator training due to the addition of a confirmation WTOR during JES3 operator-initiated termination.

__ Consider taking advantage of the longer command lengths allowed with the *I U, *F U, and *X JMF commands.

__ Consider taking advantage of the new abilities to reroute message traffic between RJP workstation consoles (*SWITCH), and to purge the queue of backlogged messages for an RJP workstation (*FREE).

__ Update your DLOG browsers and post processors if they are affected by the changes in log content.

__ Determine an alternate means of entering JES3 commands from a BDT session if you have used this function in the past. The TSO CONSOLE operator facility is one means of interacting with JES3 from a TSO session.

__ Review the MVS/ESA Conversion Notebook if you plan to make use of system symbols in JES3 commands and source JCL for demand select jobs.

__ Examine the list of new, changed, or deleted messages for JES3 5.2.1 to determine any impacts to operator procedures or automated operations.

8.8 JES3 5.2.1 - Operations

JES3 5.2.1 introduces operations changes in the following areas:

- Selecting the Hardcopy Medium
- Replacing the JES3 *SEND Command with MVS ROUTE
- Replacing the JES3 *I R Command with MVS DISPLAY REQUESTS
- Activating the MVS Action Message Retention Facility (AMRF)
- Using JES3 Command Prefixes
- Automation Considerations for JES3 Command Responses
- Confirming JES3 Operator-Initiated Termination
- Using Longer Command Lengths
- Controlling RJP Workstation Consoles
- Changes in the JES3 DLOG Content
- Changes Affecting JES3 Command Entry from a BDT Session
- Sharing JES3 Commands and Source JCL for Demand Select Jobs

8.8.1 Selecting the Hardcopy Medium

The hardcopy medium (also known as the hardcopy log) records command and message traffic for your systems. MVS and JES3 provide four forms of the hardcopy medium:

OPERLOG centrally records command and message traffic for systems in a sysplex in Message Data Block (MDB) format

JES3 DLOG centrally records command and message traffic for systems in a JES3 complex in JES3 format. The JES3 DLOG is written to SYSLOG on the global processor.

SYSLOG individually records command and message traffic for each system in MVS format

Device individually records command and message traffic for each system in MVS format on a hardcopy printer

Replacement of JES3′s hardcopy log (DLOG) with the MVS operations log is recommended. The MVS operations log records considerably more information than does DLOG, and its processing is distributed in the sysplex rather than being centralized on the global processor. Migration to the MVS operations log

may be accomplished at installation of JES3 5.2.1, or may be staged through the use of JES3 DLOG during the migration period.

It is no longer possible to write the JES3 complex hardcopy log stream concurrently to JES3′s DLOG and a device (MLOG). Concurrent disk and device logging can be achieved through the use of the MVS Operations Log together with an MCS hardcopy log device.

8.8.1.1 Migration Actions

Plan your migration to the MVS operations log (OPERLOG). This migration should occur either concurrent with the migration to JES3 5.2.1, or staged following that migration. Hardcopy log post-processors may need to be changed to work with the additional information provided in the MVS operations log and SYSLOG log formats. *MVS/ESA Planning: Operations* contains additional information on OPERLOG migration considerations.

If you need to record the hardcopy log concurrently to disk and a hardcopy printer, use the MVS operations log in conjunction with an MCS hardcopy log device.

8.8.2 Replacing the JES3 *SEND Command with MVS ROUTE

The MVS ROUTE command replaces the JES3 *SEND command with JES3 5.2.1. The ROUTE command has significantly more function than *SEND. For example, aggregation of command responses, routing of commands to a group or list of systems, and the ability to route commands prior to JES3 initialization. The JES3 *SEND command continues to support sending commands through the NJE network.

8.8.2.1 Migration Actions

Automation scripts and operator training should be reviewed as a result of this change.

8.8.3 Replacing the JES3 *I R Command with MVS DISPLAY REQUESTS

The MVS DISPLAY REQUESTS command and MVS Action Message Retention Facility (AMRF) provide sysplex-wide action message management, duplicating JES3 function. As a result, the JES3 *I R command and JES3 message retention facilities (JMRF) are removed in JES3 5.2.1.

8.8.3.1 Migration Actions

Automation scripts and operator training should be reviewed as a result of this change.

Table 5 assists in converting JES3 *I R commands to MVS D R.

Table 5 (Page 1 of 2). Converting JES3 *I R to MVS DISPLAY REQUESTS

JES3 Command	Display Command Equivalent	Differences
*I R - Displays all outstanding action messages including WTORs. SETUP-related messages are not displayed.	D R,L	SETUP-related messages are displayed by the D R command. The D R command also displays messages that were issued before JES3 was fully initialized.

Table 5 (Page 2 of 2). Converting JES3 *I R to MVS DISPLAY REQUESTS		
JES3 Command	**Display Command Equivalent**	**Differences**
*I R,main - Displays all outstanding action messages associated with the named processor. Messages that are issued by a specific JES3 DSP via the JES3 MESSAGE macro are not displayed by this command.	D R,L,SYS=main	All messages issued from the named system are displayed by the D R command.
*I R,dspname - Displays all outstanding action messages associated with the named JES3 DSP.	D R,L,KEY=dspname	No major differences.
*I R,SETUP - Displays all outstanding action messages issued by the JES3 Setup DSP (e.g. mount messages for a job).	D R,I,KEY=MOUNT	No major content differences. Change in the KEY name from prior releases.
*I R,SETUP,J=jobno - Displays all outstanding action messages issued by the JES3 Setup DSP for a specific job.	D R,I,KEY=MOUNT, JOB=jobname	The D R command uses job name rather than job number.
*I R,SETUP,C=Snn - Displays all outstanding action messages for the designated setup message destination class.	D R,I,ROUT=nnn	The route code equivalent of the JES3 destination class must be used on the Display command.
*I R,J=? - Displays the job numbers of the three jobs with the most action messages currently being retained.	none	No equivalent Display command
*I R,J=jobno - Displays the number of action messages currently being retained for the specified job	none	No equivalent Display command

Beginning with MVS/ESA SP5.2, if the issuing console has master authority, the system will display all WTORs in response to a D R command; even if the issuing console did not see those WTORs when originally issued. This allows the issuer to see all the WTORs which may be responded to at that master authority console.

If the operator at any console wishes to receive all action messages, the parameter CN=(ALL) may be specified on the D R command.

JES3 SETUP messages IAT5210 and IAT5310 are now issued with a WTO KEY of MOUNT, rather than SETUP. Any use of the D R,L,KEY=SETUP command to display these messages must be changed to D R,L,KEY=MOUNT. This form of the command will include the tape mount messages from MVS Allocation, and other users of the MOUNT key. (DFSMS/MVS uses the MOUNT key beginning with the DFSMS/MVS 1.3 release.)

Software developers are encouraged to use the WTO KEY of MOUNT for all tape mount related action messages and WTORs.

8.8.4 Activating the MVS Action Message Retention Facility (AMRF)

JES3 no longer maintains a queue of action messages retrievable by the *I R command. The *I R command is no longer supported. Instead, the MVS DISPLAY REQUESTS (D R) command is used with JES3 5.2.1 to retrieve action messages. MVS maintains a sysplex-wide queue of action messages using the Action Message Retention Facility (AMRF). The activation and content of AMRF is controlled using the CONSOLxx and MPFLSTxx members of SYS1.PARMLIB and via operator command.

8.8.4.1 Migration Actions

If not already active, AMRF should be activated with the migration to JES3 5.2.1. AMRF is activated via the CONSOLxx INIT AMRF(Y) parmlib specification or via the CONTROL M,AMRF=Y command.

By default, AMRF retains messages with descriptor codes 1 (System Failure), 2 (Immediate Action), 3 (Eventual Action), or 11 (Critical Eventual Action) whereas JES3 retains messages with descriptor codes 1 or 2. The message types retained by AMRF can be tailored using the MPFLSTxx RETAIN keyword. To set up AMRF to retain a similar set of messages as JES3 formerly retained, include the following statement in your MPFLSTxx member of SYS1.PARMLIB:

```
.NO_ENTRY RETAIN(I)
```

With this statement, AMRF will retain messages with descriptor code 1 (System Failure), or 2 (Immediate Action).

Notes:

1. AMRF also retains action messages issued when JES3 is not initialized, and action messages issued from programs running under MASTER rather than JES3.
2. Outstanding WTOR requests are always retrievable using the D R command.

The sequence numbering of action messages changes when the MCS sysplex is active. As a result, it may be necessary to specify a larger range on the K C command when deleting messages from an MCS console (e.g. K C,A,1-999999).

8.8.5 Using JES3 Command Prefixes

JES3 5.2.1 makes use of the MVS Command Prefix Facility (CPF) to register its command prefixes. Thus, JES3 commands may be entered from anywhere in the sysplex and the response will be returned to the issuing console. Installation-written code to direct commands to the JES3 global processor may be simplified as a result of this support.

JES3 5.2.1 allows definition of multi-character sysplex- and system-scoped command prefixes. Use of sysplex-scoped prefixes is recommended as it allows for command interaction with the JES3 global processor from any operator's console in the sysplex, regardless of physical or logical attachment. Communication with a specific local processor is achieved with a system-scoped prefix. All JES3 processors share a common set of system-scoped prefixes.

The default sysplex-scoped prefix is *. The default system-scoped prefix is 8. Thus, using the default prefixes, issuing the *I Q command displays the JES3 job queue at any console in the sysplex; and ROUTE SY2,8RETURN terminates JES3 on SY2 from any console in the sysplex. At a console associated with SY2, issuing 8RETURN would terminate JES3 on that system since 8 is a system-scoped prefix.

Another means of directing commands to the correct processor is to make use of the IBM-supplied IEECMDPF sample program. This program, when run on each system in the sysplex, registers that system's name as a SYSPLEX scoped CPF prefix. This method provides the customer an option of performing all JES3 communication with a single prefix. For example, if * is a sysplex-scoped prefix, *command can be issued from anywhere in the sysplex to direct a command to the JES3 global, and SYx*command can be issued from anywhere in the sysplex to direct a command to JES3 on SYx. Note that even though * is a sysplex-scoped prefix, a command issued as SYx*command is executed on SYx because CPF processing is only applied once per command.

You may also define * as a system-scoped prefix for greater compatibility with prior releases, however this is not recommended.

JES3 will ensure that * is always available for communication with JES3, as either a system- or sysplex-scoped prefix.

The D OPDATA command may be used to display JES3 system- and sysplex-scoped command prefix characters, as well as the command prefixes in use by other subsystems.

Note: To avoid conflicts with short-form WTOR reply, the prefix 8 is not registered with CPF. Therefore, prefix 8 will not appear in the D OPDATA response.

8.8.5.1 Migration Actions

JES3 5.2.1 will use any command prefixes specified on the CONSTD SYN= parameter of your current initialization stream as system-scoped prefixes. In addition, JES3 will default to using the * as a sysplex-scoped prefix. If you wish to change the command prefixes used by JES3, or their scope, use the CONSTD SYN= and PLEXSYN= parameters in your initialization stream.

Review automation scripts and provide operator training for the new sysplex command prefix support in JES3 5.2.1.

Determine if your programs can now execute commands on the JES3 global processor through use of MGCRE with a sysplex-scoped command prefix, rather than relying on other means (such as SSISERV) to direct the command to the global.

8.8.6 Automation Considerations for JES3 Command Responses

JES3 5.2.1 preserves the command and response token (CART) value and uses the appropriate descriptor code to identify *INQUIRY and *MODIFY command responses.

8.8.6.1 Migration Actions

Automation script improvements may be possible as a result of this change.

8.8.7 Confirming JES3 Operator-Initiated Termination

JES3 5.2.1 adds confirmation WTORs (IAT3807 and IAT3808) to *RETURN and *DUMP processing to confirm that the proper JES3 system is being terminated. This was done to ensure that the operator did not inadvertently select the wrong command prefix for the intended system.

8.8.7.1 Migration Actions

Automation scripts and operator training should be reviewed as a result of this change.

8.8.8 Using Longer Command Lengths

JES3 5.2.1 allows the *I U, *F U, and *X JMF commands to be up to 126 characters long. 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.

8.8.8.1 Migration Actions

Determine whether use of the longer command lengths is of benefit to your installation. Improvements to automation and operator productivity may be possible as a result of this change.

8.8.9 Controlling RJP Workstation Consoles

Prior to JES3 5.2.1, a system operator could not reroute console message traffic destined for one RJP workstation to another RJP workstation in the event of a problem. In addition, an operator could not purge the queue of console messages for an RJP workstation when the backlogged messages were unnecessary and were presenting a problem.

JES3 5.2.1 allows a system operator with sufficient authority to reroute message traffic from one RJP workstation to another using the *SWITCH command. A system operator with sufficient authority can also purge backlogged messages for an RJP workstation console using the *FREE,rjpcon command. In addition, an RJP user with sufficient authority can purge backlogged messages for his own console using the *FREE command.

8.8.9.1 Migration Actions

Instruct your operations staff on the use of the *SWITCH command to reroute console message traffic destined for one RJP workstation to another in the event of a problem. Instruct authorized RJP users and the operations staff on the use of the *FREE command to purge the backlog of messages for an RJP workstation when those messages are unnecessary and presenting a problem.

8.8.10 Changes in the JES3 DLOG Content

JES3 5.2.1 changes some portions of the JES3 DLOG content. These changes are:

- 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.

8.8.10.1 Migration Actions

Update your DLOG browsers and post processors if they are affected by the changes in log content.

8.8.11 Changes Affecting JES3 Command Entry from a BDT Session

Support for JES3 commands entered from a BDT session is removed in JES3 5.2.1.

8.8.11.1 Migration Actions

The TSO CONSOLE operator facility provides an alternative means of interacting with JES3 from a TSO session.

8.8.12 Sharing JES3 Commands and Source JCL for Demand Select Jobs

JES3 5.2.1 allows two or more systems to share JES3 commands and demand select jobs, while retaining unique values where required. If all systems in a JES3 complex share JES3 commands and demand select jobs, you can view the environment as a *single image* with one point of control.

8.8.12.1 Migration Actions

System symbols represent unique values in shared commands and demand select jobs. If you plan to use system symbols, your installation must make changes to the SYS1.PARMLIB data set on MVS. Your installation might also have to change automation routines and provide education on how to use system symbols.

See the chapter that describes migration actions for initialization and tuning and operations in *MVS/ESA SP V5 Conversion Notebook* for more information.

8.9 Checklist of Migration Actions for Problem Determination and Diagnosis of JES3 5.2.1

__ Be aware that JES3 5.2.1 removes the DOM and certain console-related information from the JMF report.

8.10 JES3 5.2.1 - Problem Determination and Diagnosis

JES3 5.2.1 introduces problem determination and diagnosis changes in the following areas:

- Changes in JMF Reports

8.10.1 Changes in JMF Reports

The Delete Operator Message (DOM) SSI Response report is deleted from JMF, because JES3 no longer sits on the DOM SSI. With the removal of JES3 operator consoles and certain console cell pools, data no longer applicable is removed from the JMF report. The affected report sections include the FCT Analysis Report, the FCT Summary and Highlight Report, and the Resqueue Cellpool and Control Block Utilization Report.

These changes are also reflected in the SMF Type 84 record, subtypes 1, 2, 4, and 8.

8.10.1.1 Migration Actions

Be aware that JES3 5.2.1 removes DOM and certain console-related information from the JMF report.

8.11 Implementation Considerations for JES3 5.2.1

Activities in the implementation phase include:

- Install the JES3 5.2.1 fallback PTF on all current JES3 systems. Migrate to JES3 5.2.1 through a complex warmstart.

  No initialization stream changes are required when migrating to JES3 5.2.1. Warning messages are issued for initialization statements and initialization parameters that are not supported in JES3 5.2.1. JES3 5.2.1 uses default values for all new parameters when they are not specified in the initialization stream.

- If a fallback to the pre-JES3 5.2.1 system (JES3 3.1.3 or above) is required, this may be accomplished through a complex warmstart. The same initialization stream may be used.

- When fallback is no longer a possibility, remove from the initialization stream all statements and keywords that are no longer supported by JES3 5.2.1.

8.12 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.

8.12.1 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. See 3.6.2, “JES3 XCF Group Name” on page 42 for the details. The XCF groupnames for the sample sysplex configuration shown in Figure 55 on page 150 are WTSCPLX3 and WTSCPLX4.

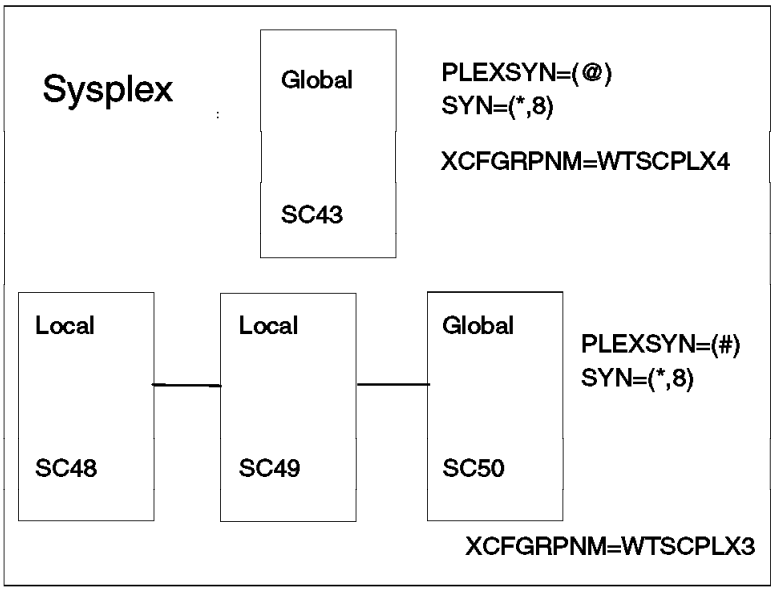


Figure 55. Multiple Globals in the Same Sysplex

8.12.2 JES3 Command Prefix Considerations

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.

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 which 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       SC48       SYSTEM    NO       PURGE
          @          JES3       SC43       SYSPLEX   NO       PURGE
          *          JES3       SC43       SYSTEM    NO       PURGE
```

Figure 56. Display of Prefixes for the Sysplex

8.12.3 JES3 Message Considerations

Most messages issued by JES3 on a global processor do not include the system name message prefix when displayed on MCS consoles defined with MFORM=(S,X). This is because the messages relate to the JES3 complex as a whole, and the system name prefix would consume the limited screen space needed for the JES3 message text and tabular displays. If an MCS console is receiving unsolicited messages from multiple JES3 globals, it may be difficult to determine the issuing system, and thus the correct command prefix to be used in commands related to the unsolicited messages. The MSCOPE console attribute can be used to limit the unsolicited messages received by the MCS console to only those systems in a particular JES3 complex. Be aware however, that this will affect the routing of all unsolicited message traffic to the console, not just the routing of JES3 messages.

8.12.4 JES3 DLOG Considerations

JES3 uses the MAINPROC names defined in the initialization stream to determine which systems to include in the DLOG produced by that JES3 global/local complex. This means that depending on how your initialization streams are set up, each DLOG will contain the hardcopy log for its own JES3 complex, and may include the hardcopy traffic from other global/local systems in the sysplex.

For example, if you bring up complex WTSCPLX3 and activate the DLOG, systems SC50, SC49, and SC48 would be in the same DLOG, assuming no other JES3 mains are defined in the WTSCPLX3 initialization stream.

If you choose to include the WTSCPLX4 messages in the DLOG of WTSCPLX3, the JES3 main names of the WTSCPLX4 complex must be defined to the WTSCPLX3 complex by defining a MAINPROC and DEVICE statement for each system in the initialization stream of WTSCPLX3.

The extended MCS console that is created for the DLOG has a console name of SYSJ3Dxx and the key is the XCF groupname. To determine the xx value of your DLOG EMCS, issue the following command:

```
D C,KEY=WTSCPLX3
 IEE892I 09.36.23 CONSOLE DISPLAY 464
 MSG: CURR=7    LIM=1500  RPLY: CURR=4    LIM=999  KEY=WTSCPLX3
   NAME       NAME       NAME       NAME       NAME       NAME       NAME
 SYSJ3R01  SYSJ3N01  SYSJ3D01
```

SYSJ3D01 is the extended MCS console name of the DLOG.

The MSCOPE value for the DLOG console shows which systems' hardcopy message traffic will be included in the DLOG.

```
D C,CN=SYSJ3D01
 IEE889I 13.34.51 CONSOLE DISPLAY 764
 MSG: CURR=10   LIM=1500 RPLY:CURR=4    LIM=999  SYS=SC50      PFK=NONE
  CONSOLE/ALT        ID  --------------- SPECIFICATIONS --------------
  SYSJ3D01          ---  COND=A      AUTH=MASTER       UD=N
   WTSCPLX3              MFORM=M          LEVEL=ALL
   SC50                  ROUTCDE=NONE
                         CMDSYS=SC50
                         MSCOPE=SC50,SC49,SC48,SC43
```

Hardcopy message traffic for systems ***SC50, SC49, SC48, and SC43*** will be included in the DLOG produced by WTSCPLX3.

8.12.5 Automatic Restart Manager (ARM) Considerations

The Automatic Restart Manager (ARM) uses JES3 services to register and restart work elements. As a result, an element's registering and restarting systems must be within the same JES3 complex.

Appendix A. Sample Program Using JESXCF Services

The JES3 local monitor DSP is activated with a *CALL dsp_name operator command and is terminated with a *CANCEL dsp_name command. The DSP may be tested using the TST DSP already defined in the IATGRPT DSP dictionary module. The TST DSP driver module has the name IATST00, and the data CSECT name is IATSTD0.

The installation of the JES3 local monitor consists of the IATUSMN and IATUSMD assemblies (the IATYUSM macro must be available in the assembly) followed by the link-edit of the objects into the SYS1.JES3LIB.

A.1 JES3 Local Monitor Driver - IATUSMN

Note: Source code for both IATUSMN and IATYUSM has statements flagged with @521 and @511. If you are installing this sample code on a JES3 5.1.1 system, delete the @521 lines and vice versa. The @521 and @511 lines show how a DSP could be changed to support the new console environment introduced in JES3 5.2.1.

```
USMN     TITLE 'IATUSMN - JES3 5.1.1 LOCAL MONITOR'
IATUSMN  AMODE 31
IATUSMN  RMODE ANY
         IATYASM
*-------
*
*        A sample JES3 DSP that uses JES XCF services to monitor
*        JES3 local systems.  The DSP monitors system events from
*        JES XCF.  When an XCF "system reported gone" is received for
*        a local JES3 main, the local is flushed (using the *START
*        local_name FLUSH operator command).  The DSP varies a local
*        main online to the global (using the *VARY local_name ON
*        operator command) when an XCF "system reported active"
*        system event is received.  The *VARY local_name ON is
*        required because the flush processing varies the local main
*        offline.
*
*        Link-Edit Attributes: RENT REUS REFR
*-------
IATUSMN  START 0
         PRINT NOGEN
* ====== If you are on JES3 511, delete the @521 lines           @521
**       TITLE 'IATYCNDB - CONSOLE DESTINATION BLOCK'            @521
         IATYCNDB DSECT=YES                                      @521
**       TITLE 'IATYCNS - CONSOLE BUFFER (INPUT MESSAGE) DSECT.'
         IATYCNS TYPE=(FCTQ,INPUT),CODES=YES
**       TITLE 'IATYDSP - DSP DICTIONARY DSECT.'
         IATYDSP
**       TITLE 'IATYEQU - SYMBOLIC MASK DEFINITIONS.'
         IATYEQU
**       TITLE 'IATYFSW - FAILSOFT WORK-AREA DSECT.'
         IATYFSW
**       TITLE 'IATYMPC - MAIN PROCESSOR DESCRIPTION.'
         IATYMPC
**       TITLE 'IATYREG - SYMBOLIC REGISTER DEFINITIONS.'
         IATYREG
**       TITLE 'IATYRSQ - RESQUEUE ENTRY DSECT.'
         IATYRSQ
**       TITLE 'IATYTVT - TRANSFER VECTOR TABLE DSECT.'
         IATYTVT
**       TITLE 'IATYSVT - SUBSYSTEM VECTOR TABLE FOR JES3.'
         IATYSVT
         PRINT GEN
         SPACE 3
         TITLE 'IATUSMN - JES3 LOCAL MONITOR'
*-------
IATUSMN  CSECT
         USING IATUSMN,R15         REG.15 ==> DRIVER ENTRY POINT
         USING TVTABLE,R12         REG.12 ==> TVT
         USING FCTSTART,R11        REG.11 ==> FCT
         IATYMOD BR=YES            MODULE ID.
         LR    R10,R15             ESTABLISH REG.10 AS BASE
         DROP  R15                 (THIS IS CONVENTIONAL BASE
         USING IATUSMN,R10         FOR JES3 MODULES).
         USING IATUSMND,R13        REG.13 ==> CSECT 'ENTRY POINT'
         L     R15,FCTDSPDC        REG.15 ==> DSP DICTIONARY ENTRY
         USING DSPSTART,R15        FOR IATUSMN.
         OI    DSPFLAG1,DSPNUDRV+DSPNUDAT  SET FLAGS TO FORCE
         DROP  R15                 REFRESH NEXT TIME.
* ------ JESTAE & LOGIN
         LA    R0,USMNXIT1         POINT AT JESTAE RETRY POINT.
         ST    R0,USMNMRYA         SAVE FOR RETRY.
         LR    R1,R13
         JESTAE USMNTAE,PARAM=(R1)
         LOGIN ENTER=USMNME00      LOGIN TO CONSOLES.
* ------ ENQ IATUSMN ACTIVE - PREVENT ANOTHER MONITOR FROM STARTING
         LA    R0,USMNDEQ          POINT AT JESTAE RETRY POINT.
         ST    R0,USMNMRYA         SAVE FOR RETRY.
         LA    R15,USMNMJ10        POINT AT SERVICE ROUTINE
         BALR  R14,R15             ENTER ENQ ROUTINE
         LTR   R15,R15             GOOD RC?
         BZ    USMNSMP             YES - CONTINUE
* ------ ISSUE ALREADY ACTIVE MESSAGE AND EXIT
* ====== If you are on JES3 521, delete the @511 lines           @521
         MESSAGE MF=(E,USMNMS00),CNDB=UCON   CONSOLE MESSAGE    @521
* ====== If you are on JES3 511, delete the @521 lines           @511
         MESSAGE MF=(E,USMNMS00)   CONSOLE MESSAGE              @511
         B     USMNXIT1
* ------ FIND ALL LOCAL MAIN PROCESSOR NAMES
USMNSMP  DS    0H
         L     R9,TVTSSVT
         USING SSVT,R9
         L     R9,SVTMPCDA         POINT AT MAIN PROCESSOR TABLES
         DROP  R9
         USING MPCSTART,R9
         LA    R6,USMNMMPC         POINT AT MP NAME SLOTS
USMNLMP  DS    0H
         LTR   R9,R9               ANY MPC'S LEFT?
         BZ    USMNJAT             NO - DONE HERE
```

```
         TM    MPSYSTEM,MPGLBL     THIS ONE FOR GLOBAL?
         BO    USMNNMP             YES - TRY NEXT
         MVC   0(L'USMNMMPC,R6),MPNAME SAVE LOCAL'S NAME
         LA    R6,L'USMNMMPC(,R6)  POINT AT NEXT MP NAME SLOT
USMNNMP  DS    0H
         L     R9,MPNEXT           NEXT MP
         B     USMNLMP             LOOP...
         DROP  R9
* ------ CREATE INSTALLATION DEFINED ATTACHMENT TO JESXCF
USMNJAT  DS    0H
         LA    R15,USMNMJ00        POINT AT SERVICE ROUTINE
         BALR  R14,R15             ENTER IXZXIXAT ROUTINE
         LTR   R15,R15             GOOD RC?
         BZ    USMNJMB             YES - CONTINUE
* ------ ISSUE IATU200 ERROR MESSAGE AND EXIT
         MVC   USMNMS23,=C'AT'
         BAL   R6,USMNMSG2
         B     USMNDEQ
* ------ ATTACH TO JESXCF DEFAULT MAILBOX
USMNJMB  DS    0H
         LA    R0,USMNJDT          POINT AT JESTAE RETRY POINT.
         ST    R0,USMNMRYA         SAVE FOR RETRY.
         LA    R15,USMNMJ20        POINT AT SERVICE ROUTINE
         BALR  R14,R15             ENTER IXZXIXMB ROUTINE
         LTR   R15,R15             GOOD RC?
         BZ    USMNJMC             YES - CONTINUE
* ------ ISSUE IATU200 ERROR MESSAGE AND EXIT
         MVC   USMNMS23,=C'MB'
         BAL   R6,USMNMSG2
         B     USMNJDT
* ------ CLEAR JESXCF DEFAULT MAILBOX
USMNJMC  DS    0H
         LA    R0,USMNJMD          POINT AT JESTAE RETRY POINT.
         ST    R0,USMNMRYA         SAVE FOR RETRY.
         LA    R15,USMNMJ30        POINT AT SERVICE ROUTINE
         BALR  R14,R15             ENTER IXZXIXMC ROUTINE
         LTR   R15,R15             GOOD RC?
         BZ    USMNSIM             YES - CONTINUE
* ------ ISSUE IATU200 ERROR MESSAGE AND EXIT
         MVC   USMNMS23,=C'MC'
         BAL   R6,USMNMSG2
         B     USMNJMD
* ------ IATUSMN ACTIVE MESSAGE
USMNSIM  DS    0H
* ====== If you are on JES3 521, delete the @511 lines           @521
         MESSAGE MF=(E,USMNMS10),CNDB=UCON   CONSOLE MESSAGE    @521
* ====== If you are on JES3 511, delete the @521 lines           @511
         MESSAGE MF=(E,USMNMS10)   CONSOLE MESSAGE              @511
* ------ AWAIT FOR OPERATOR MESSAGE OR JESXCF MESSAGE
USMNAWT  DS    0H
         LA    R0,USMNSECF
         AWAIT TYPE=ON,ECFMASK=USMNMSG+USMNJSX,ECFADD=(R0)
         XR    R0,R0               SET ALL X'FF'S..
         BCTR  R0,0                .INTO R0
         ICM   R0,B'1000'.=AL1(X'FF'-USMNJSX) COMPLEMENT OF JESXCF ECF
         L     R14,USMNSECF        PICK UP ECF
USMNDPT  DS    0H
         LR    R15,R14             DROP..
         NR    R15,R0              .JESXCF ECF..
         CS    R14,R15,USMNSECF    ..POSTING
         BNE   USMNDPT             TRY AGAIN...
         N     R14,USMNMED2        WAS THE POST FOR JESXCF MESSAGE?
         BC    NZERO,USMNJRM       YES - PROCESS JESXCF DATA.
         B     USMNJMD             MUST BE *C - EXIT
* ------ RECEIVE MAILBOX MESSAGE
USMNJRM  DS    0H
         LA    R15,USMNMJ40        POINT AT SERVICE ROUTINE
         BALR  R14,R15             ENTER IXZXIXRM ROUTINE
         LTR   R15,R15             GOOD RC?
         BZ    USMNMRS             YES - CONTINUE
         CH    R15,=H'4'           RC = 4?
         BC    NE,USMNMRE          NO - MUST BE AN ERROR
         CH    R0,=H'20'           RC = X'14' - NOTHING TO RECEIVE?
         BC    EQ,USMNAWT          CORRECT  - WAIT
* ------ ISSUE IATU200 ERROR MESSAGE AND KEEP GOING
USMNMRE  DS    0H
         MVC   USMNMS23,=C'RM'
         BAL   R6,USMNMSG2
         B     USMNJMD
* ------ PROCESS MAILBOX MESSAGE
USMNMRS  DS    0H
         L     R6,USMNMJDA         GET MESSAGE ADDRESS
         USING IXZYIXEN,R6
         CLC   YIXENEYE,=CL6'YIXEN '  IS THIS AN ENVELOPE?
         BC    NE,USMNJAC          NO - ACK IT ANYWAY
         TM    MESSAGE_CONTENT,SYSTEM_EVENT ?
         BZ    USMNJAC             NO - ACK IT
         LR    R7,R6
         AH    R7,MESSAGE_OFFSET   POINT AT SYSEVENT SECTION
         USING IXZYIXSE,R7
         CLC   YIXSEEYE,=CL6'YIXSE '  IS THIS AN SYSEVENT?
         BC    NE,USMNJAC          NO - ACK IT
         TM    YIXSE_TYPE,YIXSE_SYSEVENT  XCF SYSTEM EVENT?
         BZ    USMNJAC             NO - ACK IT
         LR    R8,R7
         AH    R8,YIXSE_OFFSET     POINT AT XCF DATA
         USING GEPL,R8
         MVC   USMNMS34,=C'GESYSACT' SYSTEM REPORTED ACTIVE
         CLI   GEPLTYPE,GESYSACT   SYSTEM REPORTED ACTIVE?
         BC    EQ,USMNJCL          YES - CHECK FOR LOCAL
         MVC   USMNMS34,=C'GESYSGON' SYSTEM REPORTED GONE
         CLI   GEPLTYPE,GESYSGON   SYSTEM REPORTED GONE?
         BC    EQ,USMNJCL          YES - CHECK FOR LOCAL
         MVC   USMNMS34,=C'GESYSDG ' SYSTEM DETECTED GONE
         CLI   GEPLTYPE,GESYSDG    SYSTEM DETECTED GONE?
         BC    NE,USMNJAC          NO - ACK IT
USMNJCL  DS    0H
         MVC   USMNMS33,GEPLSYS    SAVE SYSTEM NAME
         LA    R5,USMNMMPC         POINT AT MP NAME SLOTS
USMNLNL  DS    0H
         CLI   0(R5),X'FF'         END OF LOCAL NAMES?
         BE    USMNLGM             YES - ISSUE SYSTEM GONE MESSAGE
         CLC   0(L'USMNMMPC,R5),USMNMS33 JES3 LOCAL'S NAME
         BE    USMNLFL             YES - *S LOCAL FLUSH / *V LOCA ON
         LA    R5,L'USMNMMPC(,R5)  POINT AT NEXT MP NAME SLOT
         B     USMNLNL             LOOP...
* ------ DETERMINE ACTION FOR SYSTEM (DETECTED) GONE & SYSTEM ACTIVE
USMNLFL  DS    0H
         CLI   GEPLTYPE,GESYSACT   SYSTEM REPORTED ACTIVE?
         BC    EQ,USMNLVO          YES - VARY IT ON
         CLI   GEPLTYPE,GESYSGON   SYSTEM REPORTED GONE?
         BC    NE,USMNLGM          NO - JUST ISSUE INFO MESSAGE
         DROP  R6,R7,R8
* ------ INTERCOM *S LOCAL FLUSH (FLUSH NEEDS TO BE ISSUED TWICE)
*        MVC   USMNMCM2,USMNMS33   SYSTEM NAME
         MVI   USMNMCM2,C' '               TEMPORARY..
         MVC   USMNMCM2+1(L'USMNMCM2-1),USMNMCM2 .FIX..
         LA    R2,L'USMNMCM2               ..TO..
         LA    R14,USMNMS33+L'USMNMS33-1   ...*S MAIN FLUSH..
         LA    R15,USMNMCM2+L'USMNMCM2-1   ....WHICH..
USMNLF1  DS    0H                          .....CURRENTLY..
         CLI   0(R14),C' '                 ......CANNOT..
         BE    USMNLSB                     .......STEP..
         MVC   0(1,R15),0(R14)             ........OVER..
         BCTR  R15,0                       .........BLANKS..
USMNLSB  DS    0H                          ..........AFTER..
         BCTR  R14,0                       ...........MAIN..
         BCT   R2,USMNLF1                  ............NAME
         LA    R2,2                LOOP COUNT
USMNLF2  DS    0H
* ====== If you are on JES3 521, delete the @511 lines           @521
         INTERCOM TEXT=USMNMCM0,MSG=NO (USE CONS ID 0 - INTERNAL) @521
* ====== If you are on JES3 511, delete the @521 lines           @511
         INTERCOM CONS=DUMMY,                                   @511 *
               TEXT=USMNMCM0,MSG=NO                             @511
         BCT   R2,USMNLF2          LOOP...
         B     USMNLGM
* ------ INTERCOM *V LOCAL ON    (AS FLUSH VARIED IT OFF)
USMNLVO  DS    0H
* ====== If you are on JES3 521, delete the @511 lines           @521
         MVC   USMNMCD2,USMNMS33   SYSTEM NAME
         INTERCOM TEXT=USMNMCM0,MSG=NO (USE CONS ID 0 - INTERNAL) @521
* ====== If you are on JES3 511, delete the @521 lines           @511
         INTERCOM CONS=DUMMY,                                   @511 *
               TEXT=USMNMCD0,MSG=NO                             @511
```

```
         B     USMNLGM
* ------ SYSTEM GONE MESSAGE
USMNLGM  DS    0H
* ====== If you are on JES3 521, delete the @511 lines           @521
         MESSAGE MF=(E,USMNMS30),CNDB=UCON   CONSOLE MESSAGE    @521
* ====== If you are on JES3 511, delete the @521 lines           @511
         MESSAGE MF=(E,USMNMS30)   CONSOLE MESSAGE              @511
* ------ ACKNOWLEDGE THE RECEIVED MESSAGE
USMNJAC  DS    0H
         LA    R15,USMNMJ50        POINT AT SERVICE ROUTINE
         BALR  R14,R15             ENTER IXZXIXAC ROUTINE
         LTR   R15,R15             GOOD RC?
         BZ    USMNJRM             YES - TRY TO RECEIVE MORE
* ------ ISSUE IATU200 ERROR MESSAGE AND KEEP GOING
         MVC   USMNMS23,=C'AC'
         BAL   R6,USMNMSG2
         B     USMNJRM
* ------ DETACH FROM JESXCF DEFAULT MAILBOX
USMNJMD  DS    0H
         LA    R0,USMNJDT          POINT AT JESTAE RETRY POINT.
         ST    R0,USMNMRYA         SAVE FOR RETRY.
         LA    R15,USMNMJ70        POINT AT SERVICE ROUTINE
         BALR  R14,R15             ENTER IXZXIXMD ROUTINE
         LTR   R15,R15             GOOD RC?
         BZ    USMNJDT             YES - CONTINUE
* ------ ISSUE IATU200 ERROR MESSAGE AND CONTINUE TERMINATION
         MVC   USMNMS23,=C'MD'
         BAL   R6,USMNMSG2
         B     USMNJDT
* ------ DELETE INSTALLATION DEFINED ATTACHMENT TO JESXCF
USMNJDT  DS    0H
         LA    R0,USMNDEQ          POINT AT JESTAE RETRY POINT.
         ST    R0,USMNMRYA         SAVE FOR RETRY.
         LA    R15,USMNMJ80        POINT AT SERVICE ROUTINE
         BALR  R14,R15             ENTER IXZXIXDT ROUTINE
         LTR   R15,R15             GOOD RC?
         BZ    USMNDEQ             YES - CONTINUE
* ------ ISSUE IATU200 ERROR MESSAGE AND CONTINUE TERMINATION
         MVC   USMNMS23,=C'DT'
         BAL   R6,USMNMSG2
* ------ DEQ IATUSMN ACTIVE
USMNDEQ  DS    0H
         LA    R0,USMNXIT1         POINT AT JESTAE RETRY POINT.
         ST    R0,USMNMRYA         SAVE FOR RETRY.
         LA    R15,USMNMJ90        POINT AT SERVICE ROUTINE
         BALR  R14,R15             ENTER DEQ ROUTINE
* ------ CLEANUP, LOGOUT & OTHER RETURN PROCESSING
USMNXIT1 DS    0H
         LA    R0,USMNXIT2         POINT AT JESTAE RETRY POINT.
         ST    R0,USMNMRYA         SAVE FOR RETRY.
* ====== If you are on JES3 521, delete the @511 lines           @521
         MESSAGE MF=(E,USMNMS80),CNDB=UCON   CONSOLE MESSAGE    @521
* ====== If you are on JES3 511, delete the @521 lines           @511
         MESSAGE MF=(E,USMNMS80)   CONSOLE MESSAGE              @511
USMNXIT2 DS    0H
         JESTAE 0
         LOGOUT
         SR    R15,R15             SET RETURN CODE TO JSS.
         L     R14,JSSRETRN        JSS RETURN POINT.
         BR    R14                 RETURN TO JSS.
* ------ SUBROUTINE: ISSUE IATU200 ERROR MESSAGE
USMNMSG2 DS    0H
         UNPK  USMNMS24(5),USMNMJRR+2(3)
         TR    USMNMS24(4),USMNMH2C-C'0'
         MVI   USMNMS25,C'/'
         UNPK  USMNMS26(5),USMNMJRR+6(3)
         TR    USMNMS26(4),USMNMH2C-C'0'
         MVI   USMNMS27,C' '
* ====== If you are on JES3 521, delete the @511 lines           @521
         MESSAGE MF=(E,USMNMS20),CNDB=UCON   CONSOLE MESSAGE    @521
* ====== If you are on JES3 511, delete the @521 lines           @511
         MESSAGE MF=(E,USMNMS20)   CONSOLE MESSAGE              @511
         BR    R6
* ------ JESTAE ROUTINE
USMNTAE  DS     0H
         USING  FSWSTART,R1
         L      R10,USMNBASE-USMNTAE(,R15) PICK UP BASE ADDR.
         ST     R10,FSWSR10       SET THE BASE FOR RETRY
         L      R13,FSWPARAM      PT AT DATA CSECT
         ST     R13,FSWSR13       SET DATA CSECT ADDR FOR RETRY
         L      R0,USMNTRYA-USMNTAE(,R15) PICK UP RETRY ADDR.
         LA     R15,4             REQUEST RETRY.
         BR     R14               ATTEMPT RETRY
         DROP   R1
USMNBASE DC     A(IATUSMN)
USMNTRYA DC     A(USMNTRY)
* ------ JESTAE RETRY ROUTINE
USMNTRY  DS     0H
* ====== If you are on JES3 521, delete the @511 lines           @521
         MESSAGE MF=(E,USMNMS90),CNDB=UCON   CONSOLE MESSAGE    @521
* ====== If you are on JES3 511, delete the @521 lines           @511
         MESSAGE MF=(E,USMNMS90)   CONSOLE MESSAGE              @511
         L     R15,USMNMRYA        POINT AT RETRY ADDRESS
         BR    R15                 ENTER RETRY
*-------
         IATXPTCH LT
         DROP  R10
*-------
         TITLE 'IATUSMD - DATA CSECT MAPPING.'
IATUSMND DSECT
         IATYUSM
IATUSMN  CSECT
APARNUM  DC   CL8' '
PTFNUM   DC   CL8' '
         END  IATUSMN
```

A.2 JES3 Local Monitor Data Csect - IATUSMD

```
USMD     TITLE 'IATUSMD - JES3 LOCAL MONITOR DATA CSECT'
IATUSMD  AMODE 31
IATUSMD  RMODE ANY
         IATYASM
         PRINT NOGEN
**       TITLE 'IATYCNS - CONSOLE BUFFER (INPUT MESSAGE) DSECT.'
         IATYCNS TYPE=(FCTQ,INPUT),CODES=YES
**       TITLE 'IATYEQU - SYMBOLIC MASK DEFINITIONS.'
         IATYEQU
**       TITLE 'IATYREG - SYMBOLIC REGISTER DEFINITIONS.'
         IATYREG
**       TITLE 'IATYTVT - TRANSFER VECTOR TABLE DSECT.'
         IATYTVT
**       TITLE 'IATYSVT - SUBSYSTEM VECTOR TABLE FOR JES3.'
         IATYSVT
         PRINT GEN
IATUSMD  CSECT
         IATYUSM
IATUSMD  CSECT
APARNUM  DC    CL8' '
PTFNUM   DC    CL8' '
         END   IATUSMD
```

A.3 JES3 Local Monitor Macro - IATYUSM

```
         MACRO
         IATYUSM
         IATYMOD BR=NO,ID=IATUSMD,                                    *
               ENVIRON=J3MAINTASK
* ------ CONSOLE MESSAGE APPENDAGE
         USING TVTABLE,R12
         USING CONSMESS,R1
USMNME00 DS    0H
         LR    R10,R15             SET REG.10 AS BASE.
         USING USMNME00,R10
         LA    R15,8               RC=08 TO REJECT MESSAGE.
         CLI   CONACTN,CNCANCEL    *C REQUESTED?
         BC    NE,USMNME03         NO - IGNORE IT
USMNME01 DS    0H
         LA    R15,USMNSECF        POINT AT ECF.
         L     R2,0(0,R15)         ECF.
USMNME02 LR    R3,R2               POST ECF USING COMPARE
         O     R3,USMNMED1         AND SWAP.
         CS    R2,R3,0(R15)
         BC    NE,USMNME02
         LA    R15,4               RC=04 TO QUEUE MESSAGE.
         USING FCTSTART,R11
USMNME03 ARETURN RC=(R15)
         DROP  R1,R10,R11
         LTORG
* ------ ATTACH TO JESXCF GROUP
USMNMJ00 DS    0H
         LR    R6,R15              SET REG.6 AS BASE.
         USING USMNMJ00,R6
         LR    R8,R14              SAVE RETURN ADDRESS
         L     R9,TVTSSVT
         USING SSVT,R9
         LA    R7,SVTYMOD          POINT AT MODULE IDENTIFIER
         USING MODSTART,R7
         IXZXIXAT GROUP=USMNMJGR,  ATTACH JES3 TO JESXCF              *
               MEMBER=USMNMJMB,    MEMBER NAME IS MAIN NAME           *
               WHICHJES=JES3,                                         *
               RELEASE=MODREL,     CURRENT RELEASE FROM TVT           *
               GROUPTOKEN=SVTRSVU4 GROUP TOKEN IS TO GO HERE
         STM   R15,R0,USMNMJRR     SET RETURN/REASON CODES
         LR    R14,R8
         BR    R14
         DROP  R6,R7,R9
* ------ ENQ IATUSMN ACTIVE - ONLY ONE IATUSMN CAN BE ACTIVE
USMNMJ10 DS    0H
         LR    R6,R15              SET REG.6 AS BASE.
         USING USMNMJ10,R6
         LR    R8,R14              SAVE RETURN ADDRESS
         ENQ   (USMNMJGR,USMNMJMB,E,,STEP),RET=USE
         ST    R15,USMNMJRR        SET RETURN CODE
         LR    R14,R8
         BR    R14
         DROP  R6
* ------ ATTACH TO THE JESXCF DEFAULT MAILBOX.
USMNMJ20 DS    0H
         LR    R6,R15              SET REG.6 AS BASE.
         USING USMNMJ20,R6
         LR    R8,R14              SAVE RETURN ADDRESS
         L     R9,TVTSSVT
         USING SSVT,R9
         IXZXIXMB MBOXNAME=USMNMDMB, ATTACH TO DEFAULT MAILBOX        *
               POSTXIT=USMNMJXP,   EXIT ADDRESS                       *
               POSTDATA=TVTSSVT,   EXIT DATA IS DATA CSECT ADDRESS    *
               SYSEVENT=YES,       WANT TO SEE JESXCF SYSEVENTS       *
               GROUPTOKEN=SVTRSVU4  GROUP TOKEN IS HERE
         STM   R15,R0,USMNMJRR     SET RETURN/REASON CODES
         LR    R14,R8
         BR    R14
         DROP  R6,R9
* ------ CLEAR THE JESXCF DEFAULT MAILBOX.
USMNMJ30 DS    0H
         LR    R6,R15              SET REG.6 AS BASE.
         USING USMNMJ30,R6
         LR    R8,R14              SAVE RETURN ADDRESS
         L     R9,TVTSSVT
         USING SSVT,R9
         IXZXIXMC MBOXNAME=USMNMDMB, ATTACH TO DEFAULT MAILBOX        *
               GROUPTOKEN=SVTRSVU4  GROUP TOKEN IS HERE
         STM   R15,R0,USMNMJRR     SET RETURN/REASON CODES
         LR    R14,R8
         BR    R14
         DROP  R6,R9
* ------ RECIVE DATA FROM JESXCF DEFAULT MAILBOX.
USMNMJ40 DS    0H
         LR    R6,R15              SET REG.6 AS BASE.
         USING USMNMJ40,R6
         LR    R8,R14              SAVE RETURN ADDRESS
         L     R9,TVTSSVT
         USING SSVT,R9
         IXZXIXRM MBOXNAME=USMNMDMB, DELETE THE DEFAULT MAILBOX       *
               MSGFETCH=ALL,        SELECT EVERYTHING                 *
               DATA=USMNMJDA,       DATA ADDRESS                      *
               DATALEN=USMNMJDL,    DATA LENGTH                       *
               MSGTOKEN=USMNMJMT,   MESSAGE TOKEN                     *
               GROUPTOKEN=SVTRSVU4  GROUP TOKEN IS HERE
         STM   R15,R0,USMNMJRR     SET RETURN/REASON CODES
         LR    R14,R8
         BR    R14
         DROP  R6,R9
* ------ ACKNOWLEDGE JESXCF MESSAGE.
USMNMJ50 DS    0H
         LR    R6,R15              SET REG.6 AS BASE.
         USING USMNMJ50,R6
         LR    R8,R14              SAVE RETURN ADDRESS
         L     R9,TVTSSVT
         USING SSVT,R9
         IXZXIXAC MSGTOKEN=USMNMJMT,   MESSAGE TOKEN                  *
               GROUPTOKEN=SVTRSVU4  GROUP TOKEN IS HERE
         STM   R15,R0,USMNMJRR     SET RETURN/REASON CODES
         LR    R14,R8
         BR    R14
         DROP  R6,R9
* ------ DETACH FROM THE JESXCF DEFAULT MAILBOX.
USMNMJ70 DS    0H
         LR    R6,R15              SET REG.6 AS BASE.
         USING USMNMJ70,R6
         LR    R8,R14              SAVE RETURN ADDRESS
         L     R9,TVTSSVT
         USING SSVT,R9
         IXZXIXMD MBOXNAME=USMNMDMB, DELETE THE DEFAULT MAILBOX       *
               GROUPTOKEN=SVTRSVU4  GROUP TOKEN IS HERE
         STM   R15,R0,USMNMJRR     SET RETURN/REASON CODES
         LR    R14,R8
         BR    R14
         DROP  R6,R9
* ------ DETACH FROM JESXCF GROUP.
USMNMJ80 DS    0H
         LR    R6,R15              SET REG.6 AS BASE.
         USING USMNMJ80,R6
         LR    R8,R14              SAVE RETURN ADDRESS
         L     R9,TVTSSVT
         USING SSVT,R9
         IXZXIXDT GROUPTOKEN=SVTRSVU4 GROUP TOKEN IS HERE
         STM   R15,R0,USMNMJRR     SET RETURN/REASON CODES
         LR    R14,R8
         BR    R14
         DROP  R6,R9
* ------ DEQ IATUSMN ACTIVE.
USMNMJ90 DS    0H
         LR    R6,R15              SET REG.6 AS BASE.
         USING USMNMJ90,R6
         LR    R8,R14              SAVE RETURN ADDRESS
         DEQ   (USMNMJGR,USMNMJMB,,STEP)
         ST    R15,USMNMJRR        SET RETURN CODE
         LR    R14,R8
         BR    R14
         DROP  R6
* ------ JESXCF POST EXIT SUBROUTINE.
*        POST THE IATUSMN ECF ASSOCIATED WITH THE MAILBOX.
*        THIS ROUTINE DOES NOT RUN UNDER THE JES3 MAIN TASK,
```

```
*        BUT DOES RUN IN THE JES3 ADDRESS SPACE.
USMNMJXR DS    0H
         BAKR  R14,0               SAVE CALLER'S REGISTERS
         LR    R10,R15             SETUP SUBROUTINE'S
         USING USMNMJXR,R10           ADDRESSABILITY
         L     R5,0(,R1)           GET ADDRESSABILITY TO THE
         USING IXZYIXPE,R5           POST EXIT PARAMETER LIST
*------- VALIDATE THE INPUT PARAMETERS                        *
         CLC   YIXPEEYE,=CL6'YIXPE' EXIT P_LIST VALID?
         BC    NE,USMNMJX4         NO, INDICATE ERROR AND RETURN
         L     R9,YIXPE_POSTDATA   POINT AT POST_DATA - JES3 SSVT
         DROP  R5
*------- POST IATUSMN                                         *
         LA    R15,USMNSECF        POINT AT ECF.
         L     R2,0(0,R15)         ECF.
USMNMJXL LR    R3,R2               POST ECF USING COMPARE
         O     R3,USMNMED2         AND SWAP.
         CS    R2,R3,0(R15)
         BC    NE,USMNMJXL
*------- POST JES3                                            *
         USING SSVT,R9
         MODESET EXTKEY=ZERO,SAVEKEY=(2) MUST BE IN KEY 0
         IATXPJ3 ADDR=SSVT,TYPE=JPOST POST JES3
         MODESET KEYADDR=(2)       RESTORE ORIGINAL KEY
         XR    R15,R15             SET NORMAL RETURN CODE
         XR    R0,R0               SET NORMAL REASON CODE
         B     USMNMJXX            GO RETURN TO CALLER
*------- ERROR RETURN
USMNMJX4 DS    0H
         LA    R15,4               INDICATE INPUT ERROR
         LA    R0,4                INDICATE BAD PARAMETER LIST
*------- O.K. RETURN
USMNMJXX DS    0H
         PR    ,                   RETURN
         DROP  R10,R9
*-------
         LTORG
*-------
USMNMJDA DC    A(0)                MESSAGE DATA ADDRESS
USMNMJDL DC    A(0)                MESSAGE DATA LENGTH
USMNMJMT DC    XL8'0'              MESSAGE TOKEN
*-------
USMNMDMB DC    CL16'SYSJES$DEFAULT' DEFAULT MAILBOX NAME
USMNMJXP DC    A(USMNMJXR)         POST EXIT ADDRESS
*-------
USMNMJGR DC    CL8'J3LOCAL'        XCF GROUP NAME
USMNMJMB DC    CL16'PRIVATEEYE'
USMNMJRR DC    2F'0'
*-------
USMNMSG  EQU   X'80'               ECF - OPERATOR MESSAGE
USMNJSX  EQU   X'40'               ECF - JESXCF MESSAGE
USMNSECF DC    A(0)                ECF
USMNMED1 DC    AL1(USMNMSG),XL3'00' ECF MASK FOR OPERATOR MESSAGE
USMNMED2 DC    AL1(USMNJSX),XL3'00' ECF MASK FOR JESXCF MESSAGE
*-------
USMNMRYA DS    F                   RETRY ADDRESS
*-------
USMNMS00 MESSAGE MF=L,CLASS=JES,TEXT=USMNMS01
USMNMS01 DC    AL1(USMNMS09-USMNMS02)     ALREADY ACTIVE MESSAGE.
USMNMS02 DC    C'IATU110 - LOCAL MONITOR ALREADY ACTIVE'
USMNMS09 DS    0C
*-------
USMNMS10 MESSAGE MF=L,CLASS=JES,TEXT=USMNMS11
USMNMS11 DC    AL1(USMNMS19-USMNMS12)     LOGON MESSAGE.
USMNMS12 DC    C'IATU100 - LOCAL MONITOR ACTIVE'
USMNMS19 DS    0C
*-------
USMNMS20 MESSAGE MF=L,CLASS=JES,TEXT=USMNMS21
USMNMS21 DC    AL1(USMNMS29-USMNMS22)     IXZXIXNN ERROR MESSAGE.
USMNMS22 DC    C'IATU200 - IXZXIX'
USMNMS23 DC    C'XX'
         DC    C' RETURN/REASON CODE = '
USMNMS24 DC    C'XXXX'
USMNMS25 DC    C'/'
USMNMS26 DC    C'XXXX'
USMNMS27 DC    C' '
USMNMS29 DS    0C
*-------
USMNMS30 MESSAGE MF=L,CLASS=JES,TEXT=USMNMS31
USMNMS31 DC    AL1(USMNMS39-USMNMS32)     XCF SYSTEM GONE MESSAGE.
USMNMS32 DC    C'IATU300 - SYSTEM '
USMNMS33 DC    CL8' '
         DC    C'-'
USMNMS34 DC    CL8' '
USMNMS36 DC    C' RECEIVED FROM XCF'
USMNMS39 DS    0C
*-------
USMNMS80 MESSAGE MF=L,CLASS=JES,TEXT=USMNMS81
USMNMS81 DC    AL1(USMNMS89-USMNMS82)     LOGOUT MESSAGE.
USMNMS82 DC    C'IATU999 - LOCAL MONITOR TERMINATED'
USMNMS89 DS    0C
*-------
USMNMS90 MESSAGE MF=L,CLASS=JES,TEXT=USMNMS91
USMNMS91 DC    AL1(USMNMS99-USMNMS92)     ABEND MESSAGE.
USMNMS92 DC    C'IATU997 - ABEND IN LOCAL MONITOR'
USMNMS99 DS    0C
*-------
USMNMCM0 DC    AL1(USMNMCM3-USMNMCM1)
USMNMCM1 DC    C'*S '
USMNMCM2 DC    CL8' '
         DC    C' FLUSH '
USMNMCM3 DS    0C
*-------
USMNMCD0 DC    AL1(USMNMCD3-USMNMCD1)
USMNMCD1 DC    C'*V '
USMNMCD2 DC    CL8' '
         DC    C' ON '
USMNMCD3 DS    0C
*-------
USMNMH2C DC    C'0123456789ABCDEF'
USMNMMPC DC    32XL8'FFFFFFFFFFFFFFFF'
* ====== If you are on JES3 511, delete the @521 lines            @521
*-------                                                         @521
UCON     IATYCNDB DSECT=NO                                      @521
UCONRTCD DC    XL17'40000000000000000000000000000000'  DEFINE   @521
*                                ROUTE CODE 2                   @521
         DS    0D
         PUSH  PRINT
         PRINT NOGEN
*        TITLE 'IATYFCT - FUNCTION CONTROL TABLE DSECT.'
         IATYFCT
*        MODULE ENTRY POINT IDENTIFIER
         IATYMOD DSECT=YES,BR=NO
*        JESXCF MESSAGE ENVELOPE
         IXZYIXEN
*        JESXCF POST EXIT PARAMETER LIST
         IXZYIXPE
*        JESXCF SYSTEM EVENT MESSAGE
         IXZYIXSE
*        XCF GROUP EXIT PARAMETER LIST
         IXCYGEPL
         PRINT GEN
         POP   PRINT
         MEND
```

Appendix B. SYS1.PARMLIB Definitions for a Sysplex

This appendix contains the following parmlib definitions:

- IEASYS00 for the common parameters for all systems
- IEASYS50 for the specific parameters for the JES3 global
- SMFPRM00 for the SMF definitions

B.1 SYS1.PARMLIB Member IEASYS00

```
ALLOC=00,                   ALLOCATION DEFAULTS
CLOCK=00,                   TOD CLOCK INITIALIZATION
CLPA,
CMB=(UNITR,COMM,GRAPH,CHRDR), ADDITIONAL CMB ENTRIES
CON=(00),                   CONSOLE DEFINITIONS
COUPLE=00,                  COUPLE  DEFINITIONS
CSA=(2048,20480),           MVS/XA CSA RANGE
DIAG=01,                    CSA/SQA TRACING
DUMP=NO,                    DYNAMIC ALLOCATION ACTIVE (COMMND00)
FIX=00,                     FIX MODULES SPECIFIED              /*J3*/
GRS=TRYJOIN,                LETS GET THIS BABY GOING
GRSCNF=00,                  GRS CONFIG DEFINITIONS
GRSRNL=01,                  GRS RNLS DEFINITIONS
ICS=00,                     SELECT IEAICS00 INSTALL CNTL SPECS FOR SRM
LNK=00,                     SPECIFY LNKLST00
LNKAUTH=APFTAB,             LINKLIST APF AUTHORIZATION VIA APFTAB
LPA=00,                     SELECT LPALST00 CONCATENATED LPA LIBRARY
LOGCLS=Y,                   WILL NOT BE PRINTED BY DEFAULT
LOGLMT=999999,              MUST BE 6 DIGITS, MAX WTL MESSAGES QUEUED
LOGREC=SYS1.&SYSNAME..LOGREC,  DATASET NAME FOR LOGREC (WITH &SYSNAME)
MAXCAD=25,                  CICSPLEX CMAS NUMBER OF COMMON DSPACES
MAXUSER=250,                (SYS TASKS + INITS + TSOUSERS)
MLPA=02,                    SELECT IEALPA02 MODULES LOADED INTO PLPA
MSTRJCL=01,                 MSTJCL WITHOUT UADS & WITH IEFJOBS
NSYSLX=55,                  CICSPLEX CAS/ESSS LINKAGE INDEXES
OPI=YES,                    ALLOW WOL OVERRIDE TO IEASYS00
OPT=00,                     SPECIFY IEAOPT00 (SRM TUNING PARMETERS)
PAGE=(PAGE.&SYSNAME..PLPA,  PLPA PAGE DATA SET
      PAGE.&SYSNAME..COMMON, COMMON PAGE DATA SET
      PAGE.&SYSNAME..LOCAL1,L),  LOCAL PAGE DATA SET
PAGTOTL=(8,3),              ALLOW ADDITION 5 PAGE D/S AND 3 SWAP D/S
PAK=00,                     IEAPAK00
PLEXCFG=(MULTISYSTEM,OPI=NO),   MULTI-SYSTEM SYSPLEX ONLY
PROG=(00),                  DYNAMIC APF
REAL=512,                   ALLOWS 2 64K JOBS OR 1 128K JOB TO RUN V=R
RSVSTRT=25,                 RESERVED ASVT ENTRIES              DEFAULT
RSVNONR=25,                 RESERVED ASVT ENTRIES              DEFAULT
SCH=00,                     SCHEDULER LIST SCHED00
SMF=00,                     SELECT SMFPRM00, SMF PARMETERS
SMS=00,                     SMS PARAMETER
SQA=(3,18),                 MVS/XA SQA APPROX 1MB
SSN=00,                     SUBSYSTEM INITIALIZATION NAMES
SVC=00,                     SVC TABLE IEASVC00
VAL=00,                     SELECT VATLST00                    DEFAULT
VIODSN=SYS1.&SYSNAME..STGINDEX,  DATASET NAME FOR STGINDEX-DS
VRREGN=512                  DEFAULT REAL-STORAGE REGION SIZE   DEFAULT
```

B.2 SYS1.PARMLIB Member IEASYS50

```
CMD=(50,00),     COMMND(50) SC50 SPECIFIC COMMANDS,(00) COMMON COMMANDS
CON=00,
SYSNAME=SC50,                SYSTEM NAME (&SYSNAME)
LNK=(J3,AA,00),              JES3 + LNKLST00
MLPA=(AA),                   PULL IN "AA" FOR MVS 5.2
LPA=(J3,AA,00),              JES3 + LPALST00
PROG=(J3,00),                DYNAMIC AUTHORIZED PROGRAM FACILITY
SSN=J3                       SUBSYSTEM INITIALIZATION NAMES
```

B.3 SYS1.PARMLIB Member SMFPRM00

```
ACTIVE                       /*ACTIVE SMF RECORDING*/
DSNAME(SYS1.&SYSNAME..MAN1,  /*SMF DATA SET NAMES*/
       SYS1.&SYSNAME..MAN2,  /*SMF DATA SET NAMES*/
       SYS1.&SYSNAME..MAN3)  /*SMF DATA SET NAMES*/
NOPROMPT                     /*DON'T PROMPT THE OPERATOR*/
REC(PERM)                    /*TYPE 17 PERM RECORDS ONLY*/
INTVAL(10)                  /* INTERVALL RECORDING EVERY 10 MIN */
MAXDORM(3000)               /* WRITE AN IDLE BUFFER AFTER 30 MIN*/
STATUS(010000)              /* WRITE SMF STATS AFTER 1 HOUR*/
JWT(0800)                   /* 522 AFTER 2 HOURS   */
SID(&SYSNAME(1:4))          /* SYSTEM ID IS &SYSNAME  */
LISTDSN                     /* LIST DATA SET STATUS AT IPL*/
LASTDS(MSG)                 /*DEFAULT TO MESSAGE          */
NOBUFFS(MSG)                /*DEFAULT TO MESSAGE          */
SYS(TYPE(0:255),EXITS(IEFU83,IEFU84,IEFU85,IEFACTRT,IEFUJV,IEFUSI,
               IEFUJP,IEFUSO,IEFUJI,IEFUTL,IEFU29),
     NOTYPE(99),NOINTERVAL,NODETAIL)

/* WRITE ALL RECORDS AS THE SYSTEM DEFAULT, TAKE ALL KNOWN
   EXITS. THERE ARE NO DEFAULT INTERVAL RECORDS WRITTEN.
   ONLY SUMMARY TYPE 32 RECORDS ARE WRITTEN FOR TSO.
   NOTE: FOR JES2, JES EXITS ARE CONTROLLED BY JES. FOR
   JES3, THE EXITS ARE CONTROLLED BY SMF.             */

SUBSYS(STC,NOTYPE(99),
     EXITS(IEFU29,IEFU83,IEFU84,IEFU85,IEFACTRT,IEFUJI,
          IEFUJP,IEFUSO))

/* WRITE ALL RECORDS AS BY  SYSTEM DEFAULT, TAKE ONLY THREE
   EXITS, NOTE: IEFU29 EXECUTES IN THE MASTER ASID WHICH IS A
   STC ADDRESS SPACE SO IEFU29 MUST BE ON FOR STC. USE ALL OTHER
   SYS PARAMETERS AS A DEFAULT  */
```

Appendix C. Sample ISPF Dialog to Interface Operlog

C.1 Program to Browse Operlog Data

The following sample program reads records from the OPERLOG log stream, converts them to SYSLOG format, and passes the records to ISPF browse for online viewing. It also accepts operator commands (/oper_cmd) from the browse panel and attempts to execute them using the TSO CONSOLE command. The program is based on the IEAMBDLG sample program found in SYS1.SAMPLIB.

```
* START OF SPECIFICATIONS  ******************************************
*                                                                   *
*01* DESCRIPTIVE-NAME: Sample BRIF program to read records from an  *
*                      Operations Log stream, convert them to       *
*                      SYSLOG format, and pass the records to ISPF  *
*                      BROWSE.                                      *
*01* 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.            *
*01* FUNCTION:                                                     *
*      This program is an example of how to use the services of the *
*      MVS System Logger to retrieve and delete records from the    *
*      Operations Log stream.  In response to parameters, it can    *
*      read the records created in a given time span, convert       *
*      them from Message Data Block (MDB) format to Hard-copy Log   *
*      format (HCL or JES2 SYSLOG), and pass the SYSLOG-format      *
*      records to the ISPF BROWSE using BRIF interface.             *
*      The parameters are as follows:                               *
*                                                                   *
*        COPY([start_date][,end_date])                              *
*            (>nnn)                                                 *
*                                                                   *
*      COPY:  Records are to be passed to the ISPF browse.          *
*        start_date,end_date: The starting and ending dates of      *
*          the time span, both in the format YYYYDDD.               *
*        >nnn: Indicates that records dated more than nnn days      *
*          before today are to be selected. The time span will      *
*          start with the date of the oldest record in the log      *
*          stream and end nnn+1 days before today (that is,         *
*          records dated more than nnn days before today will       *
*          be selected.                                             *
*01* OPERATION:                                                    *
********************************************************************
* Initialization:                                                  *
*                                                                  *
*  If COPY was specified, get end and start dates or calculate     *
*  defaults, yesterday and "oldest" respectively.                  *
*  Obtain a buffer area for logger record and set up its base      *
*  Connect to the log stream                                       *
********************************************************************
* Copy:                                                            *
*                                                                  *
*  When COPY is specified:                                         *
*   Start a log stream browse session and position the log stream  *
*   to first record in the range                                   *
*   Copy loop:                                                     *
*    Read successive records from the stream, starting with the    *
*    earliest record bearing the start date and return the records *
*    to ISPF Browse as requested                                   *
*    For each record (MDB) that is read:                           *
*     Get the general and CP objects                               *
*     Extract the fixed info                                       *
*     For every line (text object) in the message:                 *
*      Pass a syslog-format line through the BRIF interface        *
*      If line was too long, also create a continuation line       *
*    If an operator command (indicated by a '/' in front of the    *
*    text entered on the browse primary command line on the         *
*    browse panel) is entered, the command is passed to MVS         *
*    using the TSO CONSOLE command.                                 *
*   End the log stream browse session                               *
*   Close the output file                                           *
********************************************************************
* Cleanup:                                                         *
*  Disconnect from the log stream                                  *
*  Free the buffer area                                            *
*  Exit                                                            *
********************************************************************
*01* RECOVERY OPERATION:                                          *
*    None.  This program make no attempt to recover from failures. *
*    The caller's recovery environment, if any, will remain in     *
*    effect.                                                       *
*02* LINKAGE EDITOR ATTRIBUTES                                    *
*             LINKAGE EDITOR ATTRIBUTES =  REUS                    *
*             AMODE                     =  31                      *
*             RMODE                     =  ANY                     *
****END OF SPECIFICATIONS*****************************************
IEAMDBBR CSECT ,
IEAMDBBR AMODE 31
IEAMDBBR RMODE ANY
* ------ Local macros
         MACRO
&NAME    POINTER  &C
         LCLA     &A
         LCLA     &B
&NAME    DS    0F
         ACTR  6000
&B       SETA  0
&A       SETA  1
         AIF   (T'&C EQ 'O').L
&A       SETA  &C
.L       ANOP
&B       SETA  &B+1
         DC    F'0',XL8'0',F'0'
         AIF   (&B NE &A).LC
&NAME.L  EQU   &NAME.E-*
.LC      ANOP
         AIF   (&B LE &A).L
&NAME.E  DC    XL4'FFFFFFFF'
         MEND
* ------
         MACRO
&NAME    BUFFER   &C,&L
         LCLA     &A
         LCLA     &B
&NAME    DS    0F
&B       SETA  0
&A       SETA  2
         AIF   (T'&C EQ 'O').L
&A       SETA  &C
         AIF   (&A GT 1).L
&A       SETA  2
.L       ANOP
&B       SETA  &B+1
```

```
         AIF   (&B NE 1).M
         DC    AL4(*+16+&L),AL4(&NAME.E),AL4(0),AL4(0),CL(&L)' '
         AGO   .L
.M       ANOP
         AIF   (&B EQ &A).F
         DC    AL4(*+16+&L),AL4(*-20-&L),AL4(0),AL4(0),CL(&L)' '
         AIF   (&B LT &A).L
.F       ANOP
&NAME.E  DC    AL4(&NAME),AL4(*-20-&L),AL4(0),AL4(0),CL(&L)' '
&NAME.L  EQU   *-&NAME.E
         MEND
* ------ The program
         BAKR  R14,0              save regs
         USING IEAMDBBR,R15
         MODID ,                  eye catcher and date
         L     R9,0(R1,0)          save parm addr
         LM    R11,R12,BASES      get bases
         DROP  R15
         USING IEAMDBBR,R11,R12   address them
         B     BASESOK            skip over adcons
BASES    DC    A(IEAMDBBR,IEAMDBBR+4096)
BASESOK  LA    R13,SV             point r13 to save area
         MVC   4(4,R13),=C'F1SA'  set acro in save area
* ------ Begin initialization
*  If COPY was specified, get end and start dates or calculate
*  defaults, yesterday and "oldest" respectively.
*  R9 -> parm
*  Results:
*    PFLAGS -- COPY flag set if COPY was specified,
*    SDATE - If COPY is specified, the specified starting date or
*            default to 1900001; otherwise zero
*    EDATE - If COPY is specified, the day after the specified
*            ending date or default to today;otherwise zero
*    SSTCK, and ESTCK are the same dates in STCK format.
*    COPYDAYS - If COPY(>nnn) is specified, the number of days
*            nnn; otherwise binary zero
         MVI   PFLAGS,0           clear out parm flags
         XC    SDATE,SDATE        clear out start date
         XC    EDATE,EDATE        clear out end date
         XC    COPYDAYS,COPYDAYS  clear out number of days
* -----
         LOAD  EP=ISPLINK
         LR    R15,R0             ISPLINK EP
         ST    R0,ISPLINK
* -----  Define ISPF variables
         CALL  (15),(VDEF,NMLS,VDAR,FMAR,LNAR,OPLS),VL
* -----
         LH    R3,0(R9)           length of parm
         LA    R9,2(R9)           get past length
         CH    R3,=H'0'           is there a parm?
         BE    BADPARM            no, error (parm is required)
         CH    R3,=H'256'         is it too long (for TRT)?
         BNH   PLOOP              no, ok
BADPARM  LA    R2,BADPMSG         point to parm error msg
         BAS   R14,MESSR          display it
         B     DONEDONE           return
PLOOP    DS    0H
*        loop through parameter processing each entry
*        r9 = address of remaining parm
*        r3 = length of remaining parm
         LR    R14,R9             initial starting point
PLOOPR   DS    0H
* ------ resume the scan
*        r14 = address of resume point
         LA    R1,0(R3,R9)        point past parm (in case there      X
                                  is no comma)
         LR    R15,R1             end of parm + 1
         SR    R15,R14            subtract start addr to get len
         BCTR  R15,0              subtract 1 to get machine len
         SR    R2,R2              clear reg. for character found
         EX    R15,TRT1           scan the parm; r1 will point to     X
                                  comma or lt paren or end+1 of parm
         C     R2,ZLPAREN         did it stop on left paren?
         BNE   PSCANOK            no, ok
* ------ scan to right paren
         LA    R15,0(R3,R9)       end of parm + 1
         SR    R15,R1             subtract start addr to get len
         EX    R15,TRT2           scan to right paren
         BZ    BADPARM            error if not found
         LR    R14,R1             set resume address
         B     PLOOPR             resume the scan
PSCANOK  DS    0H
         LR    R4,R1              save pointer to comma or end
         SR    R4,R9              length of this parm entry
* ------ interpret and process a parm entry
*        r9 = address of parm entry
*        r4 = length of parm entry
         CH    R4,=H'4'           is length at least 4 ("COPY")?
         BL    BADPARM            no, error
         CLC   =C'COPY'.0(R9)     is it COPY?
         BNE   BADPARM            no
         TM    PFLAGS,COPY        was COPY already processed?
         BO    BADPARM            yes,error
         OI    PFLAGS,COPY        set COPY flag
         CH    R4,=H'5'           is length 5?
         BL    PNEXT              length less than 5, must be 4,      X
                                  'COPY'. use defaults
         BE    BADPARM            length 5, error
         CLI   4(R9),C'('         does it start with left paren?
         BNE   BADPARM            no, error
         LA    R15,0(R4,R9)       end of parm + 1
         BCTR  R15,0              end of parm
         CLI   0(R15),C')'        does it end with right paren?
         BNE   BADPARM            no, error
         CH    R4,=H'6'           is it 'COPY()'
         BE    PNEXT              yes, use defaults
         CH    R4,=H'21'          are both dates given?
         BNE   PNOTBOTH           no, keep checking
         CLI   12(R9),C','        is there a comma?
         BNE   BADPARM            no, error
         MVC   SDATE,5(R9)        save start date
         MVC   EDATE,13(R9)       save end date
         B     PNEXT              look for next entry
PNOTBOTH CH    R4,=H'13'          is it start all alone?
         BE    PSTART             yes, so save it
         CH    R4,=H'14'          could it be one date w/comma?
         BNE   PCOPYND            no, must be ">nnn"
         CLI   12(R9),C','        does it end with comma?
         BE    PSTART             yes, so it's "start_date,"
         CLI   5(R9),C','         does it start with comma?
         BNE   BADPARM            no, error
         MVC   EDATE,6(R9)        save end date
         B     PNEXT              look for next entry
PSTART   MVC   SDATE,5(R9)        save start date
         B     PNEXT              look for next entry
PCOPYND  DS    0H                 must be COPY(>nnn)
         CH    R4,=H'8'           is it too short?
         BL    BADPARM            yes, error
         CH    R4,=H'10'          is it too long?
         BH    BADPARM            yes, error
         CLI   5(R9),C'>'         does it start with >?
         BNE   BADPARM            no, error
* ------ save number of days, n thru nnn
         MVC   COPYDAYS,=C'000'   initialize receiving field
         LR    R14,R4             get length of entry
         SH    R14,=H'8'          get length of number - 1
         LA    R15,COPYDAYS+2     end of receiving field
         SR    R15,R14            back up to correct position
         EX    R14,MVCCOPY        move in number of days
         B     PNEXT              look for next entry
TRT1     TRT   0(*-*,R14),TRTTAB1 scan parm for comma or l. paren
TRT2     TRT   0(*-*,R1),TRTTAB2  scan parm for r. paren
MVCCOPY  MVC   0(*-*,R15),6(R9)   move in number of days
MVCDEL   MVC   0(*-*,R15),8(R9)   move in number of days
PNEXT    DS    0H
* ------ get to next parm entry
         A     R4,=F'1'           add comma to len of this entry
         AR    R9,R4              point to next entry
         SR    R3,R4              calculate remaining length
         BP    PLOOP              loop back until parm is done
* ------ see if defaults are needed
         TIME  ,                  get today's date
         ST    1,DATEWORK         copy it
         AP    DATEWORK(4),=P'1900000' add to correct the century
         UNPK  TDATE,DATEWORK     convert and save today's date
         OI    TDATE+6,C'0'       fix sign
```

```
* ------ check parameters
         TM    PFLAGS,COPY         was copy specified?
         BNO   PDATOK              no, so see if delete
* ------ check for valid copy days
         NC    COPYDAYS,COPYDAYS   was copy days given?
         BZ    PNCOPYD             no, so check for dates
         LA    R15,L'COPYDAYS      length of copydays field
PCOPYDL  LA    R14,COPYDAYS-1(R15) position within copydays
         CLI   0(R14),C'0'         is character less than 0?
         BL    BADPARM             yes, error
         CLI   0(R14),C'9'         is character greater than 9?
         BH    BADPARM             yes, error
         BCT   R15,PCOPYDL         loop to check all chars
         PACK  DAYSWORK,COPYDAYS   convert to decimal
         PACK  DATEWORK,TDATE      get today's date
         SP    DATEWORK+2(2),DAYSWORK subtract days from today
         BAS   R14,FIXDATE         adjust for the year
         UNPK  EDATE,DATEWORK      save it as end date
         OI    EDATE+6,C'0'        fix sign
         MVC   SDATE,=C'1900001'   set start date to earliest
         B     PDATOK              go see if delete was specified
PNCOPYD  DS    0H   ">nnn" was not specified
* ------ see if start date was given, get default if not
         CLC   SDATE,=XL7'00'      was start date given?
         BE    PSTARTDF            no, so get default
         LA    R9,SDATE            point to start date
         BAS   R14,CHKDATE         see if it is valid
         LTR   R15,R15             is date valid?
         BNZ   BADPARM             error if not
         B     PENDCK              check end date
* ------ get default start date of 1900001
PSTARTDF DS    0H
         MVC   SDATE,=C'1900001'   Get default start date
* ------ check for valid end date
PENDCK   DS    0H
         CLC   EDATE,=XL7'00'      was end date given?
         BE    PENDDEF             no, so get default
         LA    R9,EDATE            point to end date
         BAS   R14,CHKDATE         see if it is valid
         LTR   R15,R15             is date valid?
         BNZ   BADPARM             error if not
         CLC   EDATE,TDATE         is it after today?
         BH    BADPARM             yes, error
* ------ recalculate end date as the day after the given date
         PACK  DATEWORK,EDATE      convert end date to decimal
         AP    DATEWORK+2(2),=PL1'1' add 1 to day
         BAS   R14,FIXDATE         adjust for the year
         UNPK  EDATE,DATEWORK      save it as end date
         OI    EDATE+6,C'0'        fix sign
         B     PENDOK
* ------ set default end date as today
PENDDEF  MVC   EDATE,TDATE         get today's date
PENDOK   DS    0H
         CLC   SDATE,EDATE         see if start date < end date
         BNL   BADPARM             error if not
PDATOK   DS    0H
* ------ convert dates to stck format
         LA    R3,SDATE            start date yyyyddd
         LA    R4,SSTCK            field for stck form
         BAS   R14,CONVSTCK        convert yyyyddd to stck format
         LA    R3,EDATE            end date yyyyddd
         LA    R4,ESTCK            field for stck form
         BAS   R14,CONVSTCK        convert yyyyddd to stck format
* ------ Connect to the log stream
         MVC   LOGRMSGT,=CL10'IXGCONN-1' insert in case of error
         IXGCONN REQUEST=CONNECT,  connect to the log stream          X
               AUTH=WRITE,                                            X
               STREAMNAME=STRNAME,                                    X
               STREAMTOKEN=STRTOKEN,                                  X
               ANSAREA=ANSAREA,                                       X
               ANSLEN=ANSLEN,                                         X
               RETCODE=RETCODE,                                       X
               RSNCODE=RSNCODE
         CLC   RETCODE,=AL4(IXGRETCODEOK) did it work ok?
         BE    CONNOK              yes, continue
* ------ error during connect
*        see if it is "possible loss of data" and ignore it if so
*        IXGRSNCODECONNPOSSIBLELOSSOFDATA = possible loss of data
         L     R14,RSNCODE         get reason code
         N     R14,=AL4(IXGRSNCODEMASK) and it with logger mask
         C     R14,=AL4(IXGRSNCODECONNPOSSIBLELOSSOFDATA) is it       X
                                   possible loss of data?
         BE    CONNOK              yes, disregard
         BAS   R14,LOGRERR
         B     DONEDONE            free storage and quit
LOGRERR  DS    0H   error return code from a system logger request
         BAKR  R14,0
         UNPK  LOGRMRET(4),RETCODE+2(3) get return code
         TR    LOGRMRET,HEXTAB     make it printable
         MVI   LOGRMRET+3,C'.'     replace lost character
         UNPK  LOGRMRSN(5),RSNCODE+2(3) get reason code
         TR    LOGRMRSN,HEXTAB     make it printable
         MVI   LOGRMRSN+4,C' '     replace lost character
         LA    R2,LOGRMSG          point to error message
         BAS   R14,MESSR           display it
         PR    ,
CONNOK   DS    0H
* ------ End initialization
* ------ Begin BROWSE
*   Invoke ISPF browse
*   Start a log stream browse session and position the log stream
*   to first record in the range
*   Copy loop:
*    Read successive records from the stream, starting with the
*    earliest record bearing the start date and supplying browse
*    with the requested records
*    For each record (MDB) that is read:
*     Get the general and CP objects
*     Extract the fixed info
*     For every line (text object) in the message:
*      Write a syslog-format line to the output file
*      If line was too long, also write a continuation line
*   End the log stream browse session
*   Exit
         TM    PFLAGS,COPY         was copy specified?
         BNO   DONE                no, so split it
* ------ Obtain a buffer area for logger record and set up its base
         STORAGE OBTAIN,LENGTH=STRBUFFL,BNDRY=PAGE get storage for buff
         LR    R10,R1              save its address
         USING STRBUFF,R10         addressability
         ST    R10,STORAR
* -----  Invoke BRIF browse
         L     R15,ISPLINK
         CALL  (15),(BRIF,DATANAME,RECFM,LRECL,READR,CMDR,,PANEL),VL
* ------ Obtain a buffer area for logger record and set up its base
         STORAGE RELEASE,LENGTH=STRBUFFL,ADDR=(R10) free the buffer
         B     COPYDONE              Back from browse - all done
* ====== BRIF read record routine
BREADR   DS    0D
         SAVE  (14,12)
         LM    R11,R12,BASESA-BREADR(R15) reload bases
         B     BASESO              go on
BASESA   DC    A(IEAMDBBR,IEAMDBBR+4096)
BASESO   DS    0H
         LA    R15,RXSA            new save.
         ST    R13,4(,R15)
         LR    R13,R15
         LM    R2,R5,0(R1)         parm pointers
         L     R10,STORAR          point at dynamic storage
* ------ set up log record base
         LA    R9,LOGBUF           syslog record
         USING HCL,R9              addressability
* ------ check whether requested record in buffer
CKAGAIN  DS    0H
         L     R6,BUFFFST          point at first rec in buffer
         ICM   R7,15,BUFFFST+4     pick up rec nbr
         BNZ   NTEMPTY             not empty
         BAS   R14,IFILL           empty - fill
NTEMPTY  DS    0H
         ICM   R8,15,BUFFLST+4     pick up last rec nbr
         C     R8,0(,R4)           requested browse record in buffer?
         BL    HFILL               no - try to replenish buffer
         ICM   R7,15,BUFFFST+4     pick up first rec nbr
         C     R7,0(,R4)           requested browse record in buffer?
         BNH   FNDREC              yes - locate record in buffer
         B     HFILL               no - replenish buffer from note
```

```
* ------ request buffer refill
HFILL    DS    0H
         TM    CFLG,CFLGEOF       EOF set?
         BZ    HFILLA             not
         ICM   R0,15,LREC#        last rec nbr
         BZ    HFILLA
         C     R0,0(,R4)          beyond last?
         BL    RC8                yes - return rc 8
HFILLA   DS    0H
         BAS   R14,AFILLX         yes
         B     *+4(R15)
         B     NTEMPTY            rc 0 - requested rec
         B     RC4MORE            rc 4 - temporary EOF - check more
         B     RC8                rc 4 - permanent EOF - last rec
RC4MORE  DS    0H
         L     R1,0(,R4)          pick up requested record nbr
         C     R1,BUFFLST+4       in buffer?
         BNH   NTEMPTY            yes - return record
         B     RC4                rc 4 - temporary EOF - last rec
* ------ locate requested record in buffer
FNDREC   DS    0H
         L     R7,0(,R4)          pick up browse record nbr
FNDRECL  DS    0H
         C     R7,8(,R6)          is this the record
         BE    RECFND             yes - give it to BRIF
         L     R6,0(,R6)          take next..
         B     FNDRECL            .buffer
* ------ return requested record to BRIF
RECFND   DS    0H
         LA    R8,16(,R6)         rec addr..
         ST    R8,0(,R2)          .back to BRIF
         LA    R0,L'CREC          rec lth..
         ST    R0,0(,R3)          .back to BRIF
         B     RC0                back to BRIF
* ------ various returns to BRIF
RC20     DS    0H                 EOF
         LA    R15,20             RC = abort
         B     RET                back to BRIF
RC8      DS    0H                 EOF
         OI    CFLG,CFLGEOF
         LA    R15,8              RC = EOF
         B     RC48
RC4      DS    0H                 temporary EOF
         LA    R15,4              RC = temporary EOF
RC48     DS    0H                 temporary EOF
         L     R0,LREC#           find new last..
         ST    R0,0(,R4)          .to BRIF
         LA    R1,LREC            .data.
         ST    R1,0(,R2)          ..addr to BRIF
         LA    R0,L'CREC          rec lth..
         ST    R0,0(,R3)          .back to BRIF
         B     RET                back to BRIF
RC0      DS    0H
         XR    R15,R15            RC = O.K.
RET      DS    0H
         L     R13,4(,R13)
         RETURN (14,12),RC=(15)
* ------ refill display buffer from note list or last known record
AFILLX   DS    0H
*        R4 - requested record number pointer
         BAKR  R14,0
         ICM   R15,15,LREC#       has there been an EOF yet?
         BZ    AFILLXN            no - check notelist
         L     R14,0(,R4)         requested browse record nbr
         BCTR  R14,0              -1
         C     R14,BUFFLST+4      request for next after last in buf?
         BNE   AFILLXN            no check notelist
*        fill buffer starting ater last in  buffer
         NI    CFLG,X'FF'-CFLGNF  update first buffer controls
         MVC   SSTCK(8),CTST         set start timestamp
         L     R14,BUFFLST        point at last buffer
         MVC   REQRB,0(R14)       set buffer to fill
         MVC   REQRM,12(R14)      set MDB positioning
         L     R15,8(,R14)        compute rec..
         LA    R15,1(,R15)        .nbr
         ST    R15,REQRN          set requested rec nbr
         L     R15,0(R4)          get requested record number
         ST    R15,REQRC          set requested record count
         C     R15,=F'99999999'   max down?
         BE    AFILLXP            yes
         MVC   REQRC,=A(BUFF#/2+1) set requested record count
AFILLXP  DS    0H
         BAS   R14,POSATTSP       position at starting MDB
         LTR   R15,R15            errors?
         BNZ   AFILLXQ            yes - quit w/ rc
         BAS   R14,COPYLOIN       fill display buffers
         B     *+4(R15)           process return codes
         B     AFILLR0
         B     AFILLR4
         B     AFILLR8
*        check notelist
AFILLXN  DS    0H
*        R4 - requested record number pointer
         L     R15,0(,R4)         get requested record number
         C     R15,=F'99999999'   max down?
         BE    AFILLXS            yes - search note from beginning
         XR    R14,R14            calc..
         BCTR  R15,0              ..list..
         D     R14,=A(NOTE#)      ...offset
         L     R14,NOTEA          point at note list
         MH    R15,=AL2(NOTEL)    compute note.
         LA    R15,0(R15,R14)     .entry addr
         OC    0(NOTEL,R15),0(R15) has entry been filled?
         BNZ   AFILLXSF           yes use it
*        locate last used notelist entry
AFILLXS  DS    0H
         L     R15,NOTEA          point at note list
AFILLXSL DS    0H
         OC    NOTEL(NOTEL,R15),NOTEL(R15) is next entry in use?
         BZ    AFILLXSF           no - last active entry found
         LA    R15,NOTEL(,R15)    point at next entry
         B     AFILLXSL           loop
AFILLXSF DS    0H
*        R15 points at note entry
         NI    CFLG,X'FF'-CFLGNF  update first buffer controls
         MVC   SSTCK(8),4(R15)       set start timestamp
*        invalidate buffer
         L     R14,BUFFA          point at beginning of buffer
         LA    R0,BUFF#           nbr of entries
AFILLXSI DS    0H
         XC    8(32,R14),8(R14)   clear
         L     R14,0(,R14)        point at next buffer
         BCT   R0,AFILLXSI        loop...
         MVC   REQRN,0(R15)       set requested rec nbr
         MVI   REQRM,X'FF'        set MDB positioning - none
         ICM   R0,15,12(R15)      compute..
         BZ    *+4+2+4            .MDB..
         BCTR  R0,0               ..positioning..
         ST    R0,REQRM           ...if any
         MVC   REQRB,BUFFA        set buffer to fill
         L     R15,0(R4)          get requested record number
         ST    R15,REQRC          set requested record count
         C     R15,=F'99999999'   max down?
         BE    AFILLXSN           yes
         S     R15,REQRN          subtract fires to get rec nbr
         LA    R15,BUFF#/2        add half of the buffer size
         CH    R15,=AL2(BUFF#)    try to..
         BNL   *+4+4              .request..
         LH    R15,=AL2(BUFF#)    ..a buffer's worth of data
         ST    R15,REQRC          set requested record count
AFILLXSN DS    0H
         MVC   BUFFFST+4(4),REQRN update first entry of..
         MVC   BUFFFST(4),REQRB   .buffer controls
         B     AFILLXP            go get data
AFILLR8  DS    0H
         OI    CFLG,CFLGEOF
         LA    R15,8              rc = 8  - EOF
         B     AFILLXQ
AFILLR4  DS    0H
         LA    R15,4              rc = 4  - temporary EOF
         B     AFILLXQ
AFILLR0  DS    0H
         XR    R15,R15            rc = 0
AFILLXQ  DS    0H
         PR    ,
* ------ initial display buffer fill
```

```
IFILL    DS    0H                 r6 buff element ptr
         BAKR  R14,0
         BAS   R14,POSATTSP       position at starting MDB
         LTR   R15,R15            errors?
         BNZ   IFILLAB
         OI    CFLG,CFLGNF        do not update first buffer controls
         MVI   REQRM,X'FF'        set no inter MDB positioning
         L     R4,BUFFFST         point at first rec in buffer
         ST    R4,REQRB           set first buffer to fill
         LA    R15,BUFF#          nbr elements
         ST    R15,REQRC          set fill count
         LA    R3,1               first rec number
         ST    R3,REQRN           set requested rec nbr
         ST    R3,BUFFFST+4       update first entry of..
         ST    R4,BUFFFST         .buffer controls
         BAS   R14,COPYLOIN       fill display buffers
         OC    RECCOUNT,RECCOUNT  any records?
         BNZ   IFILLBT            yes
IFILLAB  DS    0H
         L     R2,BUFFA           point at first rec in buffer
         L     R0,=F'1'           get record nbr
         ST    R2,BUFFFST         set..
         ST    R0,BUFFFST+4       .buffer..
         ST    R2,BUFFLST         ..controls..
         ST    R0,BUFFLST+4       ...to display..
         ST    R0,LREC#           ....error data
         MVC   LREC(EMPTYLEN),EMPTYMSG+2
         MVC   16(L'LREC,R2),LREC
         ST    R0,8(,R2)
         B     IFILLR20
IFILLBT  DS    0H
         B     *+4(R15)           process return codes
         B     IFILLR0
         B     IFILLR4
         B     IFILLR8
IFILLR20 DS    0H
         OI    CFLG,CFLGEOF
         LA    R15,20             rc = 20 - EOF
         B     IFILLRT
IFILLR8  DS    0H
         OI    CFLG,CFLGEOF
         LA    R15,8              rc = 8  - EOF
         B     IFILLRT
IFILLR4  DS    0H
         LA    R15,4              rc = 4  - temporary EOF
         B     IFILLRT
IFILLR0  DS    0H
         XR    R15,R15            rc = 0
IFILLRT  DS    0H
         PR    ,
* ------ Begin copy loop
*  Read successive records from the stream, starting with the
*  earliest record bearing the start date and ending with the
*  latest record on or before the end date
COPYLOIN DS    0H
         BAKR  R14,0
COPYLOOP DS    0H
* ------ get next record from log stream
         MVC   LOGRMSGT,=CL10'IXGBRWSE-2' insert in case of error
         IXGBRWSE REQUEST=READCURSOR, read next record                 X
               BROWSETOKEN=BRWTOKEN,                                   X
               BUFFER=STRBUFF,                                         X
               BUFFLEN=STRBLEN,                                        X
               DIRECTION=OLDTOYOUNG,                                   X
               RETBLOCKID=CURRBLK,                                     X
               TIMESTAMP=CURRSTCK,                                     X
               STREAMTOKEN=STRTOKEN,                                   X
               ANSAREA=ANSAREA,                                        X
               ANSLEN=ANSLEN,                                          X
               RETCODE=RETCODE,                                        X
               RSNCODE=RSNCODE
         CLC   RETCODE,=AL4(IXGRETCODEOK) did it work ok?
         BE    READCOK             yes, continue
* ------ error in BROWSE READCURSOR
*        see if it is just a gap in the stream, and continue if so
*        see if we reached EOF and end if so
*        IXGRSNCODEWARNINGGAP = request successful but data missing
*        IXGRSNCODEEOFDELETE  = reached EOF when records deleted
*        IXGRSNCODEENDREACHED = reached EOF
*        IXGRSNCODEEOFGAP     = reached prematurely EOF            @JPV
         L     R14,RSNCODE         get reason code
         N     R14,=AL4(IXGRSNCODEMASK) and it with logger mask
         C     R14,=AL4(IXGRSNCODEWARNINGGAP) is it a gap?
         BE    READCOK             yes, ok
         C     R14,=AL4(IXGRSNCODEEOFDELETE) is it EOF?
         BE    READEOF             yes, done
         C     R14,=AL4(IXGRSNCODEEMPTYSTREAM) is it EOF?
         BE    READEOF             yes, done
         C     R14,=AL4(IXGRSNCODEENDREACHED) is it EOF?
         BE    READEOF             yes, done
         C     R14,=AL4(IXGRSNCODEEOFGAP) is it EOF?                @JPV
         BE    READEOF             yes, done
         C     R14,=AL4(IXGRSNCODEENDREACHED) is it EOF?
         BE    READEOFP            yes, done
         BAS   R14,LOGRERR         display the error
READEOFP DS    0H
         LA    R15,8               some error - permanent eof
         B     READNOK             save last known good record status
READEOF  DS    0H
         LA    R15,4               temporary eof
READNOK  DS    0H
         MVC   LTST,CTST           set last rec timestamp
         MVC   LRECM,CRECM         set last rec in MDB
         MVC   LREC,CREC           set last known record
         MVC   LREC#,CREC#         set nbr, too
         PR    ,
READCOK  DS    0H
         XC    MDBREC,MDBREC       clear rec nbr in MDB
* ------ increment record count
         L     R15,RECCOUNT        get record count
         A     R15,=F'1'           add 1
         ST    R15,RECCOUNT        store new count
* ------ For each record (MDB) that is read:
*     Get the general and CP objects
*     Extract the fixed info
*     For every line (text object) in the message
*        Write a syslog-format line to the output file
*        If line was too long, also write a continuation line
STARTMDB LR    R8,R10              point to mdb in buffer
         USING MDB,R8
         MVI   FLAGS1,0            clear processing flags
         MVC   HCL(128),=CL128' '  clear out log record
         LR    R6,R8               calc end of mdb in R6
         AH    R6,MDBLEN           start+mdblen in header
         LA    R7,MDBHLEN(0,R8)    address of first object
         CR    R7,R6               see if this is the end
         BNL   COPYLOOP            get another MDB if so (no           X
                                   objects)
         DROP  R8
         USING MDB,R7
* ------  scan MDB objects looking for general and CP objects, and
* save syslog information from them
OBJLP    DS    0H                  loop through the objects
         LH    R3,MDBTYPE          get type
         C     R3,=A(MDBGOBJ)      check for general object
         BNE   NOTG                not general object
         TM    FLAGS1,FLAGGO       see if first general object
         BO    NXTOBJ              no, skip it
         OI    FLAGS1,FLAGGO       show general object was found
         USING MDBG,R7             addressability to general object
* ------ Move general object fields into log record or save them
         MVC   HCLSYSID,MDBGOSNM   system name
         MVC   HCLDATE,MDBGDSTP+2  date (note that MDB form is yyyyddd)
         MVC   HCLTIME,MDBGTIMH    time
         MVI   HCLTIME+2,C':'      separate with
         MVI   HCLTIME+5,C':'       colons
         MVC   JOBNAME,MDBGJBNM    save job name
         SR    R15,R15             clear a reg
         ICM   R15,7,MDBGSEQ       message sequence number
         CVD   R15,DBLWD           convert to decimal
         UNPK  MLID,DBLWD          save low 3 digits as ebcdic         X
                                   multiline id
         OI    MLID+2,C'0'         make it printable
         B     NXTOBJ              bump to next object
NOTG     DS    0H
         C     R3,=A(MDBCOBJ)      check for control prog object
```

```
         BNE   NXTOBJ              not control prog object, get next
         TM    FLAGS1,FLAGCO       see if first control prog object
         BO    NXTOBJ              no, skip it
         USING MDBSCP,R7           addressability to ctl prog object
         CLC   MDBCPNAM,=C'MVS '   make sure it is an MVS object
         BNE   NXTOBJ              if not, just skip cp object
         CLC   MDBCVER,=AL4(MDBCVER5) see if it's the right version
         BL    COPYLOOP            skip MDB if not
         OI    FLAGS1,FLAGCO       set processed control prog object
* ------ Move control pgm object fields into log record or save them
* set up record type
         MVI   HCLRECTP,HCLMLWTO   assume a multiline msg
         CLC   MDBCLCNT,=F'1'      see if more than one line
         BH    CPROK               ok if so
         MVI   HCLRECTP,HCLWTO     make it a single line message
         TM    MDBMLVL1,MDBMLR     is it a wtor?
         BNO   CPROK               no, ok
         MVI   HCLRECTP,HCLWTOR    make it a wtor
* set up request type
CPROK    TM    MDBCMSC2,MDBCOCMD   is it an operator cmd echo?
         BNO   CPNOP               no, try next
         MVI   HCLREQTP,HCLCMD     mark it as operator command
         CLC   MDBCCNID,=F'0'      is it an internal cmd (cons id=0)?
         BNE   CPRC                no, so HCLCMD is ok
         MVI   HCLREQTP,HCLINTNL   mark it as internal command
         B     CPRC
CPNOP    TM    MDBCATT1,MDBCMCSC   is it a command response?
         BO    CPRSP               yes, mark it so
         TM    MDBDESC1,MDBDESCE   is it desc=5 (also cmd resp)
         BNO   CPRC                no, not cmd response
CPRSP    MVI   HCLREQTP,HCLRESP    mark it as command response
* make routing codes printable
CPRC     UNPK  TWORK(9),MDBCERC(5) unpack first 7 routing codes
         TR    TWORK(7),HEXTAB     make them printable
         MVC   HCLROUTC,TWORK      move into record
* make request flags printable
         UNPK  TWORK(9),MDBCXMOD(5) unpack request flags
         TR    TWORK(8),HEXTAB     make them printable
         MVC   HCLREQFL,TWORK      move into record
         MVC   REQFL,TWORK         save for second line
* save console id, console name, MCS flags, and descriptors
         MVC   CONSID,MDBCCNID     save console id
         MVC   CONSNAME,MDBCCNNM   save console name
         MVC   MCSFLAGS,MDBCMCSF   save MCS flags
         MVC   DESCS,MDBCDESC      save descriptor codes
* remember whether this is a WTL
         MVI   WTLFLAG,C'N'        assume not wtl
         TM    MDBCMSC2,MDBCWTL    is it a wtl?
         BNO   NXTOBJ              no, ok
         MVI   WTLFLAG,C'Y'        show it's a wtl
NXTOBJ   DS    0H                  find next object
         TM    FLAGS1,FLAGGO+FLAGCO see if we found general and SCP
         BO    FNDTXT              got them, loop through text objs
         USING MDB,R7
         AH    R7,MDBLEN           bump to next object
         CR    R7,R6               see if this is the end
         BL    OBJLP               no, process this object
         B     COPYLOOP            missing necessary objects,          X
                                   skip it
         DROP  R7
* ------ find text objects, convert them to syslog records
FNDTXT   DS    0H
         LA    R7,MDBHLEN(0,R8)    address of first object
         CR    R7,R6               see if this is the end
         BNL   COPYLOOP            get another MDB if so               X
                                   objects)
         USING MDB,R7
* ------ fill in jobname/consname field
         CLI   HCLREQTP,HCLCMD     is it a command?
         BNE   NOTCMD              no, try next
         CLC   CONSID,=F'128'      is it an instream command?
         BE    GETCONS             yes, use console name (INSTREAM)
         CLC   JOBNAME,=CL8' '     is jobname blank?
         BE    GETCONS             yes, use console name
         B     CHKMCS              go see if it's an MCS console
NOTCMD   CLI   HCLREQTP,HCLINTNL   is it an internal command?
         BNE   NOTINTL             no, try next
         CLC   JOBNAME,=CL8' '     is jobname blank?
         BNE   GETJOB              no, so use job name
         MVI   HCLREQTP,HCLCMD     change request type to "command"
         B     GETCONS             use console name (INTERNAL)
NOTINTL  CLI   HCLREQTP,HCLRESP    is it a command response?
         BNE   GETJOB              no, so use job name
         TM    MCSFLAG1,MDBMCSB+MDBMCSH was it sent by console id      X
                                        in reg 0?
         BZ    GETJOB              no, use job name
         CLC   JOBNAME,=CL8' '     is jobname blank?
         BNE   GETJOB              no, so use job name
         CLC   CONSID,=F'0'        is it internal?
         BE    GETCONS             yes, use console name (INTERNAL)
         CLC   CONSID,=F'128'      is it instream?
         BE    GETCONS             yes, use console name (INSTREAM)
CHKMCS   CLI   CONSID,X'00'       is console class zero (MCS)?
         BE    GETJOB              yes, use job name
         TM    CONSID,X'E0'        is console class a JES3 console?
         BNZ   GETJOB              yes, so use job name
* ------ move console name into hcl
GETCONS  MVC   HCLCONID,CONSNAME   move in console name from MDB
         CLC   CONSID,=F'0'        is it "internal"?
         BNE   CONSNOTI            no, ok
         MVC   HCLCONID,=CL8'INTERNAL' move in "internal"
         B     MDBJOBOK
CONSNOTI CLC   CONSID,=F'128'      is it "instream"?
         BNE   MDBJOBOK            no, ok
         MVC   HCLCONID,=CL8'INSTREAM' move in "instream"
         B     MDBJOBOK            done with job/console field
* ------ move in job name
GETJOB   MVC   HCLJOBID,JOBNAME    move in job name
MDBJOBOK DS    0H
* ------ erase the timestamp if NOTIME was requested
         TM    MCSFLAG2,MDBMCSI    was "NOTIME" requested?
         BNO   TIMEOK              no, so leave time alone
         MVC   HCLTIME,=CL11' '    blank out the time stamp
TIMEOK   DS    0H
* ------ remember this is the first line in the message
         MVI   FIRSTLNE,C'Y'       set first-line indicator
* ------ scan MDB looking for text objects
TOBJLP   CLC   MDBTYPE,=AL2(MDBTOBJ) check for text object
         BNE   NXTTOBJ             not text object, try next
* ------ text object - convert it to syslog record
         USING MDBT,R7             addressability to text object
* calculate length of text in R2
         LH    R2,MDBTLEN          get text object length
         S     R2,=A(MDBTMSGT-MDBTLEN) subtract non-text size
         BNP   NXTTOBJ             skip it if length is zero or less
         LA    R3,MDBTMSGT         get address of text
         CLI   WTLFLAG,C'Y'        is it a wtl?
         BNE   NOTWTL              no, skip to the non-wtl case
* message came from a wtl
* PUT only the text (no control info) from the first line
         CH    R2,=H'128'          does text length exceed max?
         BNH   WTLLOK              no, ok
         LA    R2,128              set it to max
WTLLOK   S     R2,=F'1'            subtract 1 for mvc
         BM    COPYLOOP            skip it if negative (length < 1)
         EX    R2,WTLMV            move in the text
         LA    R2,5(0,R2)          add for RDW and get back the 1
         STH   R2,LOGBUFL          set record length
         BAS   R14,PUTRECB         move record into buffer
         LTR   R15,R15             requested records received?
         BZ    COPYLOOP            no - get next MDB
         XR    R15,R15             set good rc
         PR    ,
WTLMV    MVC   LOGBUF(*-*),0(R3)   executed above
*
NOTWTL   DS    0H
* not a wtl
* see if this is line 2 or greater of a multiline, and if so
*     show the multiline id and
*     fill in the record type from the line type
         CLI   FIRSTLNE,C'Y'       see if this is the first line
         BE    MDBMLOK             bypass if so
         MVC   HCLMLID,MLID        move in the multiline id number
         MVC   HCLREQFL,REQFL      move in request flags
         TM    MDBTLNT1,MDBTLABT   see if label line
         BNO   TXTNL               no, try data
```

```
         MVI   HCLRECTP,HCLLABEL   show it is a label
TXTNL    TM    MDBTLNT1,MDBTDATT   see if data line
         BNO   TXTND               no, try end
         MVI   HCLRECTP,HCLDATA    show it is data
TXTND    TM    MDBTLNT1,MDBTENDT   see if end line
         BNO   MDBMLOK             no, ok
         MVI   HCLRECTP,HCLDTEND   show it is the end line
MDBMLOK  DS    0H
* place text behind control info
         MVC   LOFF,=F'0'          init offset of text in log record
*                                   (zero for first line, then one)
* loop through text, issue PUT for each piece of text up to length 128
TXTLP    C     R2,=A(128-HCLHEADL) see if text is too long for buffer
         BNH   TXTDN               do last piece if not
* truncate at a blank or comma, get length in R4
         LA    R4,127-HCLHEADL(0,R3) starting position
         LA    R1,118-HCLHEADL(0,R3) ending position
         MVC   BLANKCT,=F'1'       assume there is a blank
TXTSC    CLI   0(R4),C' '          look for a blank
         BE    TXTL                stop if found
         CLI   0(R4),C'.'          look for a comma
         BNE   TXTBK               not found, try previous position
         LA    R4,1(,R4)           keep comma on this line
         MVC   BLANKCT,=F'0'       show no blank found
         B     TXTL                split the line
TXTBK    BCTR  R4,0                back up
         CR    R4,R1               see if at end position
         BNL   TXTSC               loop back if not
         LA    R4,128-HCLHEADL(0,R3) too big - trunc at 128
         MVC   BLANKCT,=F'0'       show no blank found
TXTL     SR    R4,R3               calculate length
* issue PUT for the partial text
         BAS   R14,PUTREC          PUT the syslog record
         LTR   R15,R15             all done this time?
         BNZ   TXTDNOU             yes
         MVI   HCLRECTP,HCLSPLIT   show this is a continuation
         MVC   LOFF,=F'1'          adjust text offset for cont. lines
         A     R4,BLANKCT          skip the blank if there was one
         SR    R2,R4               reduce the count
         AR    R3,R4               bump down the record
         B     TXTLP               loop to do all pieces
TXTDNOU  DS    0H
         XR    R15,R15
         PR    ,
* issue PUT for last (or only) piece
TXTDN    DS    0H
* if this is the first line of a multiline and is not an operator
*     request (descriptor code 9), append the multiline id to the
*     text.  If there is not enough room in the line for the id,
*     print it on the next (split) line by itself.
         CLI   HCLRECTP,HCLMLWTO   is it first line of multiline?
         BNE   NOTFIRST            no, ok
         TM    DESC2,MDBDESCI      is it descriptor code 9?
         BO    NOTFIRST            yes, ok
         C     R2,=A(128-HCLHEADL-4) see if there is room for mlid
         BH    MLSPLIT             go split line if not
*        build line with mlid appended
         LR    R14,R2              get text length
         BCTR  R14,0               subtract 1 for MVC
         EX    R14,MLMVC           move in the text
         LA    R14,MLTEMPLN(R2)    end of text + 1
         MVC   0(4,R14),HCID       append mlid
         A     R2,=F'4'            add 4 to length for mlid
         LA    R3,MLTEMPLN         point to new line
         B     NOTFIRST            go put the line
MLMVC    MVC   MLTEMPLN(*-*),0(R3) executed above
*        put the line and build a split line containing the MLID
MLSPLIT  DS    0H
         LR    R4,R2               get text length
         BAS   R14,PUTREC          put the line
         LTR   R15,R15             all done this time?
         BNZ   TXTDNOU             yes
         MVI   HCLRECTP,HCLSPLIT   show this is a continuation
         L     R2,=F'4'            length of mlid
         LA    R3,HCID             address of mlid
NOTFIRST DS    0H
         LR    R4,R2               get length of text
         BAS   R14,PUTREC          PUT the syslog record
         LTR   R15,R15             all done this time?
         BNZ   TXTDNOU             yes
* bump to next object
NXTTOBJ  DS    0H
         USING MDB,R7
         AH    R7,MDBLEN           bump to next object
         CR    R7,R6               see if this is the end
         BL    TOBJLP              no, look at this object
         DROP  R7
         B     COPYLOOP            done with this mdb; get next
* ------ End copy loop
COPYDONE DS    0H
* ------ End the log stream browse session
         MVC   LOGRMSGT,=CL10'IXGBRWSE-3' insert in case of error
         IXGBRWSE REQUEST=END,                                         X
               BROWSETOKEN=BRWTOKEN,                                   X
               STREAMTOKEN=STRTOKEN,                                   X
               ANSAREA=ANSAREA,                                        X
               ANSLEN=ANSLEN,                                          X
               RETCODE=RETCODE,                                        X
               RSNCODE=RSNCODE
         CLC   RETCODE,=AL4(IXGRETCODEOK) did it work ok?
         BNE   LOGRERR             no, display error
DONE     DS    0H
* ------- Begin Cleanup
*  Disconnect from the log stream
*  Free the buffer area
*  Exit
         MVC   LOGRMSGT,=CL10'IXGCONN-2' insert in case of error
         IXGCONN REQUEST=DISCONNECT, disconnect from the log stream    X
               STREAMTOKEN=STRTOKEN,                                   X
               ANSAREA=ANSAREA,                                        X
               ANSLEN=ANSLEN,                                          X
               RETCODE=RETCODE,                                        X
               RSNCODE=RSNCODE
         CLC   RETCODE,=AL4(IXGRETCODEOK) did it work ok?
         BNE   LOGRERR             no, display error
DONEDONE DS    0H
         PR    ,                   exit
* ------ Begin subroutines
* ------ issue BROWSE START to get browse session going
*        and position to first record in range
POSATTSP DS    0H
         BAKR  R14,0                   save
         MVC   LOGRMSGT,=CL10'IXGBRWSE-1' insert in case of error
         IXGBRWSE REQUEST=START,                                       X
               SEARCH=SSTCK,           start date                      X
               GMT=NO,                 local time                      X
               BROWSETOKEN=BRWTOKEN,                                   X
               STREAMTOKEN=STRTOKEN,                                   X
               ANSAREA=ANSAREA,                                        X
               ANSLEN=ANSLEN,                                          X
               RETCODE=RETCODE,                                        X
               RSNCODE=RSNCODE
         L     R15,RETCODE
         CLC   RETCODE,=AL4(IXGRETCODEOK) did it work ok?
         BE    POSATTOK            yes, so we have starting position
* ------ error in BROWSE START
*        see if it is just a gap in the stream, and continue if so
*        see if there are just no records in range
*        IXGRSNCODEWARNINGGAP = request successful but data missing
*        IXGRSNCODENOBLOCK = block does not exist
         XR    R15,R15
         L     R14,RSNCODE         get reason code
         N     R14,=AL4(IXGRSNCODEMASK) and it with logger mask
         C     R14,=AL4(IXGRSNCODEWARNINGGAP) is it a gap?
         BE    POSATTOK            yes, continue
         C     R14,=AL4(IXGRSNCODENOBLOCK) is it block not found?
         BNE   POSATTAB            no, display the error
         ICM   R2,15,RECCOUNT
         BNZ   POSATTOK
         LA    R2,EMPTYMSG         point to info msg
         BAS   R14,MESSR           display it
         B     POSATTER
POSATTAB DS    0H
         BAS   R14,LOGRERR        no, display the error
POSATTER DS    0H
         L     R15,RETCODE         set return code
```

```
POSATTOK DS    0H
         PR    ,                   return
* ------ CHKDATE - validate start/end date
*   Input:
*     R9 -> date presumably in the form yyyyddd
*     R14 = return address
*   Output:
*     if date is valid, set r15 = 0, otherwise set r15 = nonzero
CHKDATE  LA    R15,1(0,0)          assume date is invalid
         TRT   0(7,R9),NUMTAB      scan for numbers
         BNZR  R14                 not all numbers, exit
         PACK  DATEWORK,0(7,R9)    pack the date
         CP    DATEWORK+2(2),=P'366'    is it gt 366?
         BHR   R14                 yes, error
         BE    CHKDLEAP            366, must be a leap year
         CP    DATEWORK+2(2),=P'1' is it lt 1?
         BLR   R14                 yes, error
         SR    R15,R15             show date is valid
         BR    R14                 exit
*        day is 366 -- make sure it's a leap year
CHKDLEAP SRP   DATEWORK,64-3,0     shift out ddd
         DP    DATEWORK,=PL1'4'    divide year by 4
         CP    DATEWORK+3(1),=P'0' see if remainder is zero
         BNER  R14                 exit if not, error
         SR    R15,R15             show date is valid
         BR    R14                 exit
* ------ FIXDATE - adjust year after adding / subtracting days
*   Input:
*     WORKDATE = date in the form yyyyddd packed; day may be zero
*                or less, or over 365 (366 for leap years)
*     R14 = return address
*   Output:
*     None; date in WORKDATE is adjusted to correct year and day
FIXDATE  DS    0H
         MVC   DATEWRK1,DATEWORK   copy date
         OI    DATEWRK1+3,X'0F'    force sign positive
         SRP   DATEWRK1,64-3,0     shift out ddd
         NC    DATEWORK,=X'0000FFFF' zero out year in datework
FIXDBACK DS    0H   back up year if day is zero or less
         CP    DATEWORK,=PL1'0'    is the day zero?
         BH    FIXDFWD             >0, ok
*        day is 0 or less; adjust day and back up to previous year
         AP    DATEWORK,=P'365'    adjust day
         SP    DATEWRK1,=PL1'1'    subtract 1 from year
*        if leap year, add 1 to day
         MVC   DATEWRK2,DATEWRK1   copy the year
         DP    DATEWRK2,=PL1'4'    divide by 4
         CP    DATEWRK2+3(1),=PL1'0' is remainder zero (leap yr)?
         BNE   FIXDBACK            no, not leap year,loop
         AP    DATEWORK,=PL1'1'    add 1 to day
         B     FIXDBACK            loop
FIXDFWD  DS    0H   add to year if day is over 365 (366 if leap yr)
         MVC   DATEWRK2,DATEWRK1   copy the year
         DP    DATEWRK2,=PL1'4'    divide by 4
         CP    DATEWRK2+3(1),=PL1'0' is remainder zero (leap yr)?
         BE    FIXDFWDL            yes
*        not leap year
         CP    DATEWORK,=PL2'365'  is day over 365?
         BNH   FIXDDONE            no, done
         SP    DATEWORK,=PL2'365'  subtract 365 from day
FIXDFWDA AP    DATEWRK1,=PL1'1'    add 1 to year
         B     FIXDFWD             loop back
*        leap year
FIXDFWDL CP    DATEWORK,=PL2'366'  is day over 366?
         BNH   FIXDDONE            no, done
         SP    DATEWORK,=PL2'366'  subtract 366 from day
         B     FIXDFWDA            add to year
FIXDDONE SRP   DATEWRK1,3,0        adjust year to form yyyyddd
         AP    DATEWORK,DATEWRK1   add year back in
         BR    R14                 exit
* ------ CONVSTCK -Convert date from yyyyddd to stck format
*   Input:
*     R3 -> date to convert, yyyyddd packed
*     R4 -> field to hold STCK format date
*     R14 = return address
*   Output:
*     None; converted date is stored at address in R4.
*     Branches to BADPARM if conversion fails.
*     Returns with no change if input date is binary zero.
CONVSTCK BAKR  R14,0               save caller's environment
         CLC   0(7,R3),=XL7'0'     is input date zero?
         BE    CONVDONE            yes, just return
         PACK  CONVDATE,0(7,R3)    move date to parm area
         SP    CONVDATE,=P'1900000' strip off century
         CONVTOD CONVVAL=CONVWORK,  convert to stck value             X
               TODVAL=(R4),                                           X
               TIMETYPE=BIN,                                          X
               DATETYPE=YYDDD
         LTR   R15,R15             did it work?
         BNZ   BADPARM             no, error
CONVDONE PR    ,                   return
* ------ MESSR -- Display a message
*   Input:
*     r2 -> text of message
*     R14 = return address
MESSR    DS    0H
         BAKR  R14,0               save caller's environment
         MVI   LMSG,C' '
         MVC   LMSG+1(L'LMSG),LMSG
         LH    R15,0(,R2)
         BCTR  R15,0
         MVC   LMSG(0),2(R2)
         EX    R15,*-6
         L     R15,ISPLINK
         CALL  (15),(SETM,MSGI),VL
         PR                        return to caller
* ------ PUTREC -- PUT a record to the output file and set up for next
*   Input:
*     r3 -> text
*     r4  = text length
*     R14 = return address
PUTREC   DS    0H
         BAKR  R14,0               save caller's environment
         LR    R1,R4               length of text
         S     R1,=F'1'            subtract 1 for mvc
         BM    PUTRECX             return if negative (length < 1)
         LA    R2,HCLTEXT          borrow r2 (PR will restore it)
         A     R2,LOFF               for offset to text in log record
         EX    R1,PUTMV            move in the text
         A     R1,=A(HCLHEADL+5)   calculate length of log record
         A     R1,LOFF               including the offset
         STH   R1,LOGBUFL          move it into prefix
         BAS   R14,PUTRECB         move record into buffer
* set up for next record
         ICM   R14,1,HCLREQTP     save request type
         MVC   HCL(128),=CL128' ' clear out log record
         STCM  R14,1,HCLREQTP     restore request type
         MVI   FIRSTLNE,C'N'       show this is no longer first line
PUTRECX  PR    ,                   return
PUTMV    MVC   0(*-*,R2),0(R3)     executed instruction
* -----
PUTRECB  DS    0H                  move record into buffer
         BAKR  R14,0
         L     R15,REQRC           get fill count
         L     R3,REQRN            get requested rec nbr
         L     R4,REQRB            get buffer address
         LA    R1,1                increment..
         A     R1,MDBREC           .MDB..
         ST    R1,MDBREC           ..record count.
         CLI   REQRM,X'FF'         inter MDB record positioning?
         BE    PUTBLL              no
         C     R1,REQRM            start with tis MDB record?
         BH    PUTBLL              yes
         BNE   *+4+4               stop positioning?
         MVI   REQRM,X'FF'         yes - take next MDB
         XR    R15,R15             keep going rc
         PR    ,                   done
PUTBLL   DS    0H
         MVI   REQRM,X'FF'         reset inter MDB record positioning
         ST    R3,8(,R4)           save rec nbr
         MVC   12(4,R4),MDBREC     update MDB rec nbr
         MVC   16(L'LOGBUF,R4),LOGBUF fill buffer
* ------ update notelist
         XR    R0,R0               take a note..
         LR    R1,R3               .for every 'NOTE#'.
         D     R0,=A(NOTE#)        ..records
```

```
         CH    R0,=H'1'           note time?
         BNE   PUTBNONO           no
         L     R14,NOTEA          point at note list
         MH    R1,=AL2(NOTEL)     compute note.
         LA    R14,0(R1,R14)      .entry index
         ICM   R6,15,0(R14)       pick up note rec number
         BNZ   PUTBNONO           entry already updated
         ST    R3,0(,R14)         note rec nbr
         MVC   4(8,R14),CURRSTCK+8  update local time stamp
         MVC   12(4,R14),MDBREC   update MDB rec count
PUTBNONO DS    0H
* ------ update first buffer entry controls
         TM    CFLG,CFLGNF        do not update first buffer controls?
         BO    PUTBLNF            true
         L     R1,0(,R4)          get next buffer addr
         ICM   R0,15,8(R1)        pick up record nbr
         BZ    PUTBLNF            none
         ST    R0,BUFFFST+4       update first entry in..
         ST    R1,BUFFFST         .buffer controls
* ------ update last buffer entry controls
PUTBLNF  DS    0H
         ST    R3,BUFFLST+4       update last entry in..
         ST    R4,BUFFLST         .buffer controls
         ST    R3,CREC#           save current record nbr
         MVC   CTST,CURRSTCK+8    save current local MDB time stamp
         MVC   CRECM,MDBREC       save current MDB record
         MVC   CREC,LOGBUF        save current current record data
         LA    R3,1(,R3)          increment record count
         ST    R3,REQRN           set requested rec nbr
         L     R4,0(R4)           point at next buffer element
         ST    R4,REQRB           save buffer address
         BCTR  R15,0              loop
         ST    R15,REQRC          save fill count
         LTR   R15,R15            all requested records returned?
         BNZ   PUTBLPR            no - keep looping
         LA    R15,4              rc = 4 - done
         PR    ,
PUTBLPR  DS    0H
         XR    R15,R15            rc = 0
         PR    ,
* ------ End subroutines
* ------ static variables
*      translate table for testing for ebcdiic numbers
NUMTAB   DC    240X'FF',10X'00',6X'FF'
*      translate tables for scanning parm field
TRTTAB1  DC    256X'00'
         ORG   TRTTAB1+C','         stop on comma
         DC    C','
         ORG   TRTTAB1+C'('         stop on right paren
         DC    C'('
         ORG   ,
TRTTAB2  DC    256X'00'
         ORG   TRTTAB2+C')'         stop on left paren
         DC    C')'
         ORG   ,
ZLPAREN  DC    0F'0',3X'00',C'('   3 zeros and a left paren
*      translate table for hex conversion
*      must be at least 240 bytes past base
HEXTAB   EQU   *-240
         DC    C'0123456789ABCDEF' must follow hextab
STRNAME  DC    CL26'SYSPLEX.OPERLOG' stream name
ANSLEN   DC    A(L'ANSAREA)       length of logger's answer area
STRBUFFL EQU   64*1024            length of largest log record
STRBLEN  DC    A(STRBUFFL)
         LTORG
         DROP  R11,R12
* ------ BRIF command routine
BRCMDR   DS    0F
         BAKR  R14,0              save regs
         LR    R12,R15            base
         USING BRCMDR,R12
         LA    R13,CXSA
         MVC   4(4,R13),=C'F1SA'  set acro in save area
         LM    R2,R3,0(R1)        pick up parameters
         CLC   =F'10',0(R2)       recursive browse?
         BE    CRET4              yes - let ISPF worry about it
*        invoke VCOPY to get access to ZCMD
         MVI   LMSG,C' '
         MVC   LMSG+1(L'LMSG-1),LMSG
         L     R15,ISPLINK
         CALL  (15),(VCOP,NMZC,LAZC,VAZC,MOZC),VL
         MVC   LMSG(44),=CL44'Command not executed - ZCMD VCOPY failed'
         LTR   R15,R15            VCOPY ok?
         BNZ   CSETM              no - message
         L     R8,VAZC            point at value
         CLI   0(R8),C'/'         an operator command?
         BNE   CRET4              no - ISPF stuff
         MVC   LMSG(44),=CL44'Invalid operator command - not executed'
         L     R7,=A(CMD)         get address of command area
         L     R6,LAZC            get length of command to execute
         SH    R6,=H'2'           adjust for EX
         BNP   CSETM              no command text
         CH    R6,=AL2(L'CMD-1)   too much command text?
         BH    CSETM              yes - message
         MVI   0(R7),C' '         clear..
         MVC   1(L'CMD-1,R7),0(R7) .command area
         MVC   0(*-*,R7),1(R8)    set command to execute
         EX    R6,*-6
         L     R15,=A(IEAMDBCM)
         BALR  R14,R15            invoke command processor
         LTR   R15,R15            command executed?
         BZ    CRET0              yes - ok
         L     R6,=A(AREA)        point at error message
         MVC   LMSG,0(R6)         set message
         B     CSETM              message
CSETM    DS    0H
         L     R15,ISPLINK
         CALL  (15),(SETM,MSGI),VL issue message
CRET4    DS    0H
         LA    R15,4              ISPF should process the function
         B     CRETQ
CRET0    DS    0H
         XR    R15,R15            normal completion
CRETQ    DS    0H
         PR    ,
         DROP  R12                cmdr
* ------ dynamic variables
SV       DS    18F                save area
DATEWORK DS    F                  work area for checking dates
DATEWRK1 DS    F                  work area for checking dates
DATEWRK2 DS    F                  work area for checking dates
DAYSWORK DS    F                  work area for checking dates
SSTCK    DS    2F                 start date in stck format
ESTCK    DS    2F                 end date in stck format
CONVWORK DC    4F'0'              parm area for convtod macro
CONVDATE EQU   CONVWORK+8,4       date in parm area
CURRSTCK DS    2F,2F              timestamps (GMT,local) of curr rec
RETCODE  DS    F                  return code from logger
RSNCODE  DS    F                  reason code from logger
BLANKCT  DS    F                  count of blanks in message segment
LOFF     DS    F                  offset of text in log record
RECCOUNT DS    F                  number of logger records read
DBLWD    DS    D                  work area for cvd
SDATE    DS    XL7                start date as ebcdiic yyyyddd
EDATE    DS    XL7                end date as ebcdiic yyyyddd
TDATE    DS    XL7                today's date as ebcdiic yyyyddd
COPYDAYS DS    CL3                copy days ebcdiic nnn
STRTOKEN DS    CL16               token for accessing stream
BRWTOKEN DS    CL4                token for browse session
ANSAREA  DS    CL(ANSAA_LEN)      answer area for log requests
CURRBLK  DS    XL8                block id of current block
DELBLK   DS    XL8                block id of blk after one to delete
TWORK    DS    CL16               work area for hex translate
JOBNAME  DS    CL8                jobname
CONSNAME DS    CL8                console name
CONSID   DS    XL4                console id
MCSFLAGS DS    0CL2               MCS flags from MDB
MCSFLAG1 DS    X                  MCS flag 1
MCSFLAG2 DS    X                  MCS flag 2
HCID     DC    CL4' '             hardcopy id
         ORG   HCID+1             multiline id goes in bytes 2-4
MLID     DC    C'NNN'             multiline id from message
         ORG   ,
MLTEMPLN DC    CL80' '            work area for split line
*
DESCS    DS    0XL2               copy of descriptor codes
```

```
DESC1    DS    XL1                 descriptor codes byte 1
DESC2    DS    XL1                 descriptor codes byte 2
*
REQFL    DC    CL8' '              copy of request flags
WTLFLAG  DS    C                   'Y' indicates a WTL
FIRSTLNE DS    C                   'Y' indicates the first msg line
*
FLAGS1   DS    XL1                 mdb flags
FLAGGO   EQU   X'01'               processed general object
FLAGCO   EQU   X'02'               processed control prog object
*
PFLAGS   DS    XL1                 parameter flags
COPY     EQU   X'01'               "COPY" was specified
DELETE   EQU   X'02'               "DELETE" was specified
*
OFF31    DC    0F'0',XL4'7FFFFFFF' anded to a reg to turn off bit 0
ON31     DC    0F'0',XL4'80000000' ored to a reg to turn on bit 0
* ------ buffer for log record
LOGBUFP  DS    0F                  prefix to log record
LOGBUFL  DS    H                   length of logbuf data + 4
         DC    H'0'
LOGBUF   DS    CL128               log record (mapped by ihahclog)
* ------ messages
BADPMSG  DC    AL2(BADPLEN),C'MLG001I BAD INPUT PARAMETERS'
BADPLEN  EQU   *-BADPMSG-2
LOGRMSG  DC    AL2(LOGRMLEN)
LOGRMSGD DC    C'MLG002I SYSTEM LOGGER -'
LOGRMSGT DC    CL10' ',C', RC ='
LOGRMRET DC    CL3' ',C', RSN ='
LOGRMRSN DC    CL4' ',C' '
LOGRMLEN EQU   *-LOGRMSG-2
EMPTYMSG DC    AL2(EMPTYLEN),C'MLG003I NO RECORDS IN RANGE'
EMPTYLEN EQU   *-EMPTYMSG-2
*
BRIF     DC    CL8'BRIF'
DATANAME DC    CL54'LIVE.SYSLOG.THROUGH.OPERLOG'
RECFM    DC    CL4'F'
LRECL    DC    F'128'
READR    DC    A(BREADR)
PANEL    DC    CL8'ISRBROBA'
BRSA     DC    18F'0'              save area
RXSA     DC    18F'0'              save area
ISPLINK  DC    F'0'
DW       DS    D
SETM     DC    CL8'SETMSG '
MSGI     DC    CL8'ISRZ001'
VDEF     DC    CL8'VDEFINE'
OPLS     DC    CL8'LIST'
NMLS     DC    C'(ZEDSMSG ZEDLMSG)' VDEFINE plist
FMAR     DC    CL8'CHAR'
         DC    CL8'CHAR'
LNAR     DC    A(L'SMSG)
         DC    A(L'LMSG)
VDAR     DS    0C
SMSG     DC    CL16' '             ISPF short msg variable
LMSG     DC    CL76' '             ISPF long msg variable
CTST     DC    XL8'0'              current MBD timestamp
CREC#    DC    F'0'                current rec nbr
CRECM    DC    F'0'                current MBD record
CREC     DC    CL128' '            current rec data
STORAR   DC    F'0'                dynamic storage pointer
REQRB    DC    F'0'                first buffer to be filled
REQRC    DC    F'0'                fill count
REQRN    DC    F'0'                requested rec nbr
REQRM    DC    F'0'                requested rec nbr in MDB
MDBREC   DC    F'0'                records in MDB
LTST     DC    XL8'0'              last rec MBD timestamp
LREC#    DC    F'0'                last rec nbr
LRECM    DC    F'0'                last MBD record
LREC     DC    CL128' '            last rec data
BUFFFST  DC    A(BUFF),A(0)        first display line in buffer
BUFFLST  DC    A(0),A(0)           last display line in buffer
BUFFA    DC    A(BUFF)             buffer address
NOTEA    DC    A(NOTE)             notelist address
CFLG     DC    X'0'                control flags
CFLGNF   EQU   X'80'               do not update first buffer pointer
CFLGEOF  EQU   X'40'               EOF (permanent)
* -----
CXSA     DC    18F'0'
CMDR     DC    A(BRCMDR)
VCOP     DC    CL8'VCOPY  '
NMZC     DC    C'(ZCMD)'           variable name
LAZC     DC    F'0'                variable length
VAZC     DC    F'0'                variable value addr
MOZC     DC    CL8'LOCATE'
         LTORG
NOTE#    EQU   100                 note interval
NOTECT   EQU   3000                note slot count
NOTE     POINTER 3000
BUFF#    EQU   201
BUFF     BUFFER  201,L'CREC
* ------ dsects
STRBUFF  DSECT                     buffer for log records
         ORG   *+STRBUFFL          length of buffer
*
         IXGANSAA LIST=YES         logger answer area
         PUSH  PRINT
         PRINT NOGEN
         IEAVG132 ,                mdb prefix
         IEAVM105 ,                mdb
         IHAHCLOG ,                hardcopy log format
         CVT   DSECT=YES           cvt
         POP   PRINT
* ------ equates
*        IXGCON ,                  System logger equates
         IXGCON DSECT=YES,LIST=YES
* ------ register usage
R0       EQU   0                   work reg
R1       EQU   1                   work and parm reg
R2       EQU   2                   work reg
R3       EQU   3                   work reg
R4       EQU   4                   work reg
R5       EQU   5                   .
R6       EQU   6                   pointer to end of the mdb
R7       EQU   7                   base for mdb objects
R8       EQU   8                   base for mdb
R9       EQU   9                   entry parameters and
*                                  base for hardcopy log record dsect
R10      EQU   10                  base for logger buffer
R11      EQU   11                  module base
R12      EQU   12                  module base
R13      EQU   13                  linkage
R14      EQU   14                  linkage
R15      EQU   15                  linkage
* ----- invoke in-storage-exec
IEAMDBCM CSECT
IEAMDBCM AMODE 31
IEAMDBCM RMODE ANY
         BAKR  R14,0               save regs
         LR    R12,R15             base
         USING IEAMDBCM,R12
         LA    R13,SAVC         .area..
         MVC   4(4,R13),=C'F1SA'   set acro in save area
* ----- invoke IRXEXEC to execute rexx
         MVI   AREA,C' '
         MVC   AREA+1(L'AREA-1),AREA
         LA    R1,PL               point at parameters
         OI    PLE,X'80'           set vl=1
         LINK  EP=IRXEXEC          invoke irxexec
         LTR   R15,R15             o.k?
         BNZ   QUIT4               nope - issue message
* ----- return rc = 0
         CLI   AREA,C' '           messages from rexx?
         BNE   QUIT8               yes - set message needed
         XR    R15,R15             set rc - o.k.
         B     QUIT
* ----- bye w/ rc
QUIT8    DS    0H
         LA    R15,8            set rc - 8
         B     QUIT
QUIT4    DS    0H
         MVC   AREA(28),=CL28'IRXEXEC invocation failed'
         LA    R15,4            set rc - 4
         B     QUIT
QUIT     DS    0H
         PR    ,
```

```
* ----- work e.t.c. areas
SAVC     DC    18F'0'
* ----- irxexec parameter list
PL       DC    A(P1,P2,P3,P4,P5,P6,P7,P8)
PLE      EQU   *-4
* ----- parameters for irxecex
P1       DC    A(EXECBL)         addr of exec-block
P2       DC    A(ARGBLK)         addr of argument-table
P3       DC    A(0)              flags
         ORG   P3
P3F0     DC    AL1(X'80')        bit 0 - exec invoked as command
         ORG
P4       DC    A(ISB)            in-storage-block - do not load exec
P5       DC    A(0)              addr of cppl - rquired in tso/e env
P6       DC    A(0)              no evaluation-block - n/a
P7       DC    A(WAD)            addr of work-area-definition
P8       DC    A(0)              no user-field
* ----- execblk
EXECBL   DS    0F
EXBI     DC    CL8'IRXEXECB'  Define execblk id, 'irxexecb'
EXBL     DC    A(EXBE-EXBI)   Length of execblk in bytes
         DC    F'0'           Reserved
EXBM     DC    CL8' '         The member name of the exec
EXBDD    DC    CL8' '         The dd from which the exec is
EXBS     DC    CL8'TSO'       Name of the initial subcommand
EXBDS    DC    A(0)           Pointer to a data set name (dsn)
EXBDL    DS    A(0)           Length of dsn pointed to by
EXBE     EQU   *              end of the execblk
* ----- argtable
ARGBLK   DS    0D             align on doubleword boundary
ATP      DC    A(A1B)         address of the argument string
ATL      DC    A(A1E-A1B)     length of the argument string
         DC    8X'FF'         eol
* ----- rexx exec parameters
A1B      DC    A(AREA)        arg str / 1 parm  - address to answer
         DC    C' '
         DC    A(AREAE-AREA)               2 parm  - lth of answer
A1E      DS    0C
*
AREA     DC    CL130' '       answer area element
AREAE    DC    C'**'
* ----- irxecex work area specification
WAD      DC    A(WAR,L'WAR)   WORK-AREA-DEFINITION
* ----- instblk
ISB      DS    0D            rexx in-storage block
IHDR     DS    0CL128        in-storage block header
         DC    CL8'IRXINSTB' the instblk identifier
IHDRL    DC    A(L'IHDR)     length of instblk header
         DC    F'0'          reserved
IISBEVA  DC    A(ISV)        address of first vector instblk_entry
IISBEVL  DC    A(ISVE-ISV)   total length of all isb-entry vector
IMEM     DC    CL8'IEAMDBCM' name of the exec
IDD      DC    CL8'IN_STOR'  name of dd for exec data set
ISUBC    DS    CL8'TSO '     name of initial subcommand env
         DC    F'0'          reserved
IDSNL    DC    F'0'          length of data set name
IDSN     DC    CL54' '       exec data set name, if known
         DC    H'0'          reserved
         DC    4F'0'         reserved - 4 words
* ----- vector of records of rexx stmts
ISV      DS    0D            the instblk_entry array
*
ISVSA    DC    A(RXS1)       address of rexx statement
ISVSL    DC    A(RXS1E-RXS1) length of the rexx statement
*
         DC    A(RXS2)       address of rexx statement
         DC    A(RXS2E-RXS2) length of the rexx statement
*
ISVE     DS    0F            end of the instblk_entry array
* ------ rexx statements - the exec
RXS1     DS    0C
         DC    C'TRACE "O";ADDRESS "TSO";'
         DC    C'PARSE ARG AA .;CA = C2X(AA);'
         DC    C'SD=SYSVAR(SOLDISP);UD=SYSVAR(UNSDISP);'
         DC    C'X=OUTTRAP(O.);'
         DC    C'"CONSPROF SOLDISP(NO) UNSOLDISP(NO)";'
         DC    C'"CONSOLE ACTIVATE";CR=RC;IF CR<8 THEN DO;'
         DC    C'"CONSOLE SYSCMD('
CMD      DC    CL126'   ',C')";'
         DC    C'IF CR<4 THEN  "CONSOLE DEACT";END;'
         DC    C'ELSE DO;X= "Command failed - '
         DC    C'CONSOLE ACTIVATE RC="CR;'
         DC    C'Y=STORAGE(CA,LENGTH(X),X);END;'
         DC    C'"CONSPROF SOLDISP("SD") UNSOLDISP("UD")";'
         DC    C'X=OUTTRAP("OFF");'
RXS1E    DS    0C
RXS2     DS    0C
         DC    C'EXIT CR;'
RXS2E    DS    0C
* ------
         LTORG
* ------ irxexec work are
WAR      DS    XL(X'1800')
* ------
         END   IEAMDBBR
```

C.2 REXX EXEC to Invoke Operlog Services

```
/* REXX */
trace "O"

/* This REXX EXEC provides a panel interface to the IEAMDBBR    */
/* operlog browse program and the IEAMDBLG sample program       */
/* (source in SYS1.SAMPLIB) that copies operlog stream data     */
/* in syslog format into a sequential data set or deletes       */
/* operlog stream data.                                         */

pgml = "VAINI.U.LOAD"            /* Default program library     */

address "ISPEXEC" "LIBDEF ISPLLIB DATASET ID('"pgml"')"
ieamdblg = "'"pgml"(IEAMDBLG)'"     /* IEAMDBLG CALL library     */
ieamdbbr = "IEAMDBBR"               /* BRIF interface pgm name   */

address "ISPEXEC" "CONTROL ERRORS RETURN"
address "ISPEXEC" "VGET (ZAPPLID) "

cvt = c2x( storage(10,4) )         /* Check MVS release         */
cvt_M20 = d2x( x2d(cvt) - x2d(20) )
FMID = storage(cvt_M20,8)
zedsmsg = ""
if FMID < "HBB5520" then do        /* MVS 5.2.0 required        */
  zedlmsg = "MVS 5.2.0 or higher required"
  address "ISPEXEC" "SETMSG MSG(ISRZ001)"
  exit 16
end
address "ISPEXEC" "VGET (ZENVIR)"  /* Check ISPF release        */
if substr(ZENVIR,6,1) <> 4 then do /* ISPF 4.1 required         */
  zedlmsg = "ISPF Version 4 or higher required"
  address "ISPEXEC" "SETMSG MSG(ISRZ000)"
  exit 16
end
go = 0
do while go = 0                    /* Main loop                 */
  CDAT = substr(date("S"),1,4)substr(date("J"),3)
  address "ISPEXEC" "DISPLAY PANEL(OPLPRIM)"
  go = RC
  zedsmsg = ""
  if go = 0 then do
    select
      when zcmd = "D" then do            /* Delete request */
        sdat = strip(sdat,"B")
        if sdat = "" then do             /* Set default    */
          sdat = '>0'
          zmemconv = "/"                 /* Force prompt!  */
        end
        if substr(sdat,1,1) <> ">" then dinfo = ,
                                         /* Delete prompt?  */
        "All records dated on or before" sdat "are deleted."
        else dinfo = ,
        "All records dated" sdat "days before today are deleted."
        if zmemconv <> "" then do       /* Delete prompt? */
          address "ISPEXEC" "ADDPOP ROW(10) COLUMN(5)"
```

```
      address "ISPEXEC" "DISPLAY PANEL(OPLPRMPT)"
      drc = RC
      address "ISPEXEC" "REMPOP"
    end
    else drc = 0
    if drc = 0 then do              /* Execute DELETE */
      "CALL" ieamdblg "'DELETE("sdat")'"
      if rc = 0 then  zedlmsg = dinfo
      else do
        if RC < 0 then rsn = "ABEND" d2x(0-RC)
        else rsn = RC
        zedlmsg = "IEAMDBLD program RC="rsn
      end
      address "ISPEXEC" "SETMSG MSG(ISRZ000)"
    end
    else do
      zedlmsg = "Records not deleted"
      address "ISPEXEC" "SETMSG MSG(ISRZ001)"
    end
  end
  when zcmd = "H" then do           /* Execute HARDCOPY*/
    if dsn = "" then ,              /* Consolidate dsn */
      cdsn = "'"strip(prj,"B")"."strip(lib,"B")"."strip(typ,"B") ||,
      "("strip(memb,"B")")'"
    else cdsn = dsn
    x = sysdsn(cdsn)                /* Extract dsn info*/
    y = ""
    if x  = "OK" | x = "MEMBER NOT FOUND" then do
      x = listdsi(cdsn)             /* Extract more info*/
      select                        /* Validate        */
        when sysdsorg = "???" then nop
        when sysdsorg = "PO" then do
          if pos("(",cdsn) = 0 then y = cdsn ,
          "is partitioned - membername not specified"
        end
        when sysdsorg = "PS" then nop
        otherwise y = cdsn ,
          "data set type" sysdsorg "not supported"
      end
      if sysrecfm ¬= "VB" & sysrecfm ¬= "??????" then y = cdsn ,
        "record format VB required - was" sysrecfm
      if syslrecl ¬= 132 & (syslrecl ¬= 0 & sysblksize ¬= 0) ,
        then y = cdsn "record lengt 132 required - was" syslrecl
    end
    else y = cdsn "data set -" x
    if y <> "" then do              /* Errors?         */
      zedlmsg = y
      address "ISPEXEC" "SETMSG MSG(ISRZ001)"
    end
    else do                         /* No errors - Copy */
      "ALLOC DD(SYSLOG) DS("cdsn") OLD REU"
      if RC = 0 then do
        y = "COPY"                  /* Build copy parms */
        if sdat <> "" then do
          y = y"("sdat
          if pos(">",sdat) = 0 then do
            if edat = "" then y = y","cdat")"
            else y = y","edat")"
          end
          else y = y")"
        end
        x = time("R")
        "CALL" ieamdblg "'"y"'"     /* Invoke COPY     */
        x = time("R")
        if rc = 0 then  do          /* Check results   */
          zedlmsg = "Requested records are copied" ,
            "- Start/end date("sdat edat") Elapsed time" x
          address "ISPEXEC" "SETMSG MSG(ISRZ000)"
          if bhcd <> "" then ,
          address "ISPEXEC" "BROWSE  DATASET("cdsn")"
        end
        else do
          if RC < 0 then rsn = "ABEND" d2x(0-RC)
          else rsn = RC
          zedlmsg = "IEAMDBLD program RC="rsn
          address "ISPEXEC" "SETMSG MSG(ISRZ000)"
        end
      "UNALLOC DD(SYSLOG)"
      end
      else do
        zedlmsg = cdsn "allocation error: RC="RC
        address "ISPEXEC" "SETMSG MSG(ISRZ001)"
      end
    end
  end
  when zcmd = "B" then do       /* Execute Browse    */
    y = "COPY"                  /* Build copy parms */
    if sdat <> "" then do
      y = y"("sdat
      if pos(">",sdat) = 0 then do
        if edat = "" then y = y","cdat")"
        else y = y","cdat")"
      end
      else y = y")"
    end
    address "ISPEXEC" "CONTROL DISPLAY SAVE"
    address "ISPEXEC" "SELECT  PGM("ieamdbbr") PARM("y")" ,
      "NEWAPPL("ZAPPLID") PASSLIB"
    address "ISPEXEC" "CONTROL DISPLAY RESTORE"
  end
  otherwise do
    zedlmsg = "Should not happen!"
    address "ISPEXEC" "SETMSG MSG(ISRZ001)"
  end
 end
end
```

C.3 ISPF Primary Panel (OPLPRIM)

The OPLPRIM panel is invoked from the REXX EXEC to request operlog data stream processing options.

```
)PANEL KEYLIST(ISRSAB,ISR)
)ATTR DEFAULT(  ) FORMAT(MIX))
 0B TYPE(AB)
 0D TYPE(PS)
 04 TYPE(ABSL)
 05 TYPE(PT)
 09 TYPE(FP)
 0A TYPE(NT)
 0C TYPE(NT) SKIP(ON)
 11 TYPE(SAC)
 19 TYPE(DT)
 22 TYPE(WASL) SKIP(ON)
 08 TYPE(CH)
 25 AREA(SCRL) EXTEND(ON)
 26 TYPE(NEF) CAPS(ON) PADC(USER)
 27 TYPE(CEF) PADC(USER) CKBOX(ON)
)ABC DESC('Menu') MNEM(1)
PDC DESC('Settings') UNAVAIL(ZPM1) MNEM(1) ACC(CTRL+S)
 ACTION RUN(ISRROUTE) PARM('SET')
PDC DESC('View') UNAVAIL(ZPM2) MNEM(1) ACC(CTRL+V)
 ACTION RUN(ISRROUTE) PARM('BR1')
PDC DESC('Edit') UNAVAIL(ZPM3) MNEM(1) ACC(CTRL+E)
 ACTION RUN(ISRROUTE) PARM('ED1')
PDC DESC('ISPF Command Shell') UNAVAIL(ZPM4) MNEM(6) ACC(CTRL+C)
 ACTION RUN(ISRROUTE) PARM('C1')
PDC DESC('Dialog Test...') UNAVAIL(ZPM5) MNEM(8) ACC(CTRL+T)
 ACTION RUN(ISRROUTE) PARM('DAL')
PDC DESC('Other IBM Products...') UNAVAIL(ZPM6) MNEM(1) ACC(CTRL+O)
 ACTION RUN(ISRROUTE) PARM('OIB')
PDC DESC('SCLM') UNAVAIL(ZPM7) MNEM(3) ACC(CTRL+L)
 ACTION RUN(ISRROUTE) PARM('SCL')
PDC DESC('ISPF Workplace') UNAVAIL(ZPM8) MNEM(6) ACC(CTRL+W)
 ACTION RUN(ISRROUTE) PARM('WRK')
PDC DESC('Status Area...') UNAVAIL(ZPMS) MNEM(8) ACC(CTRL+A)
```

```
 ACTION RUN(ISRROUTE) PARM('SAM')
PDC DESC('Exit') MNEM(2) ACTION RUN(EXIT)
)ABCINIT
.ZVARS=ISR@OPT
)ABC DESC('Help') MNEM(1)
PDC DESC('Library') MNEM(1) ACTION RUN(TUTOR) PARM('ISR31000')
PDC DESC('Data Set') MNEM(1) ACTION RUN(TUTOR) PARM('ISR32000')
PDC DESC('Move/Copy') MNEM(1) ACTION RUN(TUTOR) PARM('ISR33000')
PDC DESC('Data Set List') MNEM(2) ACTION RUN(TUTOR) PARM('ISR34000')
PDC DESC('Reset Statistics') MNEM(1) ACTION RUN(TUTOR) PARM('ISR35000')
PDC DESC('Hardcopy') MNEM(1) ACTION RUN(TUTOR) PARM('ISR36000')
PDC DESC('ISPF C/S') MNEM(1) ACTION RUN(TUTOR) PARM('ISPFTH07')
PDC DESC('Outlist') MNEM(1) ACTION RUN(TUTOR) PARM('ISR38000')
PDC DESC('Commands') MNEM(1) ACTION RUN(TUTOR) PARM('ISR39000')
PDC DESC('Format') MNEM(1) ACTION RUN(TUTOR) PARM('ISR3B000')
PDC DESC('SuperC') MNEM(1) ACTION RUN(TUTOR) PARM('ISR31200')
PDC DESC('SuperCE') MNEM(2) ACTION RUN(TUTOR) PARM('ISR31300')
PDC DESC('Search-for') MNEM(2) ACTION RUN(TUTOR) PARM('ISR314A0')
PDC DESC('Appendices') MNEM(2) ACTION RUN(TUTOR) PARM('ISR00004')
PDC DESC('Index') MNEM(2) ACTION RUN(TUTOR) PARM('ISR91000')
)ABCINIT
.ZVARS=UTILHELP
)BODY  CMD(ZCMD)
¢$ Menu$ Help¢
!-------------------------------------------------------------------------------
¢                              ¬OPERLOG Utility¢                                ¢
+Option ===>|Z                                                                  ¢
%SAREA37                                                                        %
%                                                                               %
)AREA SAREA37
@    BBrowse LOGR syslog data        ¢@    DDelete LOGR syslog data    ¢     ¢
@    HHardcopy LOGR syslog data      ¢@                                ¢     ¢

¢ ¢+Start date  .|Z      ¢ ¢+End date .  .|Z      ¢ ¢+Today's date.|Z      ¢

/Date format YYYYDDD or >nnn. Omit end date when start date specifies as >nnn. ¢

/ISPF Libray to receive syslog hardcopy data:/                                  ¢
¢ ¢+Project . . .|Z       ¢
¢ ¢+Group . . . .|Z       ¢
¢ ¢+Type  . . . .|Z       ¢
¢ ¢+Member  . . .|Z       ¢
¢
/Other Partitioned or Sequential Data Set:/                                     ¢
¢ ¢+Data Set Name . . .|Z
¢ ¢+Volume Serial . . .|Z     ¢  \(If not cataloged)  ¢
¢
+ ¢+&multipmt.                    ¢     "Z?@Confirm Delete       ¢             ¢
+                                       "Z?@Browse hardcopy LOGR syslog data   ¢
)INIT
.ZVARS = '(ZCMD SDAT EDAT CDAT PRJ LIB TYP MEMB  DSN VOL ZMEMCONV BHCD)'
.HELP = ISR00003
&ZCMD = ' '
&zut1 = 1
&ZMLCSR = '        '
IF ( &EDAT = ' ' )
  &EDAT = &CDAT
IF ( &SDAT = ' ' )
  &SDAT = &CDAT
ELSE
  &ZFC = TRUNC(&SDAT,1)
  IF (&ZFC = '>')
    &EDAT = ' '
IF ( &DSN = ' ' )
  &VOL =
IF ( &DSN ¬= ' ' )
  &MEMB = ' '
  .CURSOR = DSN
IF ( .CURSOR = ' ' )
  .CURSOR = ZCMD
&ZMEMCONV = '/'
IF (&ZGUI = ' ')
  &MULTIPMT='Enter "/" to select option    '
ELSE
  &MULTIPMT='Check box to select option    '
)REINIT
REFRESH(ZMEMCONV)
&zut1 = 1
IF (&ZMLCSR ¬= ' ')
  .CURSOR = &ZMLCSR
REFRESH (PRJ,LIB,TYP,MEMB,DSN,VOL,ZMEMCONV)
)PROC
&zut1 = 0
VER(&ZCMD,NB,LIST,B,H,D)
&ZFC = TRUNC(&SDAT,1)
IF (&SDAT ¬= ' ')
  IF (&ZFC = '>')
    &ZREM = .TRAIL
    VER (&ZREM,RANGE,0,999)
    IF (&ZCMD = 'D'.'H')
      IF (&ZFC = '>')
        &EDAT = ' '
  ELSE
    VER(&SDAT,PICT,9999999)
    &ZFC = TRUNC(&SDAT,4)
    VER (&ZFC,RANGE,1995,2100)
    &ZREM = .TRAIL
    VER (&ZREM,RANGE,1,366)
    &ZEDSMSG = 'Start date out of range'
    &ZEDLMSG = 'Start date greater than current date &CDAT'
    IF (&CDAT < &SDAT)
      .MSG = 'ISRZ001'
      EXIT
    VER (&EDAT,NB)
IF (&EDAT ¬= ' ')
  VER (&SDAT,NB)
  VER(&EDAT,PICT,9999999)
  &ZFC = TRUNC(&EDAT,4)
  VER (&ZFC,RANGE,1995,2100)
  &ZREM = .TRAIL
  VER (&ZREM,RANGE,1,366)
  IF (&CDAT < &EDAT)
    &EDAT = &CDAT
IF (&ZCMD = 'H')
  IF (&DSN = ' ')
    VER(&PRJ,NB)
    VER(&LIB,NB)
    VER(&TYP,NB)
    IF (&VOL ¬= ' ' )
      .MSG = ISRU232
    IF (&ZCMD = H)
      VER(&MEMB,NB)
  IF (&DSN ¬= ' ')
    &ZFC = TRUNC(&DSN,1)
    IF (&ZFC = '''')
      &ZREM = .TRAIL
      &ZREM2 = TRUNC(&ZREM,'''')
      IF (&ZREM2 = &ZREM)
        &DSN = '&DSN&ZFC'
    VER(&DSN,DSNAME)
&ZMEMCONV = TRANS(&ZMEMCONV,' '.' ' *,'/')
&ZMEMCONF = TRANS(&ZMEMCONV,' '.'OFF' *,'ON')
VPUT ( PRJ LIB TYP ZMEMCONF) PROFILE
)PNTS
FIELD(ZPS01001) VAR(ZCMD) VAL(B)
FIELD(ZPS01002) VAR(ZCMD) VAL(D)
FIELD(ZPS01003) VAR(ZCMD) VAL(H)
)END
```

NOTE:In the panel body hexadecimal values are
replaced with printable characters as follows:

X'04' - !	X'0A' - ¢	X'11' - @	X'26' - |
X'05' - ¬	X'0B' - $	X'19' - \	X'27' - "
X'08' - /	X'0C' - ?	X'22' -	
X'09' - +	X'0D' -	X'25' - %	

C.4 ISPF Prompt Panel (OPLPRMPT)

The OPLPRMPT panel is invoked fron the REXX EXEC to request operlog data stream data deletion confirmation.

```
)PANEL KEYLIST(ISRNAB ISR)
)ATTR DEFAULT(%?_)
  # TYPE(OUTPUT) INTENS(LOW) JUST(LEFT) CAPS(OFF) COLOR(GREEN)
  ¢ TYPE(NEF) PAD(USER) CAPS(ON)
   TYPE(PT)
   TYPE(NT)
  ¬ TYPE(PIN)
  ! TYPE(ET)
  @ TYPE(PSO)
  $ TYPE(VOI)
  ; TYPE(PIN)
  < TYPE(FP)
  ´ TYPE(CH)
  | TYPE(TEXT) INTENS(&MULTIPLE)
)BODY WINDOW(60,10)
?
<Command ===>¢ZCMD                                    ?
%
<$dinfo                                                 ?
?
´Instructions:
?
; Press!ENTER;key to confirm the delete request.
; Press!CANCEL;or!EXIT;to cancel the delete request.
)INIT
  &ZWINTTL = ´Confirm OPERLOG Record Delete´
  .HELP=ISR32030
  &ZCMD =
  .CURSOR = ZCMD
)PROC
  IF (&ZCMD ¬= &Z)
    .MSG = ISRU245
)END
```

Appendix D. Sample ARM Driver Program

The following sample programs allow a REGISTER and DEREGISTER to ARM without changing the application program code.

See Chapter 7.3, “ARM Capability without Program Changes” on page 123 for details.

D.1 ARM Driver Program 1

The following sample program generates the ARM element name from the jobname (first 8 bytes) and the stepname (second 8 bytes).

```
ARMDRVR  TITLE 'ARM DRIVER'
ARMDRVR  CSECT
ARMDRVR  AMODE 31
ARMDRVR  RMODE 24
*
*       This program creates an ARM registration and invokes the
*       batch program named on the EXEC card PARM field.
*       The PARM field syntax is:
*              PARM='program/program parameters'
*              - If the parameter data is improper, user abend 001
*                is issued.
*              - If the ARM registration fails, user abend 002
*                is issued.
*        The program to be executed may reside in a private library.
*
*       An example:
*
*    //VAINIAT JOB (999,POK),EXPERT,MSGLEVEL=1,MSGCLASS=A,
*    //        CLASS=A,NOTIFY=&SYSUID
*    //        EXEC PGM=ARMDRVR,PARM='IEFBR14/OTHER PARMS'
*    //STEPLIB DD DSN=VAINI.U.LOAD,DISP=SHR
*
*       Messages issued for the sample program:
*
*    IAT6140 JOB ORIGIN FROM GROUP=ANYLOCAL, DSP=IR , DEVICE=INTRDR
*    IRR010I  USERID VAINI    IS ASSIGNED TO THIS JOB.
*    06:30:03  IEF403I VAINIAT - STARTED - TIME=06.30.03
*    06:30:03  RC=0000 RSN=0000 - REGISTER REQUEST: VAINIATARMDRVR
*    06:30:03  RC=0000 RSN=0000 - READY REQUEST:    VAINIATARMDRVR
*    06:30:03  RC=0000 RSN=0000 - DEREGISTER RQUEST:VAINIATARMDRVR
*    06:30:03  IEF404I VAINIAT - ENDED - TIME=06.30.03
*
*       Messages issued for an abending program:
*
*    IAT6140 JOB ORIGIN FROM GROUP=ANYLOCAL, DSP=IR , DEVICE=INTRDR
*    IRR010I  USERID VAINI    IS ASSIGNED TO THIS JOB.
*    06:06:19  IEF403I VAINIAT - STARTED - TIME=06.06.19
*    06:06:19  CSV019I REQUESTED MODULE ABEN     NOT ACCESSED,
*                              IS IN NON-APF LIBRARY/CONCATENATION
*    06:06:19  CSV028I ABEND306-0C  JOBNAME=VAINIAT   STEPNAME=
*    06:06:19  RC=0000 RSN=0000 - REGISTER REQUEST: VAINIATARMDRVR
*    06:06:19  RC=0000 RSN=0000 - READY REQUEST:    VAINIATARMDRVR
*    06:06:19  IEA995I SYMPTOM DUMP OUTPUT
*    06:06:19    USER COMPLETION CODE=1234
*    ........  ............................
*    06:06:19  IEF450I VAINIAT - ABEND=S000 U1234 REASON=00000000
*    06:06:20  IEF404I VAINIAT - ENDED - TIME=06.06.20
*    06:06:20 IAT2006 PREMATURE JOB TERM - JOB VAINIAT(JOB20671) - HELD
*    06:06:20  IEF403I VAINIAT - STARTED - TIME=06.06.20
*    ........  ............................
*
*       Note: The ARMDRVR program must be APF authorized. When it
*       loads the program to be executed, an abend 306-C may occur
*       if the program resides in an non-APF program library.
*       This abend is recovered and the load is attempted as
*       unauthorized. If successful, the program execution resumes.
*
*
*       The ARM registration element name is a string of jobname
*       immediately followed by one of the following in the listed
*       order:
*           1) step name (if present)
*           2) procstep name (if present)
*           3) characters ARMDRVR
*
         SAVE  (14,12),,&SYSDATE.-&SYSTIME.-ARMDRVR
         LR    R12,R15            LOAD ENTRY ADDR
         USING ARMDRVR,R12
         LR    R2,R1              SAVE PARMS ADDR
       STORAGE OBTAIN,            GET WORKING STORAGE                  X
               LENGTH=WORKLEN1,                                        X
               LOC=(BELOW,ANY)
         ST    R1,8(,R13)         SAVE FORWARD POINTER TO THIS CSECT
         LR    R3,R13             SAVE SAVE AREA ADDR
         LR    R13,R1             SAVE ADDR OF AREA
         LR    R14,R1             LOAD ADDR OF WORK AREA TO BE ZEROED
         L     R15,=A(WORKLEN1)   LOAD LENGTH TO BE ZEROED
         SLR   R1,R1              SET PADDING BYTE & COUNT TO ZERO IN  X
                                  SOURCE (R0 WON'T MATTER)
         MVCL  R14,R0             PROPAGATE X'00' FROM PADDING BYTE
         ST    R3,4(,R13)         SAVE BACKWARD POINTER TO CALLER
         USING WORKAREA,R13       SAVE/WORK ADDRESSABILITY
         ST    R2,PPARM           SAVE PARMS ADDR
* ------ FIND PGMNM TO EXECUTE
         L     R1,0(,R2)          POINT AT PARM
         XR    R0,R0
         ICM   R0,3,0(R1)         ANY?
         BNZ   ARM002             YES - GOGO
ARM001   DS    0H
         LA    R0,20                 WTO TEXT LENGTH
         STH   R0,MSGTXTL
         MVC   MSGTXT(20),=CL20'BAD INPUT PARAMETERS'
         BAL   R9,ARMWTO
         B     ARMABN             USER ABEND
ARM002   DS    0H
         MVC   PGMNM,=CL40' '
         LR    R3,R0
         LA    R4,8
         LA    R5,PGMNM
         LA    R6,2(,R1)
ARM002A  DS    0H
         CLI   0(R6),C'/'
         BE    ARM002C
         MVC   0(1,R5),0(R6)
         LA    R5,1(,R5)
         LA    R6,1(,R6)
         BCT   R3,ARM002B
         XC    0(2,R1),0(R1)      NO USER PARM DATA
         B     ARM002D
ARM002B  DS    0H
```

```
         BCT   R4,ARM002A
         CLI   0(R6),C'/'
         BNE   ARM001            BAD PARM - ABEND
ARM002C  DS    0H
         CLI   PGMNM,C' '        PGM NAME?
         BE    ARM001            NO - ABEND
         BCTR  R3,0
         STH   R3,0(,R1)
         LTR   R3,R3             ANY USER PARM DATA?
         BZ    ARM002D           NO - DONE
         BCTR  R3,0
         MVC   2(*-*,R1),1(R6)   REFORMAT USRE PARM DATA
         EX    R3,*-6
ARM002D  DS    0H
         LA    R3,2(,R1)
         AH    R3,0(,R1)
         MVI   0(R3),0
* ------ SET ESTAE EXIT
         MVI   DSPRSU,0          INIT SWITCH
         MVI   DSRTRC,0          SET ESTAE RETRY COUNTER
         ST    R12,DSBASE        SAVE BASE FOR RETRY
         ST    R13,DSRR13        SAVE R13 FOR RETRY
         LA    R0,ARMLOR         SET..
         ST    R0,DSRETR         .RETRY ADDR
        MODESET MODE=SUP,KEY=ZERO SET SUPERVISOR STATE FOR ARM REQUESTS
         MVC   ESTAEL,ESTAEM
         LA    R2,ESTAE          POINT TO RECOVERY ROUTINE
         ESTAE (2),PARAM=(13),MF=(E,ESTAEL)
         LTR   R15,R15           ESTAE ACTIVE?
         BNZ   ARMFINA           NO - ABORT
        MODESET KEY=NZERO        SET NZERO KEY
         B     ARMLOD
* ------ DROP APF AUTHORIZATION
ARMLOR   DS    0H
         LA    R0,ARMFIN          SET..
         ST    R0,DSRETR          .RETRY ADDR
         SPKA  0
         L     R3,X'21C'(,0)      TCB ADDR.
         USING TCB,R3
         L     R15,TCBJSCB        JSCB ADDR.
         DROP  R3
         NI    X'EC'(R15),X'FF'-X'01'  ..APF AUTH OFF
         OI    DSPRSU,DSDAPF      DROP APF
         LPSW  NPSW
* ------ LOAD REQUESTED PROGRAM
ARMLOD   DS    0H
         LA    R1,PGMNM
        LOAD  EPLOC=(1)
         ST    R0,PADDR            SAVE EP ADDR
         CLM   R1,X'8'.=X'00'      APF TO BE DROPPED?
         BNE   *+4+4               NO
         OI    DSPRSU,DSDAPF       DROP APF
         LA    R0,ARMFIN           SET..
         ST    R0,DSRETR           .RETRY ADDR
* ------ INITIALIZE IXCARM ELEMENT NAME: SYSTEM NAME + JOBNAME
         MVC   ELEMNAME,=CL40' '
         L     R2,X'21C'(,0)         A(CURRENT TCB)
         L     R2,X'00C'(,R2)        A(TIOT)
         MVC   ELEJOBNM(8),X'0000'(R2) SET JOB NAME IN ELEMENT NAME
         LA    R1,ELEJOBNM+7
         CLI   0(R1),C' '            ELIMINATE..
         BNE   *+4+4                 .TRAILING..
         BCT   R1,*-4-4              ..BLANKS
         MVC   1(8,R1),ELESTEPN    SET DEFAULT IN ELEMENT NAME
         CLI   08(R2),C' '         ANY PROCSTEP NAME?
         BE    *+4+6               NO - JUMP
         MVC   1(8,R1),08(R2)      SET PROCSTEP IN ELEMENT NAME
         CLI   16(R2),C' '         ANY STEP NAME?
         BE    *+4+6               NO - JUMP
         MVC   1(8,R1),16(R2)      SET STEP IN ELEMENT NAME
         LA    R0,ARMMDS           SET..
         ST    R0,DSRETR           .RETRY ADDR
         OI    DSPRSU,DSSAPF       SET APF
         B     ARMREG
ARMMDS   DS    0H
         LA    R0,ARMFIN           SET..
         ST    R0,DSRETR           .RETRY ADDR
* ------ REQUEST=REGISTER
ARMREG   DS    0H
        MODESET MODE=SUP,KEY=NZERO
         NI    DSPRSU,X'FF'-DSSAPF   RESET SET APF
        IXCARM REQUEST=REGISTER,     GET REGISTERED                    X
               ELEMENT=ELEMNAME,     ELEMENT NAME                      X
               EVENTEXIT=EVTEXTNM,   EVENT EXIT NAME                   X
               EVENTEXITPL=EVTEXTPR, EVENT EXIT PARAMETER LIST         X
               EXITPLLEN=EVTEXTPL,   EVENT EXIT PARAMETER LIST LENGTH  X
               ELEMTYPE=ETYPE,       ELEMENT TYPE                      X
               RESTARTTIMEOUT=NORM,  NORMAL TIMEOUT INTERVAL           X
               ANSAREA=LCLANSWR,     ANSWER AREA                       X
               RETCODE=SAVERC,       RETURN CODE                       X
               RSNCODE=SAVERSN,      REASON CODE                       X
               MF=(E,IXCARML)        PARAMETER LIST AREA
* ------ SEE WHETHER REGISTERED, PERHAPS A RESTARTED JOB OR STC.
         BAL   R9,ARMRCRSN
         LA    R0,20+17+16           WTO TEXT LENGTH
         STH   R0,MSGTXTL
         MVC   MSGTXT+17(20),=CL20'- REGISTER REQUEST: '
         MVC   MSGTXT+17+20(16),ELEJOBNM
         BAL   R9,ARMWTO
         CLC   SAVERC,=A(IXCARMRC4)  RC=0 OR =4?
         BNH   ARM026                RC<=4 CONTINUE
         BAL   R9,AMSG30             RC> 4 ISSUE MESSAGE AND FAIL
         B     ARM999
* ------ DETERMINE WHETHER THIS IS A NEW REGISTRATION OR A REGISTRATION
*        AFTER BEING RESTARTED, AND ACT ACCORDINGLY.
ARM026   DS    0H
         CLC   SAVERC,=A(IXCARMRC0)  RC=0 (NOT RESTARTED)?
         BE    ARM028                YES, PROCEED
* ------ RESTART PROCESSING:  WAIT FOR ANY RESTARTED, PREDECESSOR
*        ELEMENTS REQUEST=WAITPRED
        IXCARM REQUEST=WAITPRED,     WAIT FOR ANY PREDECESSOR ELEMENTS  X
               RETCODE=SAVERC,       RETURN CODE                       X
               RSNCODE=SAVERSN,      REASON CODE                       X
               MF=(E,IXCARML)        PARAMETER LIST AREA
         BAL   R9,ARMRCRSN
         LA    R0,20+17+16           WTO TEXT LENGTH
         STH   R0,MSGTXTL
         MVC   MSGTXT+17(20),=CL20'- WAITPRED REQUEST:'
         MVC   MSGTXT+17+20(16),ELEJOBNM
         BAL   R9,ARMWTO
         CLC   SAVERC,=A(IXCARMRC4) RC=0 OR =4?
         BNH   ARM028                RC<=4 CONTINUE
         BAL   R9,AMSG30             RC>4 ISSUE MESSAGE AND FAIL
         B     ARM999
* ------ REQUEST=READY
ARM028   DS    0H
        IXCARM REQUEST=READY,        INDICATE READY                    X
               RETCODE=SAVERC,       RETURN CODE                       X
               RSNCODE=SAVERSN,      REASON CODE                       X
               MF=(E,IXCARML)        PARAMETER LIST AREA
         BAL   R9,ARMRCRSN
         LA    R0,20+17+16            WTO TEXT LENGTH
         STH   R0,MSGTXTL
         MVC   MSGTXT+17(20),=CL20'- READY REQUEST:'
         MVC   MSGTXT+17+20(16),ELEJOBNM
         BAL   R9,ARMWTO
         CLC   SAVERC,=A(IXCARMRC4) RC=0 OR =4?
         BNH   ARM030               SKIP IF OK
         BAL   R9,AMSG30            RC>4 ISSUE MESSAGE AND FAIL
         B     ARM999
* ------ DROP APF AUTHORIZATION
ARM030   DS    0H
         TM    DSPRSU,DSDAPF       DROP APF?
         BZ    ARMSMN
         LA    R0,ARM999           SET..
         ST    R0,DSRETR           .RETRY ADDR
         SPKA  0
         L     R3,X'21C'(,0)       TCB ADDR.
         USING TCB,R3
         L     R15,TCBJSCB         JSCB ADDR.
         DROP  R3
         NI    X'EC'(R15),X'FF'-X'01'  ..APF AUTH OFF
         LPSW  SPSW
ARMSMN   DS    0H
```

```
        MODESET MODE=PROB,KEY=NZERO  SET PROBLEM STATE
ARMPGO  DS    0H
* ------ INVOKE PROGRAM
        XC    DSRETR,DSRETR       NULLIFY RETRY
        L     R15,PADDR           PGMNM ENTRY ADDR
        L     R1,PPARM            SET PARMS ADDR
        BASR  R14,R15
        STH   R15,RCHALF          SAVE RC
        L     R15,=A(X'80000000'+ARM999)
        BASR  0,R15               SET 31-BIT AMODE AND EXIT
* ------ REQUEST=DEREGISTER
ARM999  DS    0H
        LA    R0,ARMRSA           SET..
        ST    R0,DSRETR           .RETRY ADDR
        OI    DSPRSU,DSSAPF       SET APF
        B     ARMDRE
ARMRSA  DS    0H
        LA    R0,ARMFIN           SET..
        ST    R0,DSRETR           .RETRY ADDR
ARMDRE  DS    0H
        MODESET MODE=SUP,KEY=NZERO SET SUPERVISOR STATE
        NI    DSPRSU,X'FF'-DSSAPF  RESET SET APF
        IXCARM REQUEST=DEREGISTER,  GET DEREGISTERED                X
              RETCODE=SAVERC,      RETURN CODE                      X
              RSNCODE=SAVERSN,     REASON CODE                      X
              MF=(E,IXCARML)       PARAMETER LIST AREA
        BAL   R9,ARMRCRSN
        LA    R0,20+17+16           WTO TEXT LENGTH
        STH   R0,MSGTXTL
        MVC   MSGTXT+17(20),=CL20'- DEREGISTER RQUEST:'
        MVC   MSGTXT+17+20(16),ELEJOBNM
        BAL   R9,ARMWTO
        CLC   SAVERC,=A(IXCARMRC0) RC=0?
        BNH   ARMRET               RC<=4 CONTINUE
        BAL   R9,AMSG30            RC>4 ISSUE MESSAGE AND CONTINUE
        B     ARMRET
* ------ EXIT PROCESSING
ARMRET  DS    0H
        MODESET MODE=PROB        BACK TO PROBLEM STATE
ARMSPL  DS    0H
        ESTAE 0                  DELETE ESTAE ENV.
        LR    R1,R13             POINT TO WORKING STORAGE
        L     R13,4(,R13)        SAVE CALLER'S R13
        LH    R11,RCHALF         SAVE RETURN CODE
        STORAGE RELEASE,         FREE WORKING STORAGE               X
              ADDR=(R1),           ADDRESS OF AREA TO BE FREED      X
              LENGTH=WORKLEN1
        XC    8(4,R13),8(R13)    CLEAR FORWARD POINTER OF CALLER
        LR    R15,R11            SET RETURN CODE
        RETURN (14,12),RC=(15)   RESTORE CALLER'S REGISTERS
* ------ USER ABENDS
ARMABN  DS    0H
        ABEND 1,DUMP
ARMFIN  DS    0H
*       ESTAE 0                  DELETE ESTAE ENV.
ARMFINA DS    0H
        ABEND 2,DUMP,STEP
* ------ MESSAGES
ARMWTO  DS    0H
        BAKR  R9,0
        MVC   WTOL(WTOLE-WTOL),WTOLM
        LA    R9,MSGTXTL
        WTO   TEXT=(9),MF=(E,WTOL)
        PR    ,
ARMRCRSN DS   0H
        MVC   MSGTXT+0(3),=CL3'RC='
        UNPK  MSGTXT+3(5),SAVERC+2(3)
        TR    MSGTXT+3(4),H2C-C'0'
        MVC   MSGTXT+3+4(5),=CL5' RSN='
        UNPK  MSGTXT+3+4+5(5),SAVERSN+2(3)
        TR    MSGTXT+3+4+5(4),H2C-C'0'
        MVI   MSGTXT+3+4+5+4,C' '
        BR    R9
AMSG30  DS    0H
        BAKR  R9,0
        LA    R0,12                WTO TEXT LENGTH
        STH   R0,MSGTXTL
        MVC   MSGTXT(12),=CL12'STOPPING...'
        BAL   R9,ARMWTO
        PR    ,                  RETURN TO MAINLINE
        DROP  R12
* ----- ESTAE TYPE RECOVERY ROUTINE.
ESTAE   DS    0H
        USING ESTAE,R15
        ST    R0,DSWRK-DSSA(,R2)
        SR    R0,R0
        C     R0,DSRETR-DSSA(,R2)                  SHOULD RETRY?
        BE    ESTAE2                               NO - PERCOLATE
        TM    DSPRSU-DSSA(R2),DSSAPF               SET APF?
        BZ    ESTAEA                               NO
        LR    R0,R2
        L     R2,X'21C'(,0)      TCB ADDR.
        USING TCB,R2
        L     R2,TCBJSCB         JSCB ADDR.
        DROP  R2
        OI    X'EC'(R2),X'01'    ..APF AUTH ON
        LR    R2,R0
        NI    DSPRSU-DSSA(R2),X'FF'-DSSAPF         RESET SET APF
ESTAEA  DS    0H
        SR    R0,R0
        IC    R0,DSRTRC-DSSA(R2)                   UPDATE ABEND..
        AH    R0,ESTAEH1                           .RECURSION..
        STC   R0,DSRTRC-DSSA(R2)                   ..COUNTER
        CLI   DSRTRC-DSSA(R2),25                   TOO BAD?
        BH    ESTAE2                               YES - SORRY!
        L     R0,DSWRK-DSSA(,R2)
        CH    0,=H'12'                 SDWA?
        BE    ESTAE1                   NO - GATHER INFO FROM REGS
        SETRP DUMP=NO,RC=4,RETADDR=ESTER,FRESDWA=YES TRY RETRY.
        BR    R14
ESTAE1  DS    0H
        LA    R15,4                                RC FOR RETRY.
        BR    R14                                  FIRST TO ABEND.
ESTAE2  DS    0H
        L     R0,DSWRK-DSSA(,R2)
        CH    0,=H'12'                 SDWA?
        BE    ESTAE3                   NO - GATHER INFO FROM REGS
        SETRP RC=0                                 ABORT THE S...
        BR    R14
ESTAE3  DS    0H
        SLR   R15,R15                              RC FOR ABORT.
        BR    R14                                  FIRST TO ABEND.
ESTAEH1 DC    H'1'
        DROP  R15
*  ----- ESTAE TYPE RETRY ROUTINE FOR FRESDWA=YES.
ESTER   DS    0H
        LR    R12,R15                TEMP BASE
        USING ESTER,R12
        LR    R13,R1                 POINT DYNAMIC STORAGE.
        USING DSSA,R13
        L     R12,DSBASE-DSSA(R13)   SET BASE REGS AS BEFORE.
        L     R15,DSRETR-DSSA(,R13)  PICK UP RETRY CON'T ADDR..
        L     R13,DSRR13-DSSA(,R13)  .AND SAVE ADDR..
        BR    R15                    ..AND GIVE IT A TRY.
        DROP  R12,R13
* ------ CONSTANTS - DATA AREAS
NPSW    DS    0D
        DC    X'078D1000'.AL4(X'80000000'+ARMLOD)
SPSW    DS    0D
        DC    X'078D1000'.AL4(X'80000000'+ARMPGO)
ETYPE   DC    CL8'XARMDRVR'        ELEMENT TYPE
EVTEXTNM DC   CL8'IEFBR14'         EVENT EXIT NAME
EVTEXTPR DS   0F                   EVENT EXIT PARAMETER LIST
        DC    C'IXCARM PARAMETER LIST'
EVTEXTLL EQU  *-EVTEXTPR           LENGTH OF PARAMETER LIST
EVTEXTPL DC   A(L'EVTEXTPR)        EVENT EXIT PARAMETER LIST LENGTH
H2C     DC    C'0123456789ABCDEF' HEX CHARS
WTOLM   WTO   TEXT=0,MF=L          DITTO
ESTAEM  ESTAE ,CT,ASYNCH=YES,PURGE=NONE,MF=L
ESTAEME DS    0H
* ------ LITERALS
        LTORG
* ------ SAVE/WORK AREA DSECT
WORKAREA DSECT
DSSA    DC    18F'0'               SAVE AREA     (DON'T CHANGE..
DSRR13  DS    A                    RESUME R13.   .THE ORDER..
```

```
DSRETR   DS    A                  RESUME ADDR.  ..THESE THREE FIELDS)
DSBASE   DS    F                  BASE REG FOR ESTAE RETRY.
DSABEND  DS    F                  ABEND CODE.
DSWRK    DS    F                  WORK
DSRTRC   DS    XL1                RETRY COUNT.
DSPRSU   DS    XL1                FLAG.
DSDAPF   EQU   X'40'              DROP APF
DSSAPF   EQU   X'20'              SET APF
         DS    0D               ALIGNMENT
* ------ IXCARM'S PARAMETER LIST AREA.
         IXCARM MF=(L,IXCARML)
PPARM    DS    F                  PROGRAM PARAMETERS
PADDR    DS    F                  PROGRAM ADDRESS
SAVERC   DS    F                  RETURN CODE FROM IXCARM SERVICE
SAVERSN  DS    F                  REASON CODE FROM IXCARM SERVICE
ELEMNAME DS    0CL16              ELEMENT NAME - TWO PARTS
ELEJOBNM DC    CL8' '             JOB NAME = 1ST PART ELEMENT NAME
ELESTEPN DC    CL8'ARMDRVR'       STEP NAME = 2ND PART ELEMENT NAME
PGMNM    DC    CL8' '             PROGRAM TO EXECUTE
RCHALF   DS    H                  RETURN CODE HALFWORD
LCLANSWR DS    XL32               ANSWER AREA
WTOL     WTO   TEXT=0,MF=L        DITTO
WTOLE    DS    0C
MSGTXTL  DS    H                  MESSAGE TEXT LENGTH
MSGTXT   DS    CL126              MESSAGE TEXT
         DS    0D                 DOUBLEWORD ALIGN END OF WORKAREA
ESTAEL   DS    XL(ESTAEME-ESTAEM) ESTAE WORK
WORKLEN1 EQU   *-WORKAREA         LENGTH OF WORKAREA
* ------ DSECTS, EQUATES
         PRINT NOGEN
         YREGS                    REGISTER EQUS
         IHAPSA                   PSA MAPPING
         IHAASCB                  ASCB MAPPING
         IKJTCB                   TCB MAPPING
         IEZJSCB                  JSCB MAPPING
         IEFJSSIB                 SSIB MAPPING
         IHASDWA
         IXCYARM
         IXCYARAA                 ARM ANSWER AREA MAPPING
         END  ARMDRVR
```

D.2 ARM Driver Program 2

The following sample program generates the ARM element name from a prefix (3 bytes of **$@$**, the jobname (8 bytes) and the first 5 bytes of the stepname.

This element name can be useful when defining policies. It's now possible to have a default policy for ARMDRVR jobs (element name $@$*).

```
ARMDRVR  TITLE 'ARM DRIVER'
ARMDRVR  CSECT
ARMDRVR  AMODE 31
ARMDRVR  RMODE 24
*
*       This program creates an ARM registration and invokes the
*       batch program named on the EXEC card PARM field.
*       The PARM field syntax is:
*              PARM='program/program parameters'
*              - If the parameter data is improper, user abend 001
*                is issued.
*              - If the ARM registration fails, user abend 002
*                is issued.
*        The program to be executed may reside in a private library.
*
*       An example:
*
*    //VAINIAT JOB (999,POK),EXPERT,MSGLEVEL=1,MSGCLASS=A,
*    //        CLASS=A,NOTIFY=&SYSUID
*    //        EXEC PGM=ARMDRVR,PARM='IEFBR14/OTHER PARMS'
*    //STEPLIB DD DSN=VAINI.U.LOAD,DISP=SHR
*
*       Messages issued for the sample program:
*
*    IAT6140 JOB ORIGIN FROM GROUP=ANYLOCAL, DSP=IR , DEVICE=INTRDR
*    IRR010I  USERID VAINI    IS ASSIGNED TO THIS JOB.
*    06:30:03  IEF403I VAINIAT - STARTED - TIME=06.30.03
*    06:30:03  RC=0000 RSN=0000 - REGISTER REQUEST: VAINIATARMDRVR
*    06:30:03  RC=0000 RSN=0000 - READY REQUEST:    VAINIATARMDRVR
*    06:30:03  RC=0000 RSN=0000 - DEREGISTER RQUEST:VAINIATARMDRVR
*    06:30:03  IEF404I VAINIAT - ENDED - TIME=06.30.03
*
*       Messages issued for an abending program:
*
*    IAT6140 JOB ORIGIN FROM GROUP=ANYLOCAL, DSP=IR , DEVICE=INTRDR
*    IRR010I  USERID VAINI    IS ASSIGNED TO THIS JOB.
*    06:06:19  IEF403I VAINIAT - STARTED - TIME=06.06.19
*    06:06:19  CSV019I REQUESTED MODULE ABEN     NOT ACCESSED,
*                              IS IN NON-APF LIBRARY/CONCATENATION
*    06:06:19  CSV028I ABEND306-0C  JOBNAME=VAINIAT   STEPNAME=
*    06:06:19  RC=0000 RSN=0000 - REGISTER REQUEST: VAINIATARMDRVR
*    06:06:19  RC=0000 RSN=0000 - READY REQUEST:    VAINIATARMDRVR
*    06:06:19  IEA995I SYMPTOM DUMP OUTPUT
*    06:06:19    USER COMPLETION CODE=1234
*    ::::::::  ::::::::::::::::::::::::::::
*    06:06:19  IEF450I VAINIAT - ABEND=S000 U1234 REASON=00000000
*    06:06:20  IEF404I VAINIAT - ENDED - TIME=06.06.20
*    06:06:20 IAT2006 PREMATURE JOB TERM - JOB VAINIAT(JOB20671) - HELD
*    06:06:20  IEF403I VAINIAT - STARTED - TIME=06.06.20
*    ::::::::  ::::::::::::::::::::::::::::
*
*       Note: The ARMDRVR program must be APF authorized. When it
*       loads the program to be executed, an abend 306-C may occur
*       if the program resides in an non-APF program library.
*       This abend is recovered and the load is attempted as
*       unauthorized. If successful, the program execution resumes.
*
*
*       The ARM registration element name is a 16 byte string of
*       a prefix ($@$), immediatedly followed by the jobname
*       immediately followed by one of the following in the listed
*       order:
*           1) step name     (if present - first 5 bytes)
*           2) procstep name (if present - first 5 bytes)
*           3) characters ADRVR
*
         SAVE  (14,12),,&SYSDATE.-&SYSTIME.-ARMDRVR
         LR    R12,R15            LOAD ENTRY ADDR
         USING ARMDRVR,R12
         LR    R2,R1              SAVE PARMS ADDR
       STORAGE OBTAIN,            GET WORKING STORAGE                 X
               LENGTH=WORKLEN1,                                       X
               LOC=(BELOW,ANY)
         ST    R1,8(,R13)         SAVE FORWARD POINTER TO THIS CSECT
         LR    R3,R13             SAVE SAVE AREA ADDR
         LR    R13,R1             SAVE ADDR OF AREA
         LR    R14,R1             LOAD ADDR OF WORK AREA TO BE ZEROED
         L     R15,=A(WORKLEN1)   LOAD LENGTH TO BE ZEROED
         SLR   R1,R1              SET PADDING BYTE & COUNT TO ZERO IN  X
                                  SOURCE (R0 WON'T MATTER)
         MVCL  R14,R0             PROPAGATE X'00' FROM PADDING BYTE
         ST    R3,4(,R13)         SAVE BACKWARD POINTER TO CALLER
         USING WORKAREA,R13       SAVE/WORK ADDRESSABILITY
         ST    R2,PPARM           SAVE PARMS ADDR
* ------ FIND PGMNM TO EXECUTE
         L     R1,0(,R2)          POINT AT PARM
         XR    R0,R0
         ICM   R0,3,0(R1)         ANY?
         BNZ   ARM002             YES - GOGO
ARM001   DS    0H
         LA    R0,20                 WTO TEXT LENGTH
         STH   R0,MSGTXTL
         MVC   MSGTXT(20),=CL20'BAD INPUT PARAMETERS'
         BAL   R9,ARMWTO
```

```
         B     ARMABN             USER ABEND
ARM002   DS    0H
         MVC   PGMNM,=CL40' '
         LR    R3,R0
         LA    R4,8
         LA    R5,PGMNM
         LA    R6,2(,R1)
ARM002A  DS    0H
         CLI   0(R6),C'/'
         BE    ARM002C
         MVC   0(1,R5),0(R6)
         LA    R5,1(,R5)
         LA    R6,1(,R6)
         BCT   R3,ARM002B
         XC    0(2,R1),0(R1)      NO USER PARM DATA
         B     ARM002D
ARM002B  DS    0H
         BCT   R4,ARM002A
         CLI   0(R6),C'/'
         BNE   ARM001             BAD PARM - ABEND
ARM002C  DS    0H
         CLI   PGMNM,C' '         PGM NAME?
         BE    ARM001             NO - ABEND
         BCTR  R3,0
         STH   R3,0(,R1)
         LTR   R3,R3              ANY USER PARM DATA?
         BZ    ARM002D            NO - DONE
         BCTR  R3,0
         MVC   2(*-*,R1),1(R6)    REFORMAT USRE PARM DATA
         EX    R3,*-6
ARM002D  DS    0H
         LA    R3,2(,R1)
         AH    R3,0(,R1)
         MVI   0(R3),0
* ------ SET ESTAE EXIT
         MVI   DSPRSU,0           INIT SWITCH
         MVI   DSRTRC,0           SET ESTAE RETRY COUNTER
         ST    R12,DSBASE         SAVE BASE FOR RETRY
         ST    R13,DSRR13         SAVE R13 FOR RETRY
         LA    R0,ARMLOR          SET..
         ST    R0,DSRETR          .RETRY ADDR
        MODESET MODE=SUP,KEY=ZERO SET SUPERVISOR STATE FOR ARM REQUESTS
         MVC   ESTAEL,ESTAEM
         LA    R2,ESTAE           POINT TO RECOVERY ROUTINE
         ESTAE (2),PARAM=(13),MF=(E,ESTAEL)
         LTR   R15,R15            ESTAE ACTIVE?
         BNZ   ARMFINA            NO - ABORT
        MODESET KEY=NZERO         SET NZERO KEY
         B     ARMLOD
* ------ DROP APF AUTHORIZATION
ARMLOR   DS    0H
         LA    R0,ARMFIN           SET..
         ST    R0,DSRETR           .RETRY ADDR
         SPKA  0
         L     R3,X'21C'(,0)       TCB ADDR.
         USING TCB,R3
         L     R15,TCBJSCB         JSCB ADDR.
         DROP  R3
         NI    X'EC'(R15),X'FF'-X'01'  ..APF AUTH OFF
         OI    DSPRSU,DSDAPF       DROP APF
         LPSW  NPSW
* ------ LOAD REQUESTED PROGRAM
ARMLOD   DS    0H
         LA    R1,PGMNM
        LOAD  EPLOC=(1)
         ST    R0,PADDR             SAVE EP ADDR
         CLM   R1,X'8',=X'00'       APF TO BE DROPPED?
         BNE   *+4+4                NO
         OI    DSPRSU,DSDAPF       DROP APF
         LA    R0,ARMFIN           SET..
         ST    R0,DSRETR           .RETRY ADDR
* ------ INITIALIZE IXCARM ELEMENT NAME: SYSTEM NAME + JOBNAME
         MVC   ELEMNAME,=CL40' '
         L     R2,X'21C'(,0)          A(CURRENT TCB)
         L     R2,X'00C'(,R2)         A(TIOT)
         MVC   ELEPRFX(3),ELPREFIX    Set element name prefix
         MVC   ELEJOBNM(8),X'000'(R2) SET JOB NAME IN ELEMENT NAME
         LA    R1,ELEJOBNM+7
         CLI   0(R1),C' '           ELIMINATE..
         BNE   *+4+4                .TRAILING..
         BCT   R1,*-4-4             ..BLANKS
         MVC   1(5,R1),ELSTEPNM     SET DEFAULT IN ELEMENT NAME
         CLI   08(R2),C' '          ANY PROCSTEP NAME?
         BE    *+4+6                NO - JUMP
         MVC   1(5,R1),08(R2)       SET PROCSTEP IN ELEMENT NAME
         CLI   16(R2),C' '          ANY STEP NAME?
         BE    *+4+6                NO - JUMP
         MVC   1(5,R1),16(R2)       SET STEP IN ELEMENT NAME
         LA    R0,ARMMDS            SET..
         ST    R0,DSRETR            .RETRY ADDR
         OI    DSPRSU,DSSAPF        SET APF
         B     ARMREG
ARMMDS   DS    0H
         LA    R0,ARMFIN            SET..
         ST    R0,DSRETR            .RETRY ADDR
* ------ REQUEST=REGISTER
ARMREG   DS    0H
        MODESET MODE=SUP,KEY=NZERO
         NI    DSPRSU,X'FF'-DSSAPF   RESET SET APF
        IXCARM REQUEST=REGISTER,     GET REGISTERED                    X
               ELEMENT=ELEMNAME,     ELEMENT NAME                      X
               EVENTEXIT=EVTEXTNM,   EVENT EXIT NAME                   X
               EVENTEXITPL=EVTEXTPR, EVENT EXIT PARAMETER LIST         X
               EXITPLLEN=EVTEXTPL,   EVENT EXIT PARAMETER LIST LENGTH  X
               ELEMTYPE=ETYPE,       ELEMENT TYPE                      X
               RESTARTTIMEOUT=NORM,  NORMAL TIMEOUT INTERVAL           X
               ANSAREA=LCLANSWR,     ANSWER AREA                       X
               RETCODE=SAVERC,       RETURN CODE                       X
               RSNCODE=SAVERSN,      REASON CODE                       X
               MF=(E,IXCARML)        PARAMETER LIST AREA
* ------ SEE WHETHER REGISTERED, PERHAPS A RESTARTED JOB OR STC.
         BAL   R9,ARMRCRSN
         LA    R0,20+17+16          WTO TEXT LENGTH
         STH   R0,MSGTXTL
         MVC   MSGTXT+17(20),=CL20'- REGISTER REQUEST: '
         MVC   MSGTXT+17+20(16),ELEMNAME
         BAL   R9,ARMWTO
         CLC   SAVERC,=A(IXCARMRC4)  RC=0 OR =4?
         BNH   ARM026                RC<=4 CONTINUE
         BAL   R9,AMSG30             RC> 4 ISSUE MESSAGE AND FAIL
         B     ARM999
* ------ DETERMINE WHETHER THIS IS A NEW REGISTRATION OR A REGISTRATION
*        AFTER BEING RESTARTED, AND ACT ACCORDINGLY.
ARM026   DS    0H
         CLC   SAVERC,=A(IXCARMRC0)  RC=0 (NOT RESTARTED)?
         BE    ARM028                YES, PROCEED
* ------ RESTART PROCESSING:  WAIT FOR ANY RESTARTED, PREDECESSOR
*        ELEMENTS REQUEST=WAITPRED
        IXCARM REQUEST=WAITPRED,    WAIT FOR ANY PREDECESSOR ELEMENTS  X
               RETCODE=SAVERC,      RETURN CODE                       X
               RSNCODE=SAVERSN,     REASON CODE                       X
               MF=(E,IXCARML)       PARAMETER LIST AREA
         BAL   R9,ARMRCRSN
         LA    R0,20+17+16           WTO TEXT LENGTH
         STH   R0,MSGTXTL
         MVC   MSGTXT+17(20),=CL20'- WAITPRED REQUEST:'
         MVC   MSGTXT+17+20(16),ELEMNAME
         BAL   R9,ARMWTO
         CLC   SAVERC,=A(IXCARMRC4) RC=0 OR =4?
         BNH   ARM028               RC<=4 CONTINUE
         BAL   R9,AMSG30            RC>4 ISSUE MESSAGE AND FAIL
         B     ARM999
* ------ REQUEST=READY
ARM028   DS    0H
        IXCARM REQUEST=READY,       INDICATE READY                    X
               RETCODE=SAVERC,      RETURN CODE                       X
               RSNCODE=SAVERSN,     REASON CODE                       X
               MF=(E,IXCARML)       PARAMETER LIST AREA
         BAL   R9,ARMRCRSN
         LA    R0,20+17+16           WTO TEXT LENGTH
         STH   R0,MSGTXTL
         MVC   MSGTXT+17(20),=CL20'- READY REQUEST:'
         MVC   MSGTXT+17+20(16),ELEMNAME
         BAL   R9,ARMWTO
         CLC   SAVERC,=A(IXCARMRC4) RC=0 OR =4?
         BNH   ARM030               SKIP IF OK
```

```
         BAL   R9,AMSG30            RC>4 ISSUE MESSAGE AND FAIL
         B     ARM999
* ------ DROP APF AUTHORIZATION
ARM030   DS    0H
         TM    DSPRSU,DSDAPF       DROP APF?
         BZ    ARMSMN
         LA    R0,ARM999           SET..
         ST    R0,DSRETR           .RETRY ADDR
         SPKA  0
         L     R3,X'21C'(,0)       TCB ADDR.
         USING TCB,R3
         L     R15,TCBJSCB         JSCB ADDR.
         DROP  R3
         NI    X'EC'(R15),X'FF'-X'01' ..APF AUTH OFF
         LPSW  SPSW
ARMSMN   DS    0H
        MODESET MODE=PROB,KEY=NZERO  SET PROBLEM STATE
ARMPGO   DS    0H
* ------ INVOKE PROGRAM
         XC    DSRETR,DSRETR       NULLIFY RETRY
         L     R15,PADDR           PGMNM ENTRY ADDR
         L     R1,PPARM            SET PARMS ADDR
         BASR  R14,R15
         STH   R15,RCHALF          SAVE RC
         L     R15,=A(X'80000000'+ARM999)
         BASR  0,R15               SET 31-BIT AMODE AND EXIT
* ------ REQUEST=DEREGISTER
ARM999   DS    0H
         LA    R0,ARMRSA           SET..
         ST    R0,DSRETR           .RETRY ADDR
         OI    DSPRSU,DSSAPF       SET APF
         B     ARMDRE
ARMRSA   DS    0H
         LA    R0,ARMFIN           SET..
         ST    R0,DSRETR           .RETRY ADDR
ARMDRE   DS    0H
        MODESET MODE=SUP,KEY=NZERO SET SUPERVISOR STATE
         NI    DSPRSU,X'FF'-DSSAPF  RESET SET APF
        IXCARM REQUEST=DEREGISTER,  GET DEREGISTERED                X
               RETCODE=SAVERC,      RETURN CODE                     X
               RSNCODE=SAVERSN,     REASON CODE                     X
               MF=(E,IXCARML)       PARAMETER LIST AREA
         BAL   R9,ARMRCRSN
         LA    R0,20+17+16          WTO TEXT LENGTH
         STH   R0,MSGTXTL
         MVC   MSGTXT+17(20),=CL20'- DEREGISTER RQUEST:'
         MVC   MSGTXT+17+20(16),ELEMNAME
         BAL   R9,ARMWTO
         CLC   SAVERC,=A(IXCARMRC0) RC=0?
         BNH   ARMRET              RC<=4 CONTINUE
         BAL   R9,AMSG30           RC>4 ISSUE MESSAGE AND CONTINUE
         B     ARMRET
* ------ EXIT PROCESSING
ARMRET   DS    0H
         MODESET MODE=PROB        BACK TO PROBLEM STATE
ARMSPL   DS    0H
         ESTAE 0                  DELETE ESTAE ENV.
         LR    R1,R13             POINT TO WORKING STORAGE
         L     R13,4(,R13)        SAVE CALLER'S R13
         LH    R11,RCHALF         SAVE RETURN CODE
         STORAGE RELEASE,         FREE WORKING STORAGE               X
               ADDR=(R1),           ADDRESS OF AREA TO BE FREED      X
               LENGTH=WORKLEN1
         XC    8(4,R13),8(R13)    CLEAR FORWARD POINTER OF CALLER
         LR    R15,R11            SET RETURN CODE
         RETURN (14,12),RC=(15)   RESTORE CALLER'S REGISTERS
* ------ USER ABENDS
ARMABN   DS    0H
         ABEND 1,DUMP
ARMFIN   DS    0H
*        ESTAE 0                  DELETE ESTAE ENV.
ARMFINA  DS    0H
         ABEND 2,DUMP,STEP
* ------ MESSAGES
ARMWTO   DS    0H
         BAKR  R9,0
         MVC   WTOL(WTOLE-WTOL),WTOLM
         LA    R9,MSGTXTL
         WTO   TEXT=(9),MF=(E,WTOL)
         PR    ,
ARMRCRSN DS    0H
         MVC   MSGTXT+0(3),=CL3'RC='
         UNPK  MSGTXT+3(5),SAVERC+2(3)
         TR    MSGTXT+3(4),H2C-C'0'
         MVC   MSGTXT+3+4(5),=CL5' RSN='
         UNPK  MSGTXT+3+4+5(5),SAVERSN+2(3)
         TR    MSGTXT+3+4+5(4),H2C-C'0'
         MVI   MSGTXT+3+4+5+4,C' '
         BR    R9
AMSG30   DS    0H
         BAKR  R9,0
         LA    R0,12                WTO TEXT LENGTH
         STH   R0,MSGTXTL
         MVC   MSGTXT(12),=CL12'STOPPING...'
         BAL   R9,ARMWTO
         PR    ,                    RETURN TO MAINLINE
         DROP  R12
* ----- ESTAE TYPE RECOVERY ROUTINE.
ESTAE    DS    0H
         USING ESTAE,R15
         ST    R0,DSWRK-DSSA(,R2)
         SR    R0,R0
         C     R0,DSRETR-DSSA(,R2)                  SHOULD RETRY?
         BE    ESTAE2                               NO - PERCOLATE
         TM    DSPRSU-DSSA(R2),DSSAPF               SET APF?
         BZ    ESTAEA                               NO
         LR    R0,R2
         L     R2,X'21C'(,0)       TCB ADDR.
         USING TCB,R2
         L     R2,TCBJSCB          JSCB ADDR.
         DROP  R2
         OI    X'EC'(R2),X'01'     ..APF AUTH ON
         LR    R2,R0
         NI    DSPRSU-DSSA(R2),X'FF'-DSSAPF         RESET SET APF
ESTAEA   DS    0H
         SR    R0,R0
         IC    R0,DSRTRC-DSSA(R2)                   UPDATE ABEND..
         AH    R0,ESTAEH1                           .RECURSION..
         STC   R0,DSRTRC-DSSA(R2)                   ..COUNTER
         CLI   DSRTRC-DSSA(R2),25                   TOO BAD?
         BH    ESTAE2                               YES - SORRY!
         L     R0,DSWRK-DSSA(,R2)
         CH    0,=H'12'                SDWA?
         BE    ESTAE1                  NO - GATHER INFO FROM REGS
         SETRP DUMP=NO,RC=4,RETADDR=ESTER,FRESDWA=YES TRY RETRY.
         BR    R14
ESTAE1   DS    0H
         LA    R15,4                                RC FOR RETRY.
         BR    R14                                  FIRST TO ABEND.
ESTAE2   DS    0H
         L     R0,DSWRK-DSSA(,R2)
         CH    0,=H'12'                SDWA?
         BE    ESTAE3                  NO - GATHER INFO FROM REGS
         SETRP RC=0                                 ABORT THE S...
         BR    R14
ESTAE3   DS    0H
         SLR   R15,R15                              RC FOR ABORT.
         BR    R14                                  FIRST TO ABEND.
ESTAEH1  DC    H'1'
         DROP  R15
*  ----- ESTAE TYPE RETRY ROUTINE FOR FRESDWA=YES.
ESTER    DS    0H
         LR    R12,R15                TEMP BASE
         USING ESTER,R12
         LR    R13,R1                   POINT DYNAMIC STORAGE.
         USING DSSA,R13
         L     R12,DSBASE-DSSA(R13)     SET BASE REGS AS BEFORE.
         L     R15,DSRETR-DSSA(,R13)    PICK UP RETRY CON'T ADDR..
         L     R13,DSRR13-DSSA(,R13)    .AND SAVE ADDR..
         BR    R15                      ..AND GIVE IT A TRY.
         DROP  R12,R13
* ------ CONSTANTS - DATA AREAS
NPSW     DS    0D
         DC    X'078D1000'.AL4(X'80000000'+ARMLOD)
SPSW     DS    0D
```

```
         DC    X'078D1000'.AL4(X'80000000'+ARMPGO)
ETYPE    DC    CL8'XARMDRVR'       Element type
ELPREFIX DC    CL3'$@$'            Element name prefix
ELSTEPNM DC    CL5'ADRVR'          Element Stepname default
EVTEXTNM DC    CL8'IEFBR14'        EVENT EXIT NAME
EVTEXTPR DS    0F                  EVENT EXIT PARAMETER LIST
         DC    C'IXCARM PARAMETER LIST'
EVTEXTLL EQU   *-EVTEXTPR          LENGTH OF PARAMETER LIST
EVTEXTPL DC    A(L'EVTEXTPR)       EVENT EXIT PARAMETER LIST LENGTH
H2C      DC    C'0123456789ABCDEF' HEX CHARS
WTOLM    WTO   TEXT=0,MF=L         DITTO
ESTAEM   ESTAE ,CT,ASYNCH=YES,PURGE=NONE,MF=L
ESTAEME  DS    0H
* ------ LITERALS
         LTORG
* ------ SAVE/WORK AREA DSECT
WORKAREA DSECT
DSSA     DC    18F'0'               SAVE AREA     (DON'T CHANGE..
DSRR13   DS    A                    RESUME R13.   .THE ORDER..
DSRETR   DS    A                    RESUME ADDR.  ..THESE THREE FIELDS)
DSBASE   DS    F                    BASE REG FOR ESTAE RETRY.
DSABEND  DS    F                    ABEND CODE.
DSWRK    DS    F                    WORK
DSRTRC   DS    XL1                  RETRY COUNT.
DSPRSU   DS    XL1                  FLAG.
DSDAPF   EQU   X'40'                DROP APF
DSSAPF   EQU   X'20'                SET APF
         DS    0D                  ALIGNMENT
* ------ IXCARM'S PARAMETER LIST AREA.
         IXCARM MF=(L,IXCARML)
PPARM    DS    F                    PROGRAM PARAMETERS
PADDR    DS    F                    PROGRAM ADDRESS
SAVERC   DS    F                    RETURN CODE FROM IXCARM SERVICE
SAVERSN  DS    F                    REASON CODE FROM IXCARM SERVICE
ELEMNAME DS    0CL16                ELEMENT NAME - TWO PARTS
ELEPRFX  DC    CL3'@$@'             Element name prefix
ELEJOBNM DC    CL8' '               JOB NAME  = 2nd PART ELEMENT NAME
ELESTEPN DC    CL5'ADRVR'           STEP NAME = 3ND PART ELEMENT NAME
PGMNM    DC    CL8' '               PROGRAM TO EXECUTE
RCHALF   DS    H                    RETURN CODE HALFWORD
LCLANSWR DS    XL32                 ANSWER AREA
WTOL     WTO   TEXT=0,MF=L          DITTO
WTOLE    DS    0C
MSGTXTL  DS    H                    MESSAGE TEXT LENGTH
MSGTXT   DS    CL126                MESSAGE TEXT
         DS    0D                   DOUBLEWORD ALIGN END OF WORKAREA
ESTAEL   DS    XL(ESTAEME-ESTAEM)   ESTAE WORK
WORKLEN1 EQU   *-WORKAREA           LENGTH OF WORKAREA
* ------ DSECTS, EQUATES
         PRINT NOGEN
         YREGS                     REGISTER EQUS
         IHAPSA                    PSA MAPPING
         IHAASCB                   ASCB MAPPING
         IKJTCB                    TCB MAPPING
         IEZJSCB                   JSCB MAPPING
         IEFJSSIB                  SSIB MAPPING
         IHASDWA
         IXCYARM
         IXCYARAA                  ARM ANSWER AREA MAPPING
         END  ARMDRVR
```

Index

Special Characters

Numerics

A

C

D

E

F

G

U

W

X

ITSO Technical Bulletin Evaluation RED000

International Technical Support Organization
MVS/ESA SP-JES3 Version 5
Implementation Guide
Release 5.1.1
Release 5.2.1
November 1995

Publication No. SG24-4582-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction	____		
Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

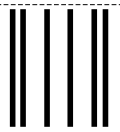
Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____ _____
Name Address

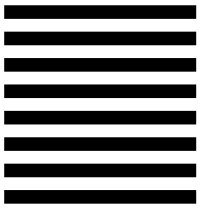
_____ _____
Company or Organization

_____ _____
Phone No.

Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Mail Station P099
522 SOUTH ROAD
POUGHKEEPSIE NY
USA 12601-5400

Fold and Tape **Please do not staple** Fold and Tape

Printed in U.S.A.